MODBUS. MONITOREO DE LA RED EMPLEANDO LABVIEW

SINLE MARCELA CARREÑO MARTÍNEZ
PEDRO ARDILA ALBARRACÍN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2005

MODBUS. MONITOREO DE LA RED EMPLEANDO LABVIEW

SINLE MARCELA CARREÑO MARTÍNEZ
PEDRO ARDILA ALBARRACÍN

Proyecto de grado para optar por el título de Ingeniero Electrónico.

Director
JULIO AUGUSTO GÉLVEZ FIGUEREDO
Magíster en Potencia Eléctrica

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2005

RESUMEN

TÍTULO: MODBUS. MONITOREO DE LA RED EMPLEANDO LABVIEW*

AUTORES: ARDILA ALBARRACÍN, PEDRO, y, CARREÑO MARTÍNEZ, SINLE MARCELA**

PALABRAS CLAVES: Modbus, sniffer, red, PLC, bus, RS-232, RS-485, Labview.

DESCRIPCIÓN:

El documento aquí presentado muestra el monitoreo de una red de PLC's que soportan el protocolo Modbus, el monitoreo se hace por medio de un computador en el cual se programó el protocolo como maestro con los dos modos de transmisión serial: ASCII y RTU, el programa se hizo en LabVIEW. También se desarrolló en LabVIEW un "SNIFFER" de la red que identifica las tramas Modbus, además de una aplicación donde se simula un proceso industrial. Para poder conectar el computador a la red se construyó un conversor de estándar RS-232 a RS-485.

En el primer capítulo se dan generalidades de estándares de comunicación serial que utiliza el protocolo Modbus. El segundo capítulo explica el funcionamiento del conversor RS-232 a RS-485. En el capítulo tres se exponen las características del protocolo Modbus y los campos del mensaje. El capítulo cuatro explica como se programó el Maestro Modbus en LabVIEW con las funciones más utilizadas del protocolo. El quinto capítulo presenta el Sniffer Modbus y por último en el capítulo seis se presenta como aplicación una red Modbus de PLC's donde se programaron en cada uno de ellos partes del proceso de una planta embotelladora, además se conectaron dos computadores, uno como un Maestro Modbus y el otro con el programa Sniffer.

El funcionamiento de la aplicación fue óptimo, evidenciando el buen desempeño de los programas y de los conversores.

---

* Trabajo de Grado.

** Facultad de Ingenierías Físico-mecánicas. Ingeniería Electrónica. Julio Augusto Gélvez Figueredo.

ABSTRACT

TITLE: MODBUS. MONITORING OF THE NETWORK USING LABVIEW*

AUTHORS: ARDILA ALBARRACÍN, PEDRO, y, CARREÑO MARTÍNEZ, SINLE MARCELA**

KEY WORDS: Modbus, sniffer, red, PLC, bus, RS-232, RS-485, Labview.

DESCRIPTION:

The present document show the monitoring of a network of PLC's that supports Modbus protocol, the monitoring is done via computer in which the protocol was programmed as master with both serial transmission modes: ASCII and RTU, the program was made in LabVIEW. Also they were developed in LabVIEW two programs: first, a sniffer of the network that identifies the frame modbus and second, an application where an industrial process is simulated. In order to be able to connect the computer to the network, a converter of standard RS-232 to RS-485 was constructed.

In the first chapter the generalities of the standards of serial communication that uses the Modbus protocol are shown. The second chapter explains the operation of converter RS-232 to RS-485. In chapter three the characteristics of the Modbus protocol and the fields of the message are exposed. Chapter four explains the programming of the Modbus Master in LabVIEW with the most used functions of the protocol.The fifth chapter presents the Sniffer Modbus and finally in chapter six, as a application, a Modbus network of PLC's was constructed, where parts of the process of a bottling plant were programmed in each one of them. Also, two computers were interconected to the PLC's network, one as a Modbus master and the other with the snilffer program.

The operation of the application was optimal, demonstrating the good performance of the programs and the converters.

---

* Work of Grade.

** Faculty of Engineering Physical-mechanics. Engineering Electronic. Julio Augusto Gélvez Figueredo.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

En la actualidad, las comunicaciones industriales revisten una vital importancia en la automatización, dado que con ellas se integran los procesos permitiendo controlar y supervisar en tiempo real y a distancias. Por ello se deben estudiar los tipos de redes y los protocolos que estas manejan. En el presente proyecto se programó en LabVIEW el Protocolo MODBUS como maestro y se desarrolló un sniffer que identifica las tramas Modbus. La razón para seleccionar el protocolo MODBUS es que es un protocolo abierto y de amplio uso en Colombia.

Se realiza una introducción a los estándares de comunicación serial RS 232, RS 485 y RS 422; debido a que los PLC's (TRILOGI, KOYO y NANO TELEMECANIQUE) y el PC poseen puertos con alguno de estos estándares seriales. Como la red estándar Modbus maneja el estándar RS 485 y el PC normalmente tiene el estándar RS 232 es necesario un conversor RS 232 a RS 485. En este documento se presentan los planos y la explicación del funcionamiento del conversor.

Desde el punto de vista académico, para la enseñanaza y comprensión de las funciones del protocolo MODBUS se diseñó una interfaz en LabVIEW que actúa como maestro de la red, en la que el usuario desde el panel frontal configura el modo de transmisión y los parámetros del puerto serial ajustándose a los que posee la red; además elige la Función Modbus en la cual se configura cada parámetro del formato del mensaje consulta (dirección del esclavo, dirección de inicio,.. y más). Si se recibe una respuesta, LabVIEW la captura y la visualiza en una ventana. Además se presenta un programa llamado SNIFFER MODBUS que captura la información que viaja por la red Modbus durante un intervalo de tiempo definido por el usuario y visualiza los mensajes consulta/respuesta de un esclavo específico.

Finalmente como aplicación, para validar el programa y verificar el correcto funcionamiento de los subvi Modbus para LabVIEW 6.1, se programó un proceso en tres PLC's de diferentes marcas y se configuró el PC como maestro de la red; en donde el PC a través de LabVIEW pregunta a cada dispositivo el estado de sus variables (Entradas, Bobinas, Registros), forzando algunas variables si se requiere y visualizando las entradas y salidas en una interfaz gráfica construida para este propósito.

Los programas y el conversor, se convierten en una herramienta que pueden ser usados por la escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones para la enseñanza e investigación en el campo de las comunicaciones industriales.

1. ESTÁNDARES DE COMUNICACIÓN SERIAL

1.1 ESTÁNDAR RS-232

Al principio de los años 60, un comité de estándares conocido como la Asociación de Industrias Electrónicas (EIA, *Electronics Industries Asociation*), desarrolló un estándar para la comunicación y conexión de dispositivos seriales, llamado RS-232. En 1991 la EIA en conjunto con la Asociación de Industrias de Telecomunicaciones (TIA, *Telecommunications Industry Association*) publicaron una nueva versión del estándar llamada EIA/TIA-232E.

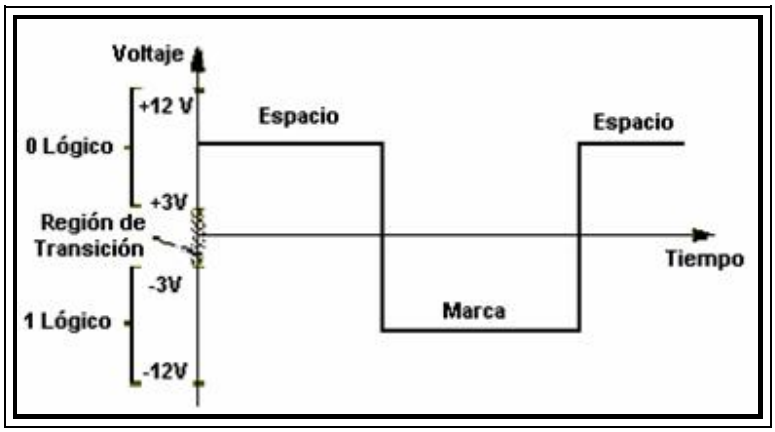
La comunicación entre dispositivos que usen el estándar RS-232 es full-duplex; donde se tiene una línea para el envío y otra para la recepción de datos. Con el estándar RS-232 solo se pueden conectar dos dispositivos; ésta comunicación se denomina punto a punto.
Hay dos términos que se deben tener en cuenta para el entendimiento de éste estándar: el Equipo Terminal de Datos DTE (Data Terminal Equipment) y el Equipo de Comunicación de Datos DCE (Data Communications Equipment).
El DTE es el dispositivo que recibe o envía información (ejemplo: el computador) y el DCE es el dispositivo que conecta el DTE a la línea de comunicación, convirtiendo las señales digitales en señales análogas (ejemplo: el módem).

1.1.1 Características Eléctricas

Las líneas que transportan las señales eléctricas en éste estándar son desbalanceadas ya que los voltajes en cada una de las líneas están referenciados a tierra. El estándar utiliza los siguientes rangos de voltaje para definir los estados lógicos, el rango entre -12V y -3V es un 1 lógico (conocido como condición de marca) y el rango entre +3V y +12V es un 0 lógico (conocido como condición de espacio). El rango de voltajes entre -3V y +3V es considerado como una región de transición conocida como “zona muerta” para la cual el

estado de la señal no está asignado, en esta zona se absorbe el ruido de la línea. (Ver Figura 1).

Figura 1. Rangos de voltaje y estados lógicos del estándar RS-232



RS 232/ EIA 232 Standard. ECE 351, Fall 2004. Página 25, reformada por los autores.

1.1.2 Líneas de Señal

El estándar RS-232 dispone de 25 líneas para la transmisión, pero muchas de estas líneas pertenecen a conexiones donde el dispositivo DCE es un módem; para cualquier dispositivo DCE que no sea un módem, o cuando dos dispositivos DTE son conectados directamente, se requieren de pocas líneas. En la Tabla 1 se hace una descripción de las líneas de las señales más utilizadas.

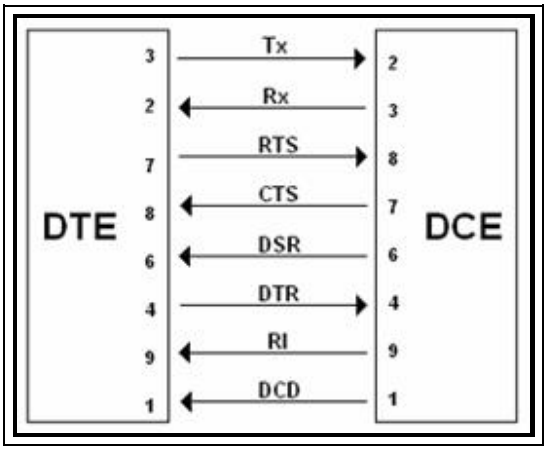
Tabla 1. Señales más utilizadas del RS-232

Señal	Función
GND	Esta línea proporciona la referencia de tierra para todas las demás señales.
Tx ***Transmitted data***	Por esta línea se transmiten los datos. Cuando no hay transmisión de datos, la señal se mantiene en la condición de marca (1 lógico, voltaje negativo).
Rx ***Received data***	Por esta línea se reciben los datos. Cuando no se están recibiendo datos, la señal se mantiene en la condición de marca (1 lógico, voltaje negativo).
DSR ***Data Set Ready***	Esta señal es sostenida en 0 lógico (voltaje positivo) por el DCE para indicar al DTE que está conectado a la línea
CTS ***Clear to Send***	Esta señal es sostenida en 0 lógico (voltaje positivo) por el DCE para informar al dispositivo DTE que la transmisión puede comenzar. RTS y CTS son usados comúnmente como señales para controlar el flujo de datos dentro del dispositivo DCE.
RTS ***Request to Send***	Esta señal es sostenida en 0 lógico (voltaje positivo) por el DTE para que el DCE se prepare a recibir el dato enviado. Tal preparación podría incluir la habilitación de los circuitos de recepción, o la colocación de la dirección del canal en aplicaciones half-duplex. Cuando el DCE está listo, éste acepta por medio de la línea CTS.
DTR ***Data Terminal Ready***	Esta señal es sostenida en 0 lógico (voltaje positivo) por el dispositivo DTE cuando éste quiere comenzar la comunicación.
RI ***Indicador Ring***	Esta señal es relevante cuando el dispositivo DCE es un módem, y es mantenido en 0 lógico (voltaje positivo) cuando una llamada está siendo recibida de la línea telefónica
DCD ***Data Carrier Detect***	Esta señal es sostenida en 0 lógico (voltaje positivo) por el DCE para avisar al DTE que está recibiendo una señal portadora con información.

Autores

La Figura 2 muestra la dirección de las señales más utilizadas en el estándar RS-232.

Figura 2. Dirección de las Señales más utilizadas en el estándar RS-232



Fuente: http://www.sinopticos.com/rs_232.html, reformada por los autores.

1.1.3 Características Físicas

El estándar sugiere una longitud máxima de cobre de 15 metros y una velocidad de transmisión máxima de 19200 baudios. Usualmente se ignoran estos límites cuando se usan cables de alta calidad y bien blindados.

Tipos de Conectores.

- **DB-25**. El conector estándar es el conocido como DB-25, que tiene 25 pines (ver Figura 3).

Figura 3. Conector DB-25, conector macho (a) y conector hembra (b).

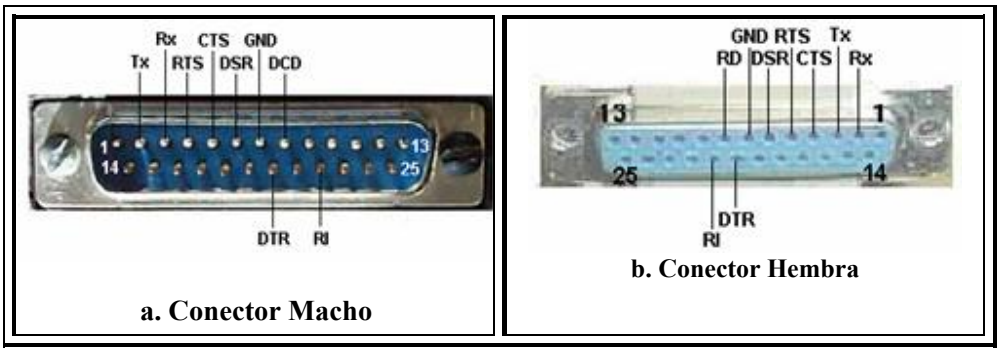
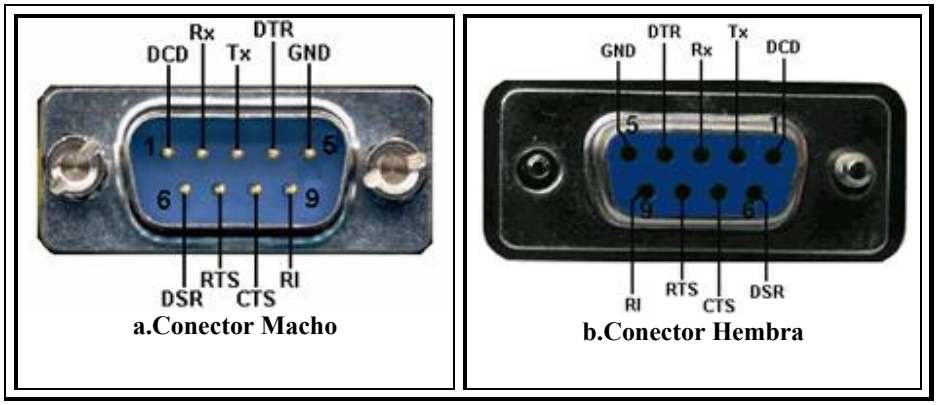


Figura a http://www.meierspage.net/how2s/modding_faq/lcd_faq/lcd_faq.php reformada por los autores.
Figura b. http://ryan.com.br/cabo_dc20.htm, reformada por los autores.

Debido al hecho que RS-232 define señales que normalmente no son empleadas, y al gran tamaño del conector DB-25, los fabricantes modifican el estándar para usar conectores más pequeños y eliminar líneas que no son utilizadas. Entre ellos se encuentran los siguientes:

- **DB-9.** Este tipo de conector se encuentra comúnmente en los PC's. Consta de nueve pines (ver Figura 4).

Figura 4. Conector DB9, conector macho (a) y conector hembra (b).



Fuente: Figura a. http://www.arcelect.com/rs232.htm , reformada por los autores. Figura b http://jlomicka.home.comcast.net/mr26a/maccable.html reformada por los autores.

- **DIN-8.** El conector DIN-8 es pequeño y casi circular. Es usado en Macs, y por ser compacto es a menudo utilizado en laptops. Él provee siete de las señales más comunes de una comunicación serial.

Figura 5. Conector DIN-8, conector macho (a) y conector hembra (b).

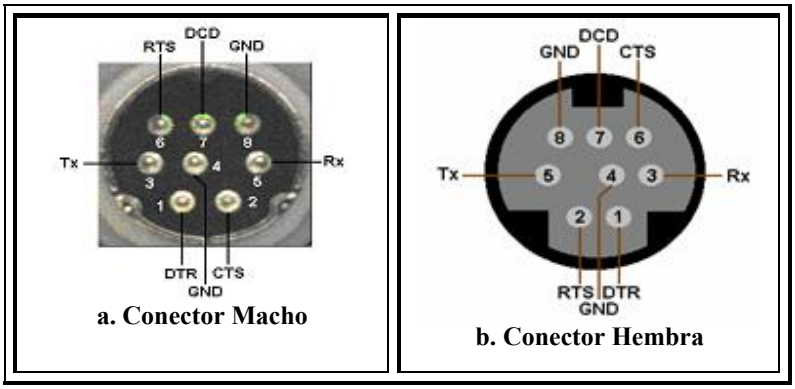


Figura a. http://francis.courtois.free.fr/jc1/serial/Basics/DIN8.html reformada por los autores. Figura b, autores.

Puede ser utilizado con RS-232 RS-485 o RS-422, cambiándose la asignación en cada pin. La Figura 5 muestra el conector con la designación de pines para RS-232.

RJ-45. Los conectores RJ-45 tienen ocho alambres. Debido a que son pequeños, ellos son a menudo usados en dispositivos que tienen muchos puertos juntos. Los terminales de servidores son un buen ejemplo de dispositivos que usan conectores RJ-45. La figura 6 muestra el conector RJ-45 y la designación de pines para RS-232.

Figura 6. Conector RJ-45 (a), designación de pines (b)

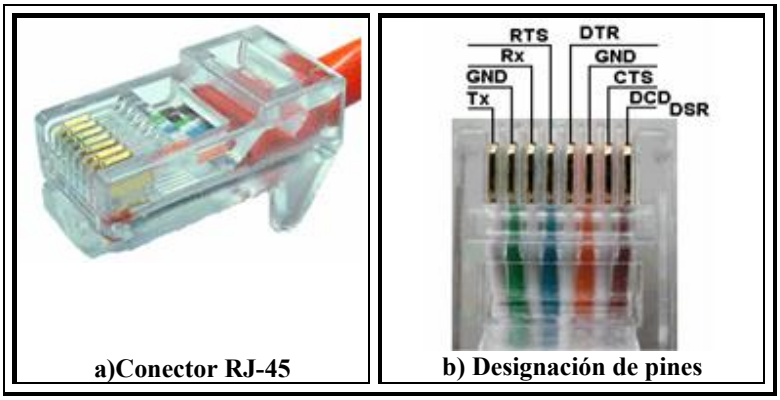


Figura a) http://www.utexas.edu/its/datacomm/connectors.html,
Figura b) http://www.coloredhome.com/cable_cruzado/cable_cruzado0002.htm, reformada por los autores.

1.1.4 Transmisión de Bits

El estándar RS-232 se basa en comunicación asíncrona, es decir, los datos no requieren una señal de sincronismo, por lo que deben tomarse precauciones para sincronizar la transmisión con la recepción.

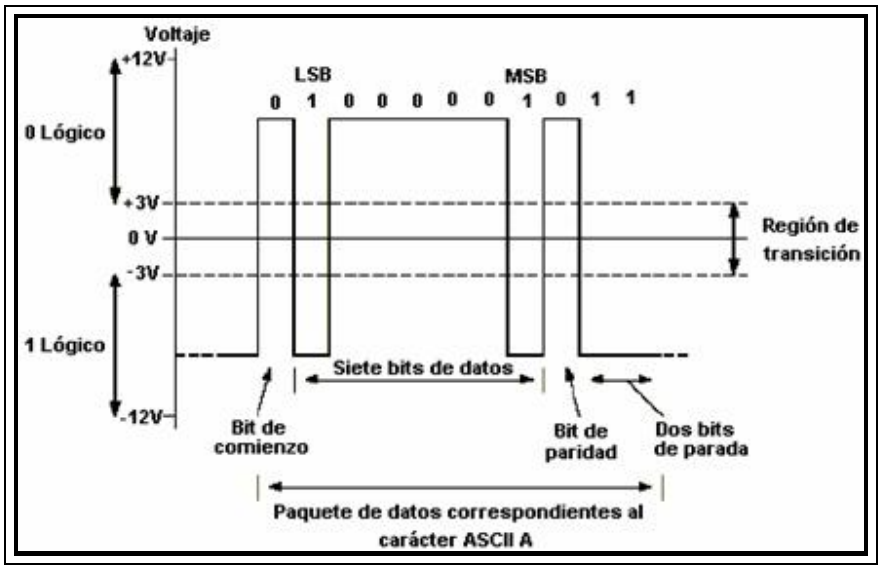
Cuando no hay transmisión la línea Tx se mantiene en estado latente (voltaje negativo, 1 lógico) y la información se inicia enviando un bit (conocido como bit de comienzo y que siempre es un 0 lógico), seguido de 5, 6,7 u 8 bits de datos, un bit adicional de paridad y 1, 1.5 o 2 bits de parada (que son 1's lógicos). Esta secuencia permite reconocer el inicio y el fin de la transmisión, los datos, y la integridad de los mismos.

Una vez que ha comenzado la transmisión de un dato, los bits tienen que llegar uno detrás

de otro a una velocidad constante (velocidad de transmisión) y en determinados instantes de tiempo. Por eso, se dice que el RS-232 es asíncrono por carácter y síncrono por bit.

La Figura 7 muestra el envío del carácter "A"en ASCII, con 1 bit de comienzo, 7 bits de datos, 1 bit de paridad y 2 bits de parada.

Figura 7. Niveles de tensión en RS-232 para el carácter A en ASCII



Carácter A, en código ASCII = 65 (decimal) y en código binario = 1000001.
Fuente: RS 232/ EIA 232 Standard. ECE 351, Fall 2004 (PDF). Página 26. Reformada por los autores.

1.1.5 Desventajas de RS-232

Aunque RS-232 es simple, universal, bien documentado y soportado en todas partes, tiene algunas desventajas:

- Solo puede comunicar dos dispositivos entre sí.
- Empleado para distancias cortas y velocidades bajas.
- No es inmune al ruido porque a una línea referenciada a tierra es imposible bloquearle efectivamente el ruido. Por medio del blindaje se puede reducir la influencia de ruido exterior, pero restos del ruido internamente generado son un problema.

- A medida que la velocidad de transferencia de datos y la longitud se incrementan, el efecto de la capacitancia entre los cables introduce serias interferencias hasta que se puede alcanzar el punto en el cual el dato es imposible de leer.

En la práctica, la longitud del cable puede ser extendida hasta 30m para cable blindado y aterrizado, y hasta 100m si el cable posee además baja capacitancia.
Las desventajas de RS-232 son en su mayor parte superadas por los estándares de línea balanceada. Entre ellos se encuentran RS-422 y RS-485.

1.2 ESTÁNDARES DE LÍNEA BALANCEADA

Las líneas balanceadas son aquellas que transportan las señales eléctricas de forma diferencial. La transmisión diferencial de datos se usa para la transmisión confiable de información a altas velocidades y sobre grandes distancias y a través de ambientes ruidosos. Se usa un par de alambres para llevar cada señal, la información se envía y recibe en cada par, como un voltaje diferencial entre éstos, comúnmente llamados A (voltaje negativo) y B (voltaje positivo).

Los estados lógicos en las líneas A y B se definen como sigue:

- Cuando el voltaje de la línea B es negativo con respecto al voltaje de la línea A, el estado lógico es 1.
- Cuando el voltaje de la línea B es positivo con respecto al voltaje de la línea A, el estado lógico es 0

Al dispositivo que envía la información por medio de un voltaje diferencial a través de las líneas A y B se le denomina *driver* (D) y al dispositivo que recibe el voltaje diferencial, *receiver* (R). Un esquema simple de línea balanceada se muestra en la Figura 8.

Figura 8. Esquema de línea balanceada

Fuente: http://www.sangoma.com/signal.htm. Reformada por los autores.

Aunque en líneas balanceadas no se necesita la señal de tierra para el envío y recepción de la información, sí se necesita para mantener el voltaje en modo común, Vcm, en el *receiver* dentro de un rango seguro. El voltaje en modo común se define como el voltaje medio de los terminales A y B con respecto a la señal de tierra, esto es: $\frac{V_A + V_B}{2} = V_{CM}$.

Dado que la línea balanceada opera con un voltaje diferencial, no es afectada por la diferencia de voltajes entre la tierra del emisor y la tierra del receptor.
El cable utilizado comúnmente en el envío de información a través de líneas balanceadas es el "par trenzado", debido a la notable reducción de la interferencia eléctrica que pueda haber a su alrededor.

Cuando la impedancia de la fuente, de la línea de transmisión y de la carga no son iguales, parte de la señal transmitida al llegar al final del bus se refleja hacia atrás dentro de la línea de transmisión. Estas reflexiones llegan a producir problemas cuando se envían datos a través de grandes distancias y a altas velocidades, la forma de eliminarlas es colocando al final del bus una impedancia apropiada en paralelo a las líneas A y B; éste método se conoce como Terminación. A menudo se coloca en vez de una impedancia, una resistencia de terminación llamada Rt. Para una mayor información sobre cómo usar la terminación y que inconvenientes conlleva se puede remitir al pdf RS-422/485 *Application Note de B&B Electronics.*

Hay dos estándares principales aprobados por la EIA para la transmisión diferencial de datos: el RS-422 y el RS-485.

1.2.1 Estándar RS-422

El estándar RS-422 se desarrolló para enviar los datos a mayores velocidades y a través de mayores distancias de lo que es posible con RS-232. Con RS-422 la velocidad máxima de transferencia de datos es 10Mbits/s y la máxima longitud del cable es 1200m.
El rango de modo común para el receiver en RS-422 es -7V < Vcm < +7V.
La resistencia de entrada en el receiver debe ser como mínimo de 4kΩ.
Los niveles de tensión entre las líneas A y B a la salida del driver y la entrada del receiver se especifican en la Tabla 2.

Tabla 2. Niveles de Tensión para RS-422

Driver		Receiver	
1 Lógico	0 Lógico	1 Lógico	0 Lógico
$V_B - V_A < -2$ V	$V_B - V_A > 2$ V	$V_B - V_A < -0.2$ V	$V_B - V_A > 0.2$ V

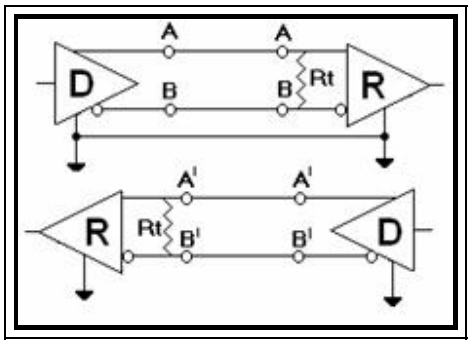
Conexión de RS-422

El estándar RS-422 permite la conexión punto a punto entre dos terminales.

Conexión punto a punto

En una comunicación punto a punto utilizando el estándar RS-422 se necesitan dos líneas para emisión, dos líneas de recepción y la línea de tierra (comunicación full-duplex), además de la utilización de 2 drivers y dos receivers, como se muestra en la Figura 9.

Figura 9. Configuración punto a punto con el estándar RS-422



Fuente: RS-422 and RS-485 Application Note B&B Electronics, página 7, reformada por los autores.

1.2.2 Estándar RS-485

Para mantener el envío confiable de datos a través de grandes distancias y a altas velocidades, y superar la limitación que tiene el estándar RS-422 de no poder conectarse con topología de bus a varios terminales, se desarrolló el estándar RS-485.

En RS-485 como en RS-422 la velocidad máxima de transferencia de datos es 10Mbits/s y la máxima longitud del cable es 1200m. El rango en modo común del *receiver* aumenta a -7V < Vcm < +12V.
La resistencia de entrada en el receiver debe ser como mínimo de 12kΩ.
Los niveles de tensión entre las líneas A y B a la salida del driver y la entrada del receiver se especifican en la Tabla 3.

Tabla 3. Niveles de Tensión para RS-485.

Driver		Receiver	
1 Lógico	0 Lógico	1 Lógico	0 Lógico
$V_B - V_A < -1.5$ V	$V_B - V_A > 1.5$ V	$V_B - V_A < -0.2$ V	$V_B - V_A > 0.2$ V

Conexión de RS-485

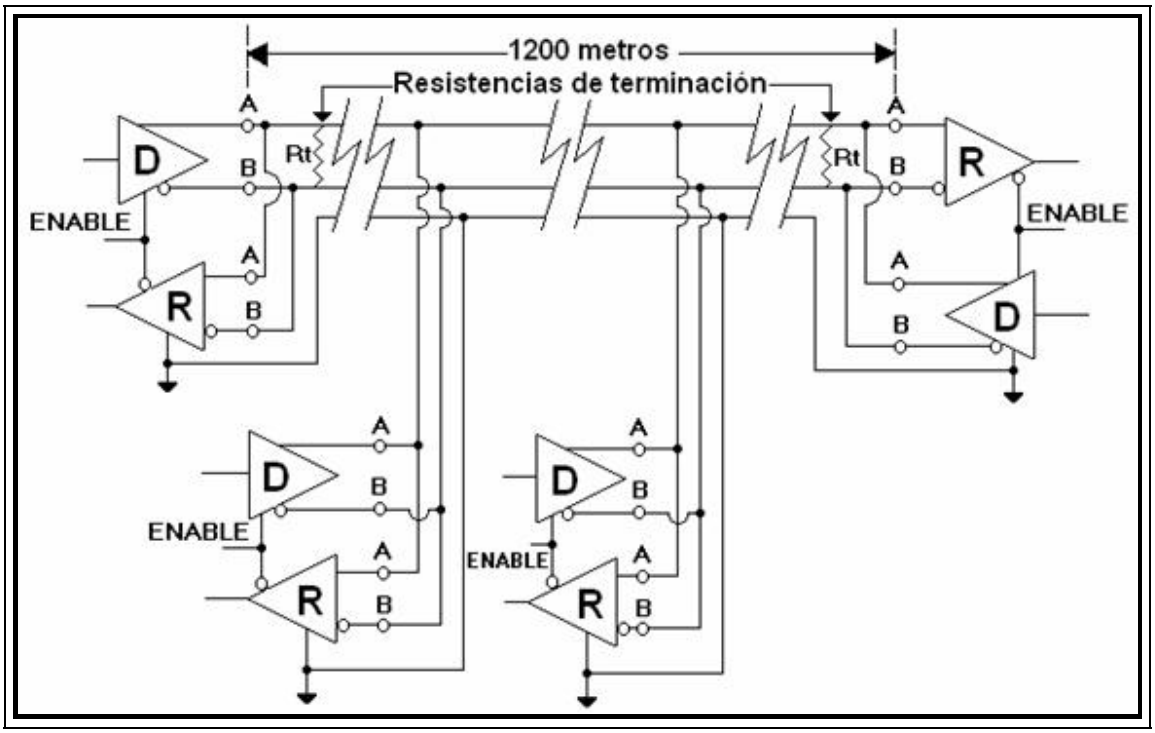
El estándar permite la conexión con topología de bus, conectándose así a varios dispositivos (comunicación multipunto); usando el mismo par de hilos para la transmisión y la recepción (comunicación *half-duplex*). Esto requiere un control en la conmutación de la línea, esta conmutación se hace por medio de la habilitación de los *drivers* y los *receivers*.
El número máximo de dispositivos que pueden ser conectados al bus es de 32.

Conexión Multipunto

Por medio de esta conexión cada dispositivo puede enviar y recibir información hacia y desde todos y cada uno de los demás dispositivos conectado al bus. La conexión multipunto utilizando el estándar RS-485 se aprecia en la Figura 10.

Cuando un dispositivo desea enviar información su driver debe estar habilitado y los drivers de los demás dispositivos no, además el receiver del dispositivo que envía información se deshabilita y los demás receivers en el bus tienen que habilitarse. Esta configuración permite que cualquier dispositivo en un momento dado sea un maestro y los demás esclavos.

Figura 10. Configuración Multipunto de Dos hilos para RS-485.



RS-422 and RS-485 Application Note B&B Electronics, página 10, reformada por los autores.

2. CONVERSOR DE ESTÁNDAR RS-232 A RS-485

En este trabajo de grado se busca conectar el computador como maestro ó como *sniffer* (Chismoso) de la red Modbus según se requiera, para ello se deben acoplar las señales eléctricas que envía y recibe el computador por su puerto serial (COM1) utilizando el estándar RS-232 con las señales eléctricas de la red Modbus, la cual utiliza para su comunicación el estándar RS-485. Para intercomunicar el computador y la red Modbus se pudo haber comprado un conversor RS-232 a RS-485, pero se optó por construir uno; no solo por disminuir costos sino por el hecho de incrementar los conocimientos de éstos estándares y su uso en las redes industriales.

El conversor no hace directamente el cambio de los niveles de tensión de RS-232 a RS-485, sino que hace un paso intermedio a lógica TTL/CMOS. La tabla número 4 muestra los niveles de tensión cuando se hace la conversión de RS-232 a RS-485.

Tabla 4. Niveles de tensión del conversor para el cambio de RS-232 a RS-485

Estados	Niveles de Tensión		
	RS-232	TTL/CMOS	RS-485
0 Lógico	3V < L < 12V	0V < L < 0.8V	$V_B - V_A > 2V$
1 Lógico	-12V < H < -3V	3.5 V < H < 5 V	$V_B - V_A < -2V$

La tabla número 5 muestra los niveles de tensión cuando se hace la conversión de RS-485 a RS-232.

Tabla 5. Niveles de tensión del conversor para el cambio de RS-485 a RS-232

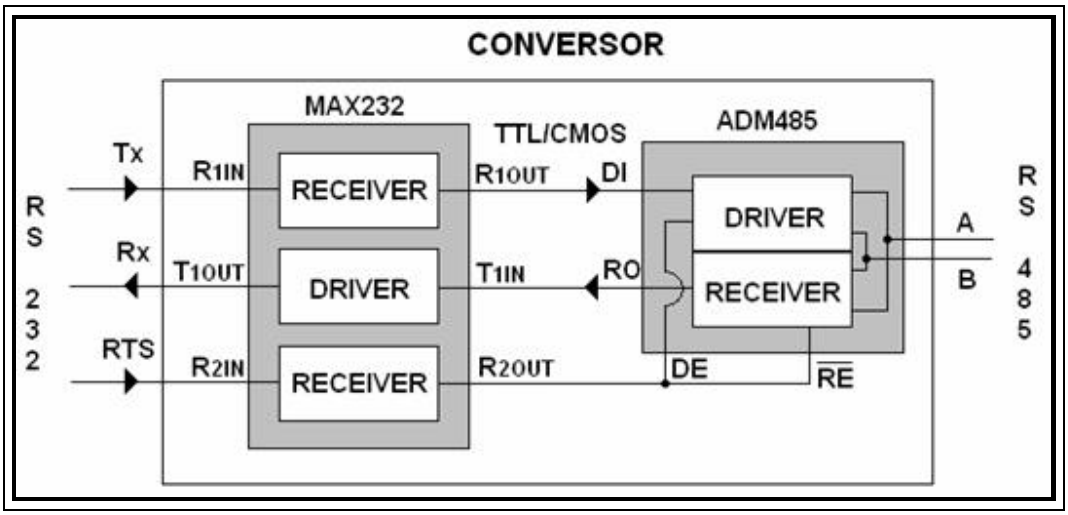
Estados	Niveles de Tensión		
	RS-485	TTL/CMOS	RS-232
0 Lógico	$V_B - V_A > 0.2V$	0V < L < 0.8V	3V < L < 12V
1 Lógico	$V_B - V_A < -0.2V$	3.5 V < H < 5 V	-12V < H < -3V

Estos límites pueden variar dependiendo del circuito integrado utilizado, más adelante cuando se describa cada circuito integrado se darán los rangos de voltaje utilizados por cada uno de ellos.

Para la implementación del conversor se utilizaron los siguientes circuitos integrados: MAX232 de *Texas Instruments*, ADM485 y ADP3367 de *Analog Devices*.

En la Figura 11 se muestra un esquema del conversor, sin tener en cuenta la alimentación de los circuitos.

Figura 11. Esquema del Conversor RS232/485

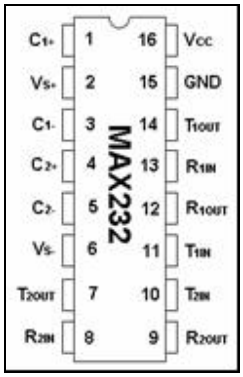


Autores

2.1 CIRCUITOS INTEGRADOS UTILIZADOS

2.1.1 MAX232

Figura 12. Diagrama de Pines del MAX232

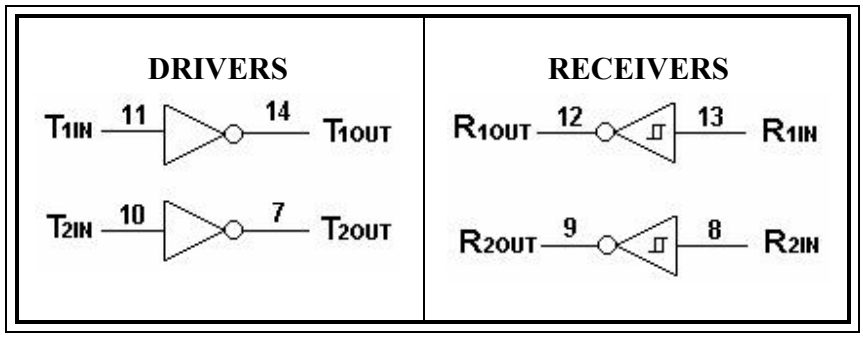


Este dispositivo por medio de sus *receivers*, convierte las entradas TIA/EIA-232-F a niveles TTL/CMOS de 5V y por medio de sus *drivers* convierte los niveles de entrada

TTL/CMOS de 5V en salidas TIA/EIA-232-F. La Figura 12 muestra la distribución de pines del MAX232.

Entre los pines 1 y 3, 4 y 5 se conectan capacitores de 1µF con la polaridad indicada en los subíndices, entre el pin 2 y Vcc se conecta un capacitor de 1µF y entre el pin 6 y tierra se conecta un capacitor de 1µF. El circuito integrado se alimenta por el pin 16 con una tensión de 5V y la tierra del circuito debe ser conectada al pin 15. La Figura 13 muestra el diagrama lógico de los *drivers* y *receivers*, los números indican el pin al que corresponde en el integrado las entradas y salidas de cada uno de ellos.

Figura 13. Diagrama lógico del MAX232



Las tablas 6 y 7 muestran la "tabla de verdad" de los *drivers* y los *receivers* respectivamente.

Tabla 6. "Tabla de verdad" driver MAX232

DRIVER	
ENTRADA T_{IN} (TTL/CMOS)	SALIDA T_{OUT} (RS-232)
0	0
1	1

Tabla 7. "Tabla de verdad" receiver MAX232

RECEIVER	
ENTRADA R_{IN} (RS-232)	SALIDA R_{OUT} (TTL/CMOS)
0	0
1	1

Se debe tener en cuenta que aunque los *drivers* y los *receivers* no cambian los niveles lógicos, sí hacen cambios en los niveles de tensión asociados a ellos. Las tablas 8 y 9,

muestran los rangos de voltajes permisibles en las entradas y los rangos que se entregan a las salidas, del *driver* y el *receiver* respectivamente.

Tabla 8. Rangos de voltaje en el driver del MAX232

DRIVER	
ENTRADA T_{IN} (TTL/CMOS)	SALIDA T_{OUT} (RS-232)
2V < H < 5.3V	-7V < H < -5V
0 < L < 0.8V	5V < L < 7V

Tabla 9. Rangos de voltaje en el receiver del MAX232

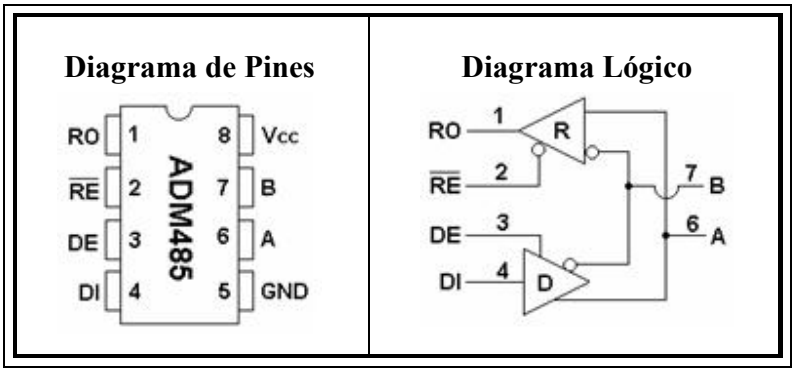
RECEIVER	
ENTRADA R_{IN} (RS-232)	SALIDA R_{OUT} (TTL/CMOS)
-30V < H < -3V	3.5V < H < 5.3V
3V< L < 30V	0V < L < 0.4V

Con el ADM485 se pasa de niveles de tensión TTL/CMOS a niveles de tensión RS-485 y viceversa.

2.1.2 ADM485

Con la utilización de este dispositivo se puede cambiar de un nivel de voltaje TTL/CMOS referido a tierra a un voltaje diferencial RS-485 por medio de su *driver* (D) y hacer el cambio de un voltaje diferencial RS-485 a un voltaje TTL/CMOS referenciado a tierra utilizando su *receiver* (R). El ADM485 es *half-duplex* para la conversión a RS-485, por eso se le debe suministrar señales de habilitación en los pines DE y $\overline{RE}$, para elegir la dirección de los datos en el bus. La señal DE habilita el *driver* permitiendo el envío de información al bus y la señal $\overline{RE}$ habilita el *receiver* para recibir información del bus.

Figura 14. Diagrama de Pines y circuito lógico del ADM485



La Figura 14 muestra la distribución de pines y el diagrama lógico del ADM485, los números que acompañan a los nombres de las señales en el diagrama lógico indican el pin correspondiente en el integrado.

El circuito integrado se alimenta por el pin 8 con una tensión de 5V y la tierra del circuito debe ser conectada al pin 5. La señal de habilitación del *driver* se conecta al pin DE (3) y la del *receiver* al pin $\overline{RE}$ (2). La conexión al bus se hace por medio de los pines A (6) y B (7), la salida del *receiver* es RO (1) y la entrada del *driver* es DI (4).

En la tabla 10 y 11 se muestra la "tabla de verdad" del *driver* y el *receiver* respectivamente.

Tabla 10. "Tabla de verdad" del driver del ADM485

DRIVER		
ENTRADAS (TTL/CMOS)		SALIDAS (RS-485)
DE	DI	$V_B - V_A$
1	1	1
1	0	0

El *driver* necesita tener la habilitación DE en alto (1) para obtener una salida en A y B; de lo contrario las salidas A y B quedan en alta impedancia.

Tabla 11. "Tabla de verdad" del receiver del ADM485

RECEIVER		
ENTRADA RE (TTL/CMOS)	**ENTRADAS V_B - V_A (RS-485)**	**SALIDA RO (TTL/CMOS)**
0	**0**	**1**
0	**1**	**0**
0	**ENTRADAS ABIERTAS**	**1**

El *receiver* solo obtiene una salida en RO si se tiene la habilitación $\overline{RE}$ en bajo (0), si $\overline{RE}$ está en alto (1) la salida RO queda en alta impedancia. Si por algún caso las salidas quedan abiertas, es decir, sin conectar; y la habilitación $\overline{RE}$ se encuentra en bajo (0); en la salida RO se obtiene un alto.

Las tablas 12 y 13, muestran los rangos de voltajes permisibles en las entradas y los rangos que se entregan a las salidas, del driver y el receiver respectivamente.

Tabla 12. Rangos de voltaje en el driver del ADM485

DRIVER	
ENTRADA	**SALIDA**
2V < H < 5.3V	**-9V < H < -4V**
0 < L < 0.8V	**3V < L < 14V**

Tabla 13.Rangos de voltaje en el receiver del ADM485

RECEIVER	
ENTRADA	**SALIDA**
-9V < H < -0.2V	**2V < H < 5.3V**
0.2V< L < 14V	**0V < L <0.8V**

Para el envío de información por las mismas dos líneas A y B, se debe habilitar el *driver* o el *receiver* de acuerdo a la dirección de la información. Si se desea enviar información al bus se debe habilitar el *driver* y deshabilitar el *receiver*, pero si el objetivo es recibir la información del bus, el *driver* se deshabilita y el *receiver* se habilita.

Como la señal que se debe enviar para habilitar al *driver* es la misma que se envía al *receiver* para deshabilitarlo y viceversa. A estas dos habilitaciones les debe llegar la misma señal para obtener una sola dirección en la comunicación.

La señal utilizada para elegir la dirección de la comunicación se envía desde el computador por la línea RTS del conector DB-9, como ésta señal está en niveles de tensión de RS-232 se pasa a través de un *receiver* del MAX232 para que se pueda aplicar a las habilitaciones DE y $\overline{RE}$ del ADM485.

A diferencia de las señales que envía el computador con el estándar RS-232 por el puerto DB-9, cuando se usa la línea RTS de forma independiente se tiene que un 0 es un voltaje negativo y un 1 es un voltaje positivo. Cuando se envía información desde el computador, primero se coloca la línea RTS en 0 y cuando se hayan enviado los datos se pasa a 1. Esto produce un pulso negativo que al pasar por un receiver del MAX232 se convierte en un pulso positivo, habilitando el *driver* (y deshabilitando el *receiver*) del ADM485; permitiendo el envío de los datos hacia el bus. Mientras el computador no envía datos, la señal RTS se mantiene en 1, que es un 0 después de pasar por un receiver del MAX232, conllevando a la habilitación del *receiver* para la recepción de información del bus.

Para verificar el correcto funcionamiento del conversor se hizo un primer montaje con circuitos integrados de empaquetado DIP, para su alimentación se utilizó el regulador LM7805, el esquema de este conversor se puede apreciar en la Figura 15.

Figura 15. Conversor RS-232/RS-485 con integrados DIP

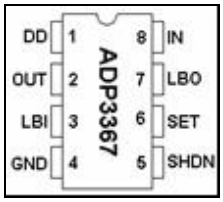
Debido al buen funcionamiento del conversor y queriendo minimizar su tamaño, se montó un nuevo conversor con circuitos integrados de montaje superficial, por esto se cambió el LM7805 por el circuito integrado ADP3367 de *Analog Devices*.

2.1.3 ADP3367

El ADP3367 es un regulador de voltaje de precisión, la tensión de salida puede variarse por medio de resistencias, desde 1.3V hasta 16V con una alimentación máxima de 16.5V. Se escogió este circuito integrado no solo por suministrar la corriente necesaria para el MAX232 y el ADM485, sino por ser de montaje superficial, lo que disminuye el tamaño del conversor.

La Figura 16 muestra la distribución de pines del ADP3367.

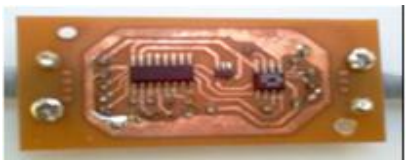
Figura 16. Diagrama de Pines del ADP3367



Para la utilización del ADP3367 como una fuente fija de 5V no se necesitan resistencias, solo un capacitor de 1 µF a la entrada (IN) donde se puede alimentar desde un voltaje de 5.15V hasta 16.5V, un capacitor de 10 µF a la salida (OUT) donde se toman los 5V; además se conectan a tierra los pines GND, SHDN y SET.

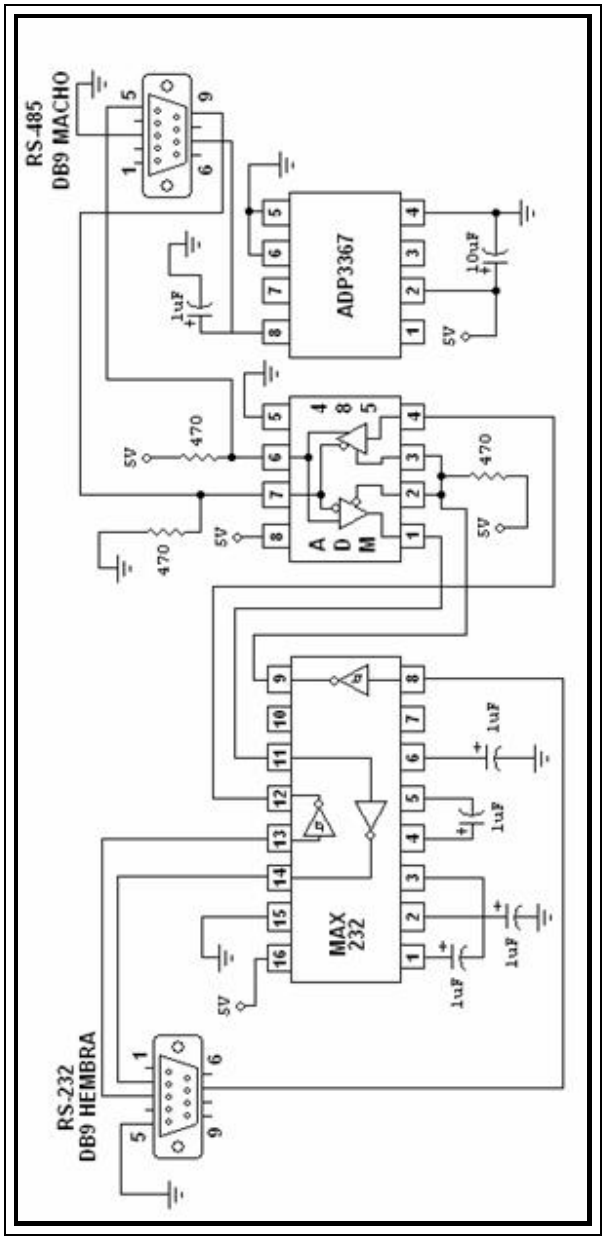
El ADP3367 puede suministrar 300mA a su salida, por lo cual puede abastecer la corriente de alimentación máxima del MAX232 de 10mA y la corriente de alimentación máxima del ADM485 de 2.2mA. El conversor con los circuitos integrados de montaje superficial se ilustra en la Figura 17, dimensiones 3 X 6 cm.

Figura 17. Conversor RS-232/RS-485 con Integrados Superficiales



La Figura 18 muestra la conexión del circuito final del conversor.

Figura 18. Circuito del Conversor RS-232/RS-485



Autores

2.2 OSCILOGRAMAS OBTENIDOS EN EL CONVERSOR

A continuación (Figuras 19 a 26) se mostrarán los oscilogramas obtenidos en el FLUKE 105 SCOPEMETER SERIES II al enviar una trama desde el computador hasta un PLC Trilogi empleando el conversor. La velocidad de transmisión fue 9600 bits/s, por consiguiente el tiempo de transmisión de un bit es aproximadamente 0.10416 ms.

La trama es la siguiente: 02 02 01 00 00 10 78 09 (Hexadecimal), ella indica según el protocolo MODBUS (en los siguientes capitulos se habla en detalle del protocolo MODBUS) que:

02 (Hex) → Se accesa al esclavo 2
02 (Hex) → Función 2 (Se pide el estado de las entradas)
01 00 (hex) → Se leen a partir de la dirección de memoria 256 (decimal)
00 10 (Hex) → Se leen 16 entradas
78 08 (Hex) → Valor del control de redundancia cíclica (CRC)

Teniendo en cuenta que los datos se envían con 1 bit de comienzo (es cero), 8 bits de datos, 1 bit de parada (es uno) y que el bit menos significativo es el primero de los 8 bits de datos; la información empaquetada con el protocolo MODBUS en modo RTU que sale del puerto del computador deben ser la siguiente:

02 (Hex) → 2 (Dec) → **0**01000000**1** (Binario)
02 (Hex) → 2 (Dec) → **0**01000000**1** (Binario)
01 (Hex) → 1 (Dec) → **0**10000000**1** (Binario)
00 (Hex) → 0 (Dec) → **0**00000000**1** (Binario)
00 (Hex) → 0 (Dec) → **0**00000000**1** (Binario)
10 (Hex) → 16 (Dec) → **0**00001000**1** (Binario)
78 (Hex) → 120 (Dec) → **0**00011110**1** (Binario)
09 (Hex) → 9 (Dec) → **0**10010000**1** (Binario)

Y haciendo la misma petición con el protocolo MODBUS en modo ASCII:

3A (Hex) → : (ASCII) → **0**0101110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
32 (Hex) → 2 (ASCII) → **0**0100110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
32 (Hex) → 2 (ASCII) → **0**0100110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
31 (Hex) → 1 (ASCII) → **0**1000110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
31 (Hex) → 1 (ASCII) → **0**1000110**01** (Binario)
30 (Hex) → 0 (ASCII) → **0**0000110**01** (Binario)
45 (Hex) → E (ASCII) → **0**1010001**01** (Binario)
42 (Hex) → B (ASCII) → **0**0100001**01** (Binario)
0D (Hex) → CR (ASCII) → **0**1011000**01** (Binario)
0A (Hex) → LF (ASCII) → **0**0101000**01** (Binario)

Los 0 en negrilla indican los bits de comienzo y los 1 en negrilla indican los bits de parada. Como se envían 10 bits por cada carácter, el tiempo de transmisión de un carácter es 1.0416 ms. La trama enviada en modo RTU consta de 8 caracteres; por lo tanto el tiempo de transmisión es aproximadamente de 8.3328 ms.
La trama enviada en modo ASCII consta de 17 caracteres ASCII, más del doble que en RTU; con un tiempo de transmisión aproximado de 17.7072 ms.

Señal que sale del computador y llega a la entrada (pin 13 R_{1IN}) del *Receiver* del MAX232:

En la Figura 19 el eje horizontal es el tiempo y el eje vertical es el voltaje. Cada cuadrícula representa en el tiempo 1 ms y en voltaje 5V. Se puede apreciar que el tiempo de

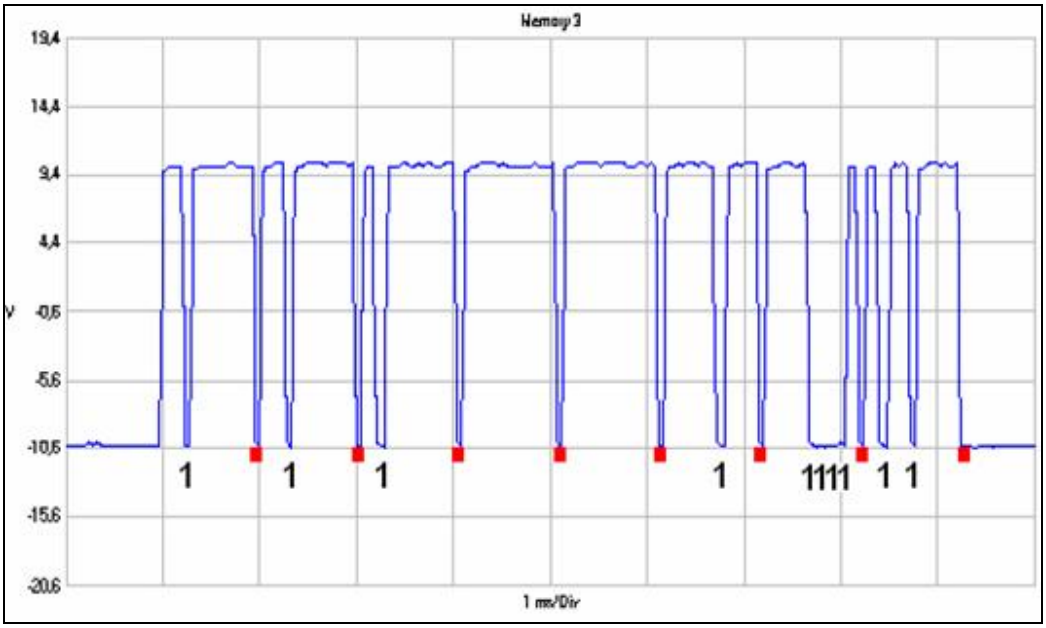
transmisión de la trama RTU es menor que 8.5 ms, coincidiendo con los cálculos teóricos de 8.3328 ms, y que los niveles lógicos están dentro de los rangos especificados para el estándar RS-232:

0 lógico = 3V < **9.4 V** < 12V

1 lógico = -12V < **-10.6 V** < -3V

En la Figura 19 se insertaron cuadrados para identificar los bits de parada y se señalizaron los unos lógicos dentro de la trama RTU.

Figura 19. Trama RTU del computador al MAX232



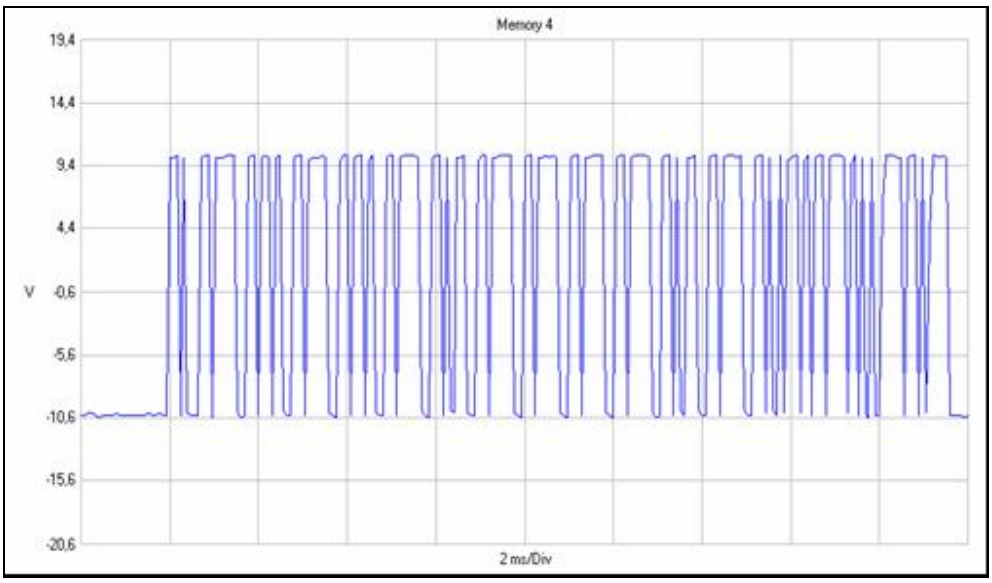
La Figura 20 tiene las mismas variables en los ejes que la Figura 19, manteniendo el valor de 5V de la cuadrícula pero representando en el tiempo 2 ms.

Se ve en la Figura 20 que la trama ASCII se transmite en un tiempo menor a 18 ms, concordando con los 17.7072 ms esperados para el envío de la trama ASCII.

Los niveles lógicos son iguales a los presentados en la Figura 19.

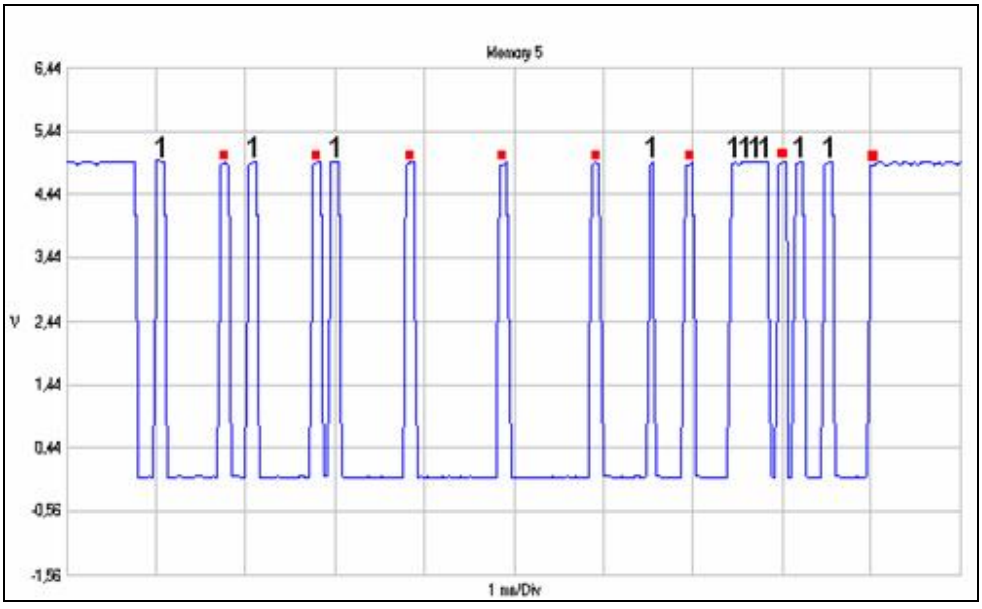
Debido al hecho que para tomar la trama ASCII completa se pierde resolución en el oscilograma no se puede discernir completamente los bits de parada y los unos lógicos.

Figura 20. Trama ASCII del computador al MAX232



Señal proveniente de la salida del *receiver* (pin 12 R_{1OUT}) del MAX232 y que llega a la entrada del *driver* (pin 4 DI) del ADM485.

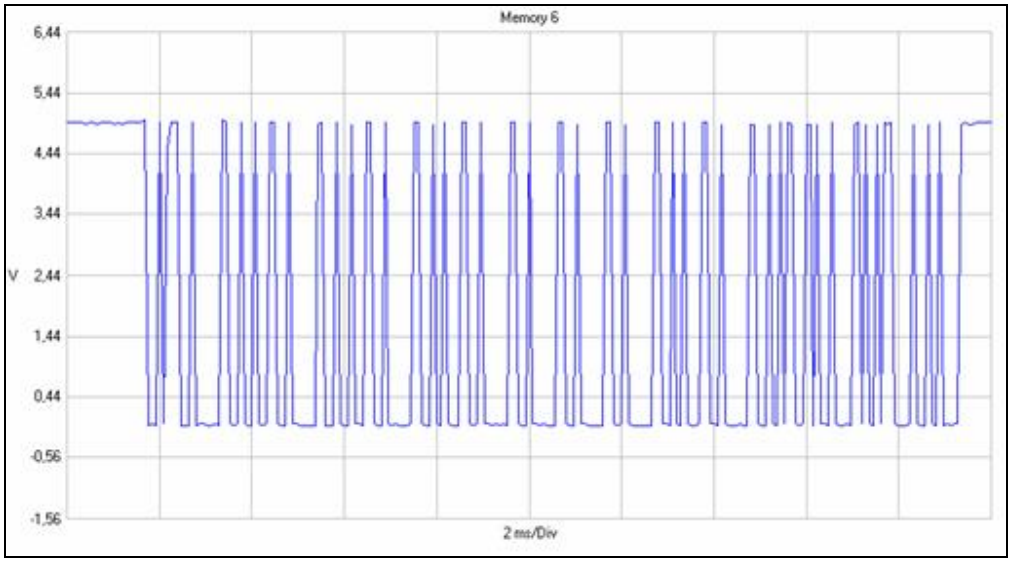
Figura 21. Trama RTU del MAX232 al ADM485



Los ejes en la Figura 21 siguen siendo, de tiempo en el horizontal y de voltaje en el vertical. La escala de cada cuadrícula es de 1 ms y 1V. Se sigue apreciando un tiempo de

transmisión un poco mayor a 8 ms. Los niveles lógicos están dentro de los rangos descritos para TTL/CMOS, ya que el 1 lógico se encuentra cercano a los 5V y el 0 lógico a los 0 V.
Al igual que en la Figura 19 se identificaron los bits de parada con un cuadrado y los unos lógicos fueron señalizados.
Al comparar la Figura 21 con la Figura 19 pareciera que se invierten los niveles lógicos, pero lo que realmente pasa es que se cambian los niveles de voltaje asociados a cada uno de ellos.

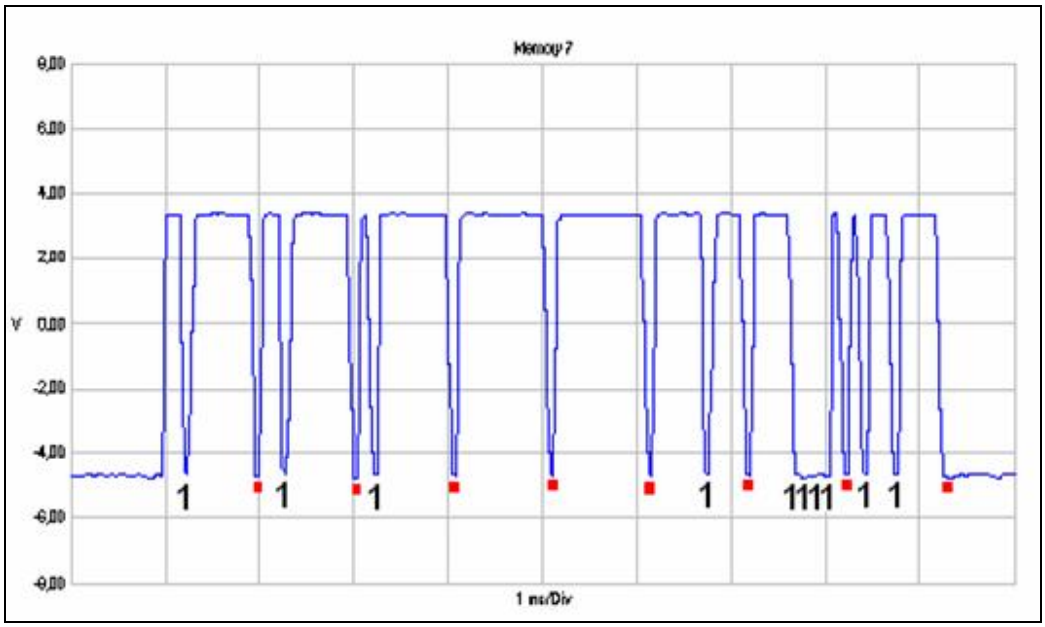
Figura 22. Trama ASCII del MAX232 al ADM485



En la Figura 22 se mantienen los ejes como en la Figura 21, manteniendo el valor de 5V para cada cuadrícula pero representando 2 ms en el tiempo. Se mantiene el tiempo de transmisión de la trama ASCII en aproximadamente 18 ms.
Los niveles de voltaje se conservan iguales a los presentados en la Figura 21. Al ver la Figura 20 y 22 también se aprecia el cambio de los niveles de voltaje asociados a cada nivel lógico.

Señal proveniente de la salida del driver (pin 7 B y pin 6 A) del ADM485 y que se envía al bus.

Figura 23. Trama RTU del ADM485 al Bus



El voltaje y el tiempo en la Figura 23 son el eje vertical y el horizontal respectivamente. Cada cuadrícula equivale en voltaje a 2 V y en el tiempo a un 1 ms. El tiempo de transmisión se mantiene en un poco más de 8 ms.

El voltaje de un 0 lógico ($V_B - V_A > 2V$) y de un 1 lógico ($V_B - V_A < -2V$) para el estándar RS-485 se cumple, puesto que, de la gráfica se puede inferir que un 0 lógico tiene un voltaje mayor a 3V y un 1 lógico un voltaje menor a -4V.

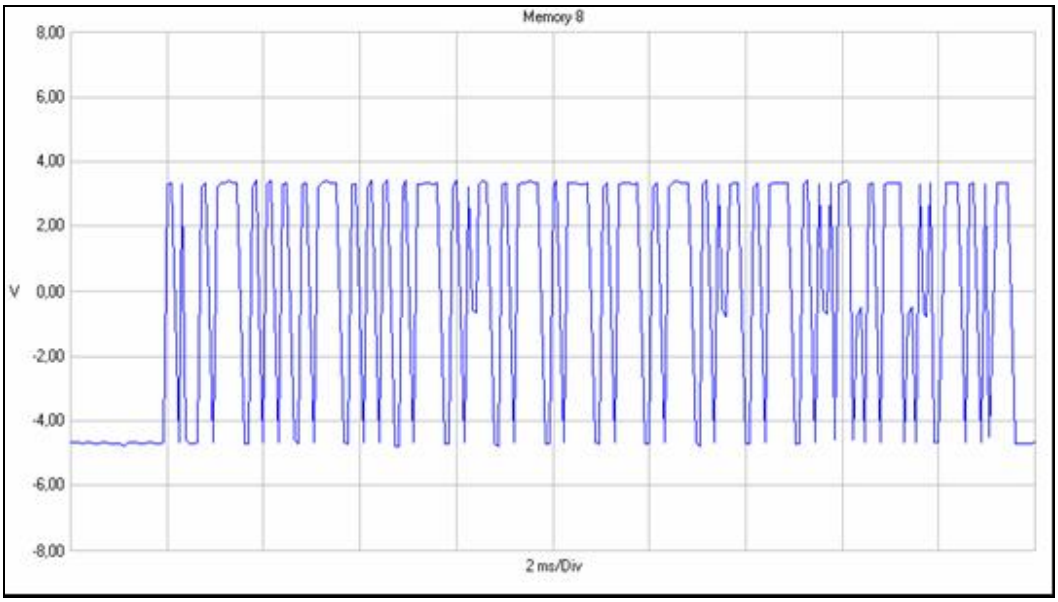
Al igual que en la Figura 19 y 21, en la Figura 23 se indicaron por medio de cuadrados los bits de parada y los unos lógicos fueron señalizados.

Los ejes en la Figura 24 corresponden a las mismas variables de la Figura 23. En tiempo cada cuadrícula equivale a 2 ms y en voltaje a 2V. Se sigue manteniendo el tiempo de transmisión en aproximadamente 18 ms.

Los niveles de voltaje son los mismos que en la Figura 23, por ende están en el rango especificado para RS-485.

Si se observan las Figuras 19 y 23 se aprecia que las dos formas de onda son semejantes, pero con diferentes niveles de voltaje, indicando que se está transmitiendo la misma información del computador al bus; claro está que habiéndose efectuado la conversión de RS-232 a RS-485. Lo mismo ocurre al comparar las Figuras 20 y 24.

Figura 24. Trama ASCII del ADM485 al bus



Consulta a un PLC Trilogi usando el Protocolo MODBUS

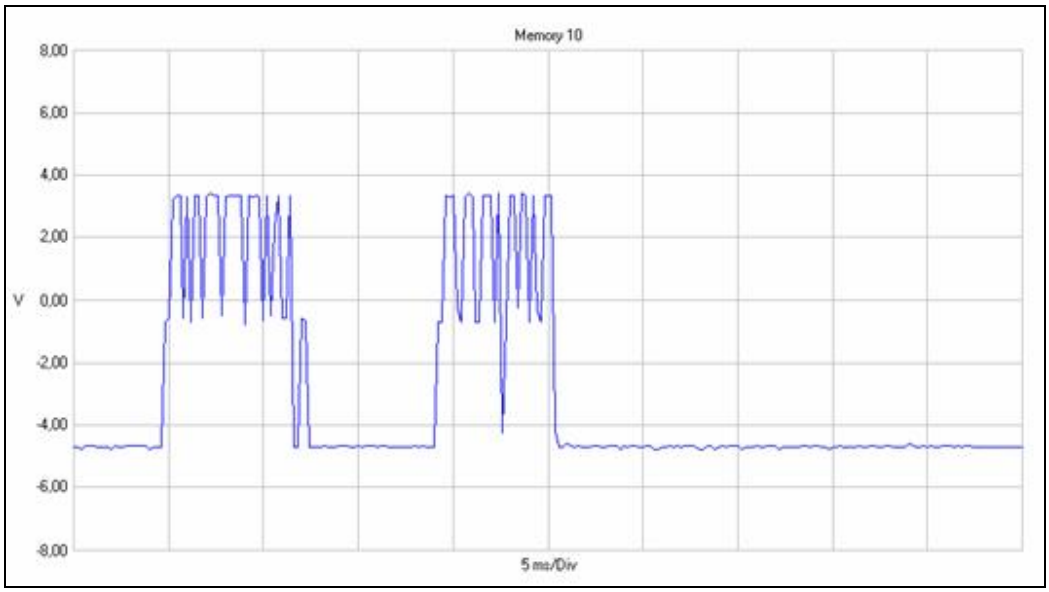
En las Figuras 25 y 26 se muestra el envío de una trama completa a un PLC Trilogi, con la respectiva respuesta de él; en modo RTU y ASCII respectivamente. El eje vertical sigue siendo el voltaje al igual que el eje horizontal es el tiempo. El voltaje y el tiempo representado por cada cuadrícula es de 2 V y 5 ms, respectivamente.

Modo RTU
Trama enviada: 02 02 00 00 00 08 79 FF (Hexadecimal)
Respuesta recibida: 02 02 01 17 E1 C2 (Hexadecimal)

En la Figura 26 la trama de la izquierda es la consulta y la trama de la derecha es la respuesta. Como la trama de consulta tiene 8 caracteres, el tiempo de transmisión es de 8.33 ms; coincidiendo aproximadamente con lo que se puede ver en la Figura 25.

Al tener la trama respuesta 6 caracteres su tiempo de transmisión es de 6.2496 ms concordando también con la Figura 25.

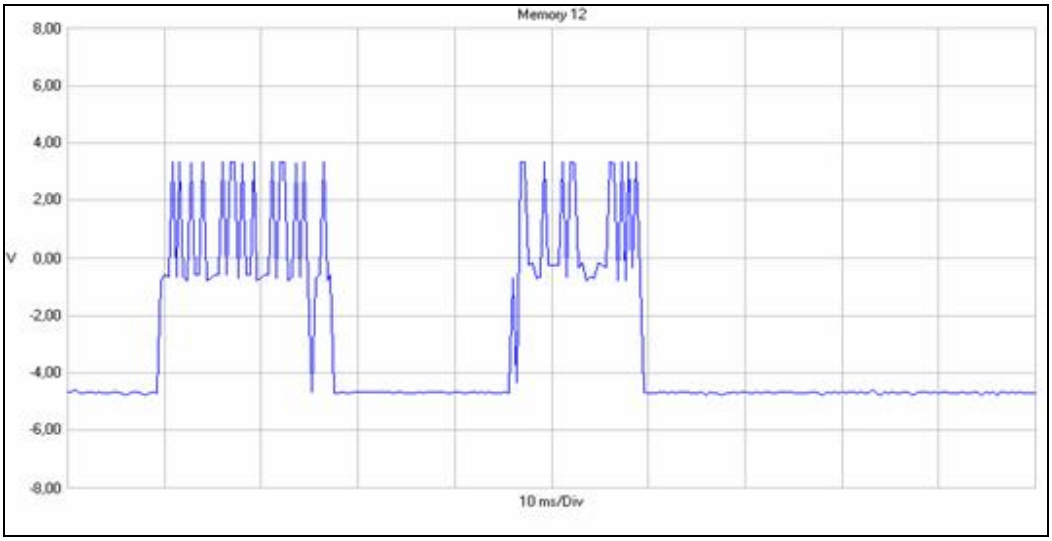
Figura 25. Trama consulta – repuesta, Modo RTU



Modo ASCII

Trama enviada: 3A 30 32 32 30 30 30 30 30 30 30 34 46 38 0D 0A (Hexadecimal)
Respuesta recibida: 3A 30 32 30 32 30 31 30 36 46 35 0D 0A (Hexadecimal)
La trama de la izquierda y la derecha en la Figura 26, son las tramas consulta y respuesta, respectivamente. Puesto que la trama de consulta tiene 16 caracteres, el tiempo que demora en transmitirse es de 16.6656 ms; y como la trama de respuesta consta de 13 caracteres su tiempo de transmisión es de 11.5408 ms

Figura 26. Trama consulta – repuesta, Modo ASCII



3. PROTOCOLO MODBUS

El protocolo Modbus es el lenguaje común que poseen algunos dispositivos, para comunicarse entre ellos y con otros dispositivos. Nace como un protocolo propietario de GOULD.INC, y luego es convertido en un protocolo abierto como resultado de un intento de estandarización a nivel de buses de campo.

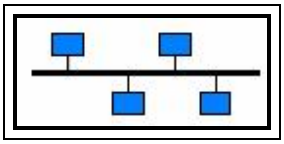
En una red Modbus, el protocolo define la forma de los mensajes consulta del maestro; además de cómo cada esclavo reconocerá un mensaje dirigido a él, determinando la acción a tomar y si se requiere construirá un mensaje respuesta con la información pedida por el maestro.

3.1 CARACTERÍSTICAS DE UNA RED ESTÁNDAR MODBUS

- **Topología**: Bus

Un bus es una red lineal sobre la que se conectan todos los dispositivos, cada dispositivo toma la información de la red y verifica si es para este, si no la desecha, ver Figura 27.

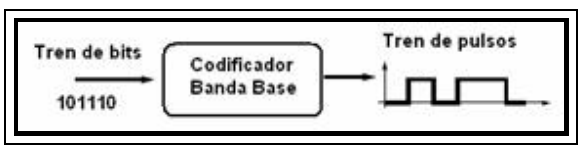
Figura 27. Topología Bus



- **Modo de emisión**: Banda base

En banda base, el tren de bits que representan los datos es codificado por medio de un codificador de banda base (ejemplo: «0» corresponde al nivel 13 V, «1» corresponde al nivel -13 V), ver Figura 28.

Figura 28. Modo de emisión banda base

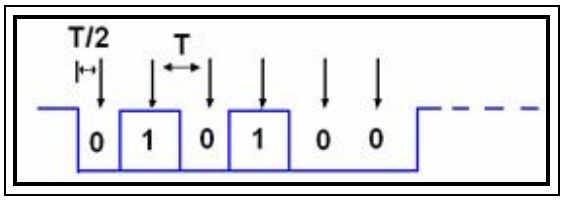


- A nivel físico las redes estándares Modbus emplean alguno de los estándares de comunicación serial tales como:
    - **EIA/TIA-232-E**: punto a punto y *fullduplex*.
    - **EIA-422**: punto a punto y *fullduplex*.
    - **EIA/TIA-485-A**: multipunto y *halfduplex*.

- La comunicación es **asíncrona**.

En la comunicación asíncrona el transmisor y el receptor deben tener la misma velocidad de comunicación (un tiempo de bit T) y el mismo número de bits en cada carácter trasmitido. Ante una señal de inicio, el receptor muestrea la línea en intervalos de periodo T, ver Figura 29.

Figura 29. Muestreo Asíncrono



- **Velocidades de transmisión** que varían desde 75 bps a 19200 bps.
- **Acceso al medio** es empleando la técnica Maestro-Esclavo:

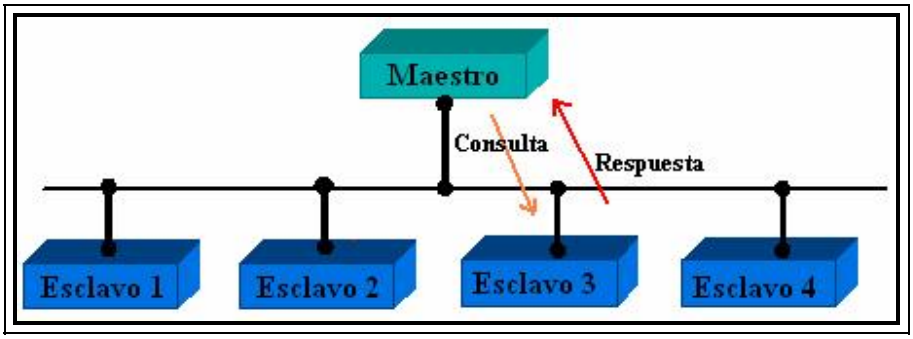
En la cual se dispone de un dispositivo maestro y uno o varios dispositivos esclavos; el dispositivo que actúe como maestro será el único que puede iniciar las transacciones, los dispositivos esclavos responderán suministrando información o realizando la acción requerida por la petición del maestro.

El maestro tiene el control del acceso al medio, existiendo dos posibilidades: es el maestro el que ocupa el medio físico, o es el esclavo que le responde al maestro.

Tipos de transacciones:

- **Consulta/Respuesta** *(Query-response)*: Es un intercambio punto a punto entre el maestro y algún esclavo. Cada esclavo tiene una dirección única, ver Figura 30.

Figura 30. Transacción consulta - respuesta



- **Difusión sin respuesta** *(broadcast /no response)*: El maestro envía un mensaje a todos los esclavos de la red, pero ninguno contesta, ver Figura 31.

Figura 31. Transacción de difusión

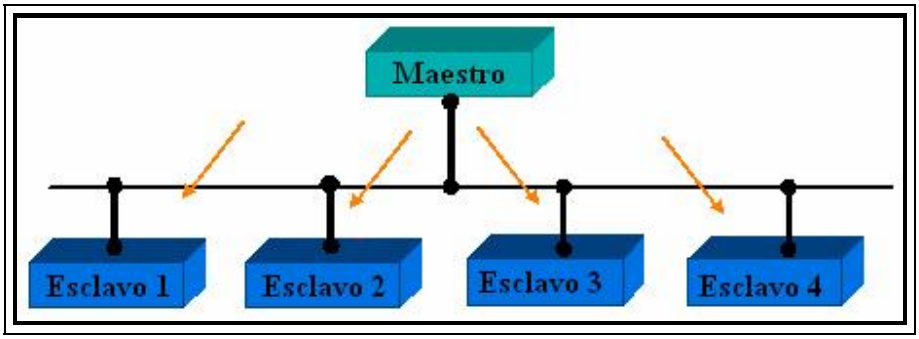
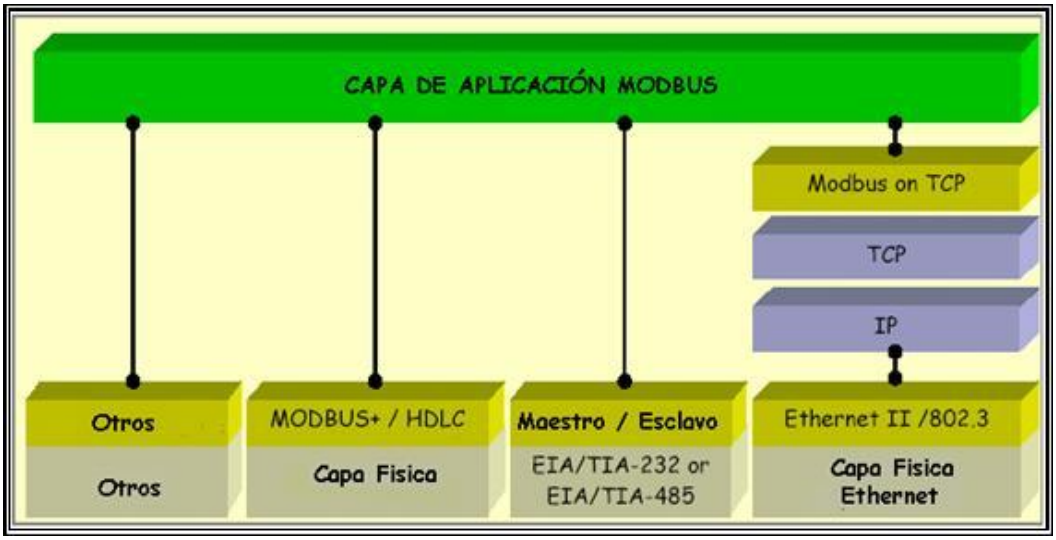


Figura 32. Capas y diferentes sistemas de comunicación Modbus.



Fuente: www.Modbus.org

▪ **Capa de aplicación Modbus**

Modbus es un protocolo de mensajería en la capa de usuario correspondiente al nivel 7 del modelo de OSI (*Open System Interconnection*); en la Figura 32 se distinguen los diferentes sistemas de comunicación Modbus tales como: Modbus sobre TCP/IP (*Ethernet*) y Modbus estándar (*EIA/TIA 232 o 485*).

En el presente trabajo se emplea la red estándar Modbus que se caracteriza por tener en su capa física el estándar EIA/TIA-485-A, en su capa de enlace la técnica maestro-esclavo y en la capa de usuario el protocolo Modbus.

3.2 MODOS DE TRANSMISIÓN SERIE

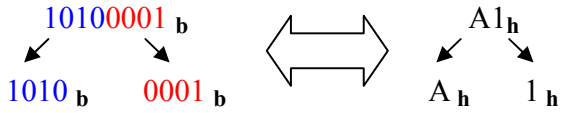
Existen dos modos de transmisión en las redes estándares Modbus estos son: ASCII y RTU.

3.2.1 Modo ASCII

El Código Estándar Americano para Intercambio de Información "ASCII" es un código alfanumérico universal, usado por la mayoría de computadoras y equipos electrónicos.

En este modo cada byte de información es codificado en ASCII. El proceso de codificación es:

1. Cada byte de información del mensaje se descompone en dos *nibbles*; ejemplo:



2. Cada *nibble* es convertido en un carácter en código ASCII sumándole:

   - 30_h si el *nibble* esta entre 0_h y 9_h
   - 37_h si el *nibble* esta entre A_h y F_h

Ejemplo:

$$1010 = A_h \xrightarrow{+37\,h} 41_h \text{ Código ASCII}$$

$$0001 = 1_h \xrightarrow{+30\,h} 31_h \text{ Código ASCII}$$

Obteniendo que: $10100001 = 41_h\ 31_h$ Código ASCII

Los valores 30 $_h$ y 37 $_h$ son obtenidos revisando las tablas de los caracteres del código ASCII ver Anexo 1.

Luego cada carácter en código ASCII se empaqueta en un carácter "serie ASCII", para ser enviado por el puerto serie; el carácter "serie ASCII" contiene un bit de inicio, 7 ó 8 bits de datos (corresponden a un carácter en código ASCII), un bit de paridad y uno o dos bit de final si no se calcula la paridad, ver Figura 33.

Figura 33. Empaquetado de un carácter "serie ASCII"

1bit	7 ó 8 Bits de datos (Caracter ASCII)							1 bit	1 bit
INICIO	MSB						LSB	PARIDAD	FIN

Características del Mensaje Modbus en Modo ASCII

- El mensaje Modbus comienzan con el símbolo ASCII dos puntos (":" equivale a 3A $_h$ código ASCII) y termina con el par de caracteres de control retorno de carro – salto de línea (CR LF equivalen a D $_h$ A $_h$ código ASCII).
- Los caracteres permitidos en la transmisión para todos los demás campos son 0$_h$ - 9$_h$ (30$_h$ -39$_h$ código ASCII) y A $_h$ - F $_h$ (41 $_h$ -46 $_h$ código ASCII).
- Los esclavos conectados vigilan el bus de red continuamente para detectar el símbolo ASCII (:). Cuando se recibe, cada dispositivo decodifica el siguiente campo (el Campo Dirección) para determinar si es el dispositivo direccionado.

El Formato del mensaje en modo ASCII es:

INICIO	DIRECCIÓN	FUNCIÓN	DATOS	CHEQUEO	FIN
1 CAR : 3A hex	2 CAR	2 CAR	*N CAR*	2 CAR	2 CAR CR LF D Y A hex

Donde CAR representa un carácter ASCII.

3.2.2 Modo RTU

En modo RTU (Unidad Terminal Remota), cada byte de información se empaquetan en un carácter "serie RTU"; este carácter "serie RTU" contiene un bit de inicio, 8 bits de datos (byte información), un bit de paridad y uno o dos bit de final si no se calcula la paridad, ver Figura 34.

Figura 34. Empaquetado de un carácter "serie RTU"

Características del Mensaje Modbus en Modo RTU

- El mensaje Modbus empieza con un intervalo de silencio de al menos 3,5 veces el tiempo de transmisión de un carácter. Esto se realiza esperando un tiempo múltiplo del tiempo de transmisión de un carácter que es inversamente proporcional a la velocidad en bps que se está utilizando en la red (es muy utilizado el tiempo de silencio de 4 veces el tiempo de transmisión de un carácter), luego se envía el primer campo que corresponde a la dirección del dispositivo.
- Después del último carácter transmitido se intercala un intervalo de tiempo equivalente minino de 3.5 veces el tiempo de transmisión de un carácter para marcar el fin del mensaje. Después de este intervalo puede comenzar un nuevo mensaje.
- Los caracteres permitidos para todos los campo son 0_h-9_h, A_h–F_h.
- Los dispositivos conectados vigilan el bus de red continuamente, incluso en los intervalos de silencio. Cuando se recibe el primer campo (el Campo de Dirección), cada unidad lo decodifica para determinar si es el dispositivo direccionado.

El formato de mensaje en modo RTU es:

INICIO	DIRECCIÓN	FUNCIÓN	DATOS	CHEQUEO	FIN
T1-T2-T3-T4	1 Byte	1 Byte	N Bytes	2 Bytes	T1-T2-T3-T4

Donde CAR representa un carácter RTU.

3.3 FORMATO DEL MENSAJE MODBUS

3.3.1 Campo de Dirección

El maestro coloca la dirección del esclavo al que quiere consultar en el campo Dirección del mensaje, para iniciar la comunicación con este. Cuando el esclavo envía su respuesta, coloca su propia dirección en el campo Dirección de la respuesta para confirmar al maestro que es su respuesta.

- Modo ASCII: contiene 2 caracteres en código ASCII.
- Modo RTU: contiene 1 byte.
- Direcciones validas para los esclavos: 1-247 decimal.
- La dirección 0 se utiliza para modo difusión, esta es reconocida por todos los dispositivos esclavos.

3.3.2 Campo de Función

Cuando se envía un mensaje desde el maestro a un dispositivo esclavo, el campo de Función contiene el Código de Operación de alguna función Modbus, el cual define la acción que debe realizar el esclavo.

- Modo ASCII: contiene 2 caracteres en código ASCII.
- Modo RTU: contiene 1 byte.
- Respuesta normal: el esclavo copia el mismo código de la operación original.
- Respuesta de excepción: el esclavo cambia el código de función para indicar que ha ocurrido una anomalía.
- Hay tres categorías de código de operación MODBUS, ver Figura 35:

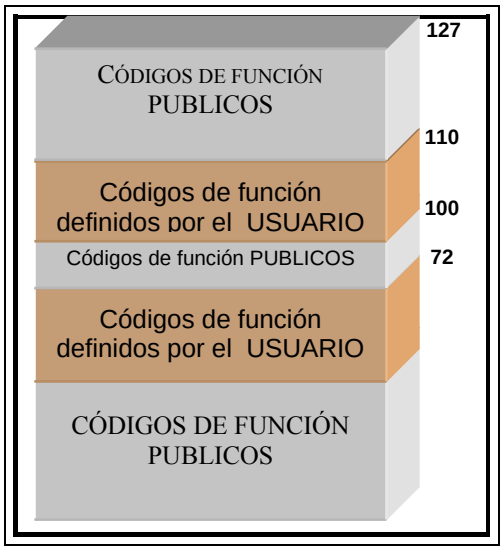
  ***Códigos de Función públicos*:** Son códigos de función bien definidos, validados y públicamente documentados por la comunidad modbus.org. Los códigos de

función público tienen tres rangos estos son de 1 a 65, 72 a 100 y 110 a 127 (decimal).

Códigos de Función definidos por el usuario: Son códigos de función que el usuario define para su aplicación específica. Existen dos rangos para estos códigos: 65 a 72 y de 100 a 110 (decimal).El usuario puede implementar estos códigos sin necesidad de su aprobación por la comunidad modbus.org

Códigos de Función reservados: Son códigos de función utilizados por algunas compañías para sus productos y que no son de uso público (rangos: 128-256).

Figura 35. Categorías de códigos de operación Modbus



Fuente:www.Modbus.org

- **Códigos Funciones de control y de datos**

En la Tabla 14 se presentan el código de las principales funciones, el nombre de la función y una breve descripción de la tarea a realizar por el dispositivo esclavo.

Tabla 14.Funciones de control y de datos

Código		Nombre	Tarea
D	Hex.		
01	**01**	**Leer estado de las Bobinas** (las bobinas son bits correspondientes a las salidas)	Lee el estado de las salidas discretas**.**
02	**02**	**Leer estado de las Entradas** (bits de las entradas)	Lee el estado de las entradas discretas.
03	**03**	**Leer Registros Internos**	Lee los contenidos binarios de los registros internos.
04	**04**	**Leer Registros de Entrada**	Lee los contenidos binarios de los registros de entrada.
05	**05**	**Forzar una "Bobina"**	Forzar una "bobina".
06	**06**	**Fijar un Valor en un Registro**	Fija un valor en un registro.
07	**07**	**Leer estado de la Excepción**	Lee el contenido de ocho bits de condición de excepción. Los bits están definidos previamente en cada dispositivo.
08	**08**	**Diagnóstico**	Permite una serie de pruebas para comprobar el sistema de comunicación entre el maestro y esclavo, o para comprobar diversas condiciones internas de error dentro del esclavo.
11	**0B**	**Solicitud de contador de eventos**	Devuelve una palabra de estado y un contador de eventos de comunicaciones del esclavo.
12	**0C**	**Solicitud de diario de eventos de comunicaciones**	Devuelve una palabra de estado, un contador de eventos, un contador de mensajes, y un campo de bytes de evento del esclavo.
15	**0F**	**Forzar varias Bobinas**	Forzar varias bobinas consecutivas.
16	**10**	**Fijar Varios registros**	Fija valores en varios registros internos consecutivos.

Las funciones sombreadas tienen permitido el modo difusión (dirección del esclavo = 00), esto significa que la función seleccionada forza la misma referencia en todos los esclavos conectados.

❖ **Subfunciones de Diagnóstico**

La función Modbus 08 "Diagnóstico" provee algunas pruebas para comprobar diferentes condiciones internas de error de un esclavo. La función de diagnóstico no afecta el programa de usuario del esclavo.

El formato del mensaje consulta para la función diagnóstico es el siguiente:

- Dirección del esclavo.
- Función: Código de operación = 08
- Campo de información:
  - Código de Subfunción (modo RTU longitud 2 bytes, modo ASCII 4 caracteres ASCII)
  - Datos (modo RTU longitud 2 bytes, modo ASCII 4 caracteres ASCII)
- CRC ó LRC.

El formato del mensaje respuesta es el mismo que el de consulta. En la Tabla 15 se presentan los códigos de las subfunciones de diagnóstico con una descripción de cada uno.

Tabla 15.Subfunciones de diagnóstico

Cod (hex)	Nombre de la subfunción	Característica
00	**Devuelve la misma información**	El campo de información de la respuesta devuelve exactamente el mismo de dato que el maestro le envió.
0B	**Devuelve contador de mensaje de bus**	El campo de información de la respuesta devuelve la cantidad de mensajes que el esclavo ha detectado en la red.
0C	**Devuelve contador de error de bus de comunicación.**	El campo de información de la respuesta devuelve la cantidad de errores (CRC ó LRC) detectados por el esclavo.
0D	**Devuelve contador de error de excepción en el bus.**	El campo de información de la respuesta devuelve la cantidad de respuestas de excepción devueltas por el esclavo.
0E	**Devuelve contador de mensajes del esclavo.**	El campo de información de la respuesta devuelve la cantidad de mensajes dirigidos al esclavo en modo difusión.
0F	**Devuelve contador no respuesta del esclavo.**	El campo de información de la respuesta devuelve la cantidad de mensajes dirigidos al esclavo que no contestó.
10	**Devuelve contador NAK del esclavo.**	El campo de información de la respuesta devuelve la cantidad de mensajes dirigidos al esclavo a los que devolvió una respuesta de excepción, Negativa de Reconocimiento NAK.

❖ **Respuesta de Excepción**

El esclavo recibe la consulta sin errores de comunicación pero no la puede efectuar por alguna causa, por lo tanto envía una respuesta de excepción informando al maestro el problema.

La respuesta de excepción contiene el siguiente formato:

- Dirección del esclavo.
- Función: Código de operación con el bit más significativo puesta a uno.
- Campo de información: el código de excepción que indica la causa del error.
- CRC ó LRC.

En la Tabla 16. Se presenta el listado de los códigos de excepción, el nombre de la excepción y su significado.

Tabla 16.Códigos de excepción

Código	Nombre	Significado
01	**Función Ilegal**	El código de operación en el campo de Función no lo tiene implementado el esclavo.
02	**Dirección de datos ilegal**	La dirección de datos recibida en la consulta no es permitida por el esclavo.
03	**Valor de datos ilegal**	El valor en el campo de datos recibida en la consulta no es permitida por el esclavo.
04	**Falla del dispositivo Esclavo**	Un error ocurre mientras el esclavo esta atendiendo la ejecución de la acción pedida.
05	**Desempeño**	El esclavo acepta la consulta y la procesa, pero un largo período de tiempo es requerido. Se retorna este código de excepción para prevenir que ocurra un error de tiempo en el maestro. El maestro puede emitir una consulta del mensaje completo para determinar si el procesamiento ha sido completado.
06	**Dispositivo esclavo Ocupado**	El esclavo esta ocupado en el procesamiento de un comando del programa de larga duración. El maestro deberá retransmitir el mensaje cuando el esclavo este libre.
07	**Negativa de reconocimiento (NAK)**	El esclavo no puede ejecutar la función de programa recibida en la consulta. El maestro deberá consultar información de diagnóstico o error a el esclavo.
08	**Error de paridad de memoria**	El esclavo intenta leer la memoria extendida, pero detecta un error de paridad de memoria.

Ejemplo: Se consulta al PLC KOYO con dirección de esclavo igual 02 el estado ON/OFF (código de operación 02) de sus 16 entradas DC, la trama enviada por el maestro en modo RTU es:

Campo de información

	Dirección del esclavo	Código de operación	Dirección de inicio	Nº de entradas	CRC	
T1-T2-T3-T4	**02**	**02**	**00 00**	**00 10**	**79 F5**	**T1-T2-T3-T4**

La repuesta de excepción se observa a continuación, el esclavo cambio el código de operación por 82 hex y agrego el código de excepción 02 en el campo de información.

Campo de información

	Dirección del esclavo	Código de operación	Código de excepción	CRC	
T1-T2-T3-T4	**02**	**82**	**02**	**D8 27**	**T1-T2-T3-T4**

El código de excepción 02 corresponde a dirección ilegal de datos (ver Tabla 16), indicando que la dirección de inicio de las entradas en la trama del maestro no es la indicada para la marca de PLC KOYO, por lo tanto este responde con la respuesta de excepción.

3.3.3 Campo de Información

Éste campo se forma a través de conjuntos de dos dígitos hexadecimales, en el rango 00-FF (recuerde que en modo ASCII cada byte del campo de datos es convertido a código ASCII, ocupando el doble de longitud que en modo RTU).

En las consultas del maestro a los esclavos, el campo de datos contiene información adicional para que el esclavo pueda llevar a cabo la petición del maestro.

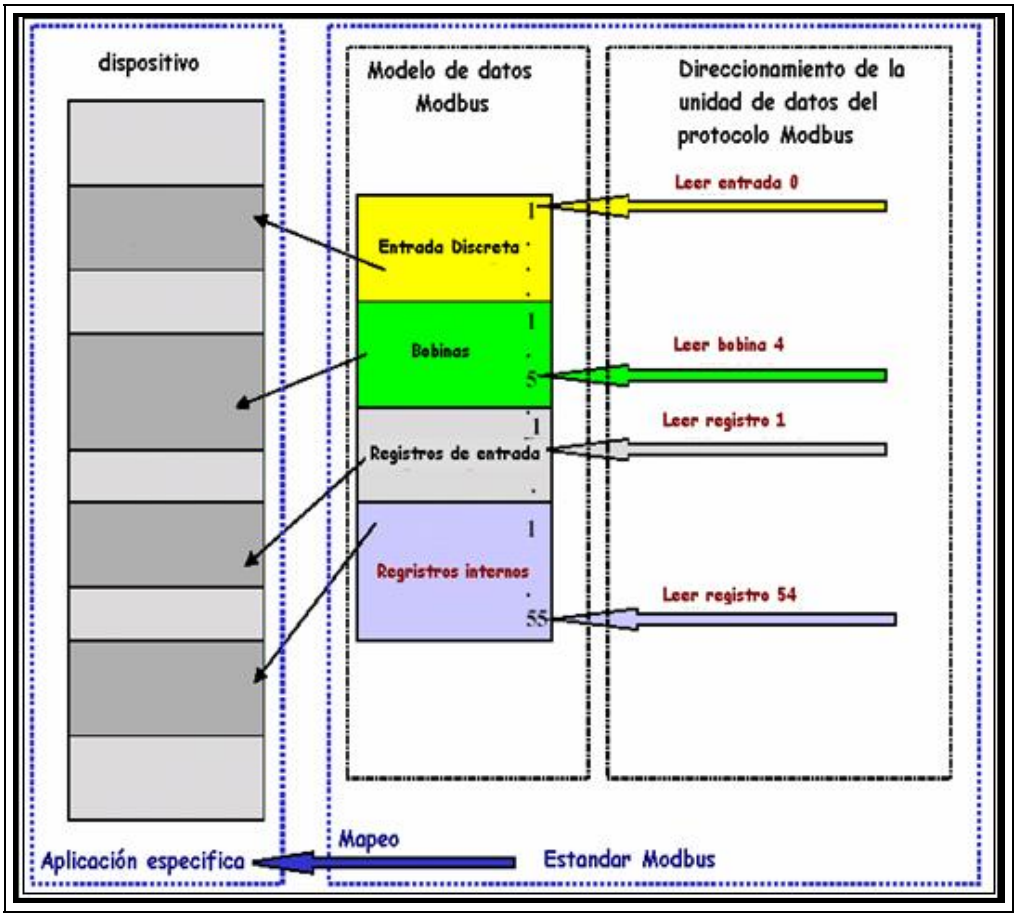
Modelo de direccionamiento Modbus

El modelo de direccionamiento Modbus esta integrado por 4 bloques de datos estos son (ver Figura 36):

- Bobinas (salidas discretas)
- Entradas discretas
- Registros internos
- Registros de entrada

Cada dispositivo que soporta el protocolo Modbus tiene su propio mapa de memoria, el direccionamiento de cada bloque esta contenido entre el rango de 0 a 65535.

Figura 36. Modelo de direccionamiento Modbus



Fuente: www.Modbus.org

Mapa de Memoria de los PLC's T100M+ TRILOGI

Tabla 17. Mapa de memoria de los TRILOGI

Elementos		Palabras MODBUS	Bit MODBUS
		Mapa direccionamiento	Mapa direccionamiento
Entradas	1-64	40001.1 - 40001.16 40002.1 - 40002.16 40003.1 - 40003.16 40004.1 - 40004.16 40005.1 - 40005.16 40006.1 - 40006.16	1 - 16 17- 32 33 - 48 49 - 64 65 - 80 81 - 96
Salidas (Bobinas)	1-64	40017.1 - 40017.16 40018.1 - 40018.16 40019.1 - 40019.16 40020.1 - 40020.16 40021.1 - 40021.16 40022.1 - 40022.16	257 - 272 273 - 288 289 - 304 305 - 320 321 - 336 337 - 352
Temporizadores	1-64	40033.1 - 40033.16 40034.1 - 40034.16 40035.1 - 40035.16 40036.1 - 40036.16	513 - 528 529 - 544 545 - 560 561 - 576
Contadores	1-64	40049.1 - 40049.16 40050.1 - 40050.16 40051.1 - 40051.16 40052.1 - 40052.16	769 - 784 785 - 800 801 - 816 817 - 832
Relés	1-256	40065.1-40065.16 40066.1 to 40066.16 40067.1 to 40067.16 40068.1 to 40068.16 40069.1 to 40069.16 40070.1 to 40070.16 40071.1 to 40071.16 40072.1 to 40072.16 40073.1 to 40073.16 40074.1 to 40074.16 **.....** 40097.1 to 40097.16	1025 a 1040 1041 a 1056 1057 a 1072 1073 a 1088 1089 a 1104 1105 a 1120 1121 a 1136 1137 a 1152 1153 a 1168 1169 a 1184 ..... 1521 a 1536
Valor presente de los temporizadores	1-64	40129 – 40192	
Valor presente de los contadores	1-64	40257 -40320	
Reloj	1-4	40513 – 40515	
Fecha		40517 – 40520	
Memoria de datos	1 - 4000	41001 – 45000	

Fuente: User's Manual, Super Programmable Controllers T100MD+.

En PLC's TRILOGI las funciones de lectura y escritura de bits (funciones: 01-02-05) se pueden realizar sobre todos los elementos sombreados de la Tabla 17, dado que se conoce la posición de memoria Modbus para acceder a cada bit.

Las funciones de lectura y escritura de registros (funciones: 03-04-06-16) se peden realizar sobre todos los elementos de la Tabla 17 ya que se conoce el direccionamiento Modbus de cada registros.

Mapa de Memoria MODICON-TELEMECANIQUE

El posicionamiento de memoria no se conoce, las funciones de lectura y escritura de bits se pueden realizar sobre los bits internos %M (que van de %M0 hasta %M127); y las funciones de lectura y escritura de registros se pueden realizar sobre las palabras internas %MW (que van de %MW0 hasta %MW255).

Mapa de Memoria de los PLC's KOYO DL-250-1

Tabla 18. Mapa de memoria de los KOYO

Elementos	Cantidad	Rango PLC (Octal)	Rango de Direcciones MODBUS (decimal)	Tipo de datos MODBUS
Entradas (X)	512	X0-X777	2048-2559	Entradas
Salidas (Y)	512	Y0-Y777	2048-2559	"Bobinas"
Temporizadores (T)	256	T0-T377	6144-6399	"Bobinas"
Contadores (CT)	128	CT0-CT177	6400-6527	"Bobinas"
Relés Control (C)	1024	C0-1777	3072-4095	"Bobinas"
Relés especiales (SP)	512	SP0-SP777	3072-3583	"Entradas"
Etapas (S)	1024	S0-S1777	5120-6143	"Bobinas"
Valor presente de los temporizadores	256	V0-V377	0-255	"Registros de Entrada"
Valor presente de los contadores	128	V1000-V1177	512-639	"Registros de Entrada"
Palabras de datos	7168	V1400-V7377 V10000-V17777	768 – 3839 4096 – 8191	Registros Internos

Fuente: AUTOMATIONDIRECT. DL205 User Manual Volume 1 of 2. D2–USER–M

3.3.4 Campo de Comprobación de Error

En redes estándar Modbus se usan dos tipos de comprobación de error. Que dependen del modo de transmisión Modbus de la red:

- Modo ASCII: El campo de comprobación de error contiene dos caracteres ASCII. El valor es el resultado de un Chequeo de Redundancia Longitudinal (LRC) basado en el contenido del mensaje, excluyendo el símbolo ASCII de inicio (:) y los caracteres de control ASCII finales CR-LF (retorno de carro-salto de línea). Los caracteres de LRC se añaden al mensaje como último campo seguidos de los caracteres CR-LF.
- Modo RTU: El campo de comprobación de error contiene dos bytes. El valor es el resultado de un cálculo de Chequeo de Redundancia Cíclica basado en el contenido del mensaje. Los caracteres de CRC se añaden en el último campo del mensaje.

Cálculo del Chequeo de Redundancia Longitudinal LRC

El procedimiento para la generación del LRC es:

- *Paso 1*: Sumar todos los bytes del mensaje, excluyendo el carácter de inicio (:) y los dos caracteres finales CRLF. Dejar el resultado en un campo de 8 bits, descartando todos los acarreos.
- *Paso 2*: Hallar el complemento a uno, restando el valor FF hex al resultado.
- *Paso 3*: Suma 1 al resultado para realizar el complemento a dos.

Cálculo del Chequeo de Redundancia Cíclica CRC

El procedimiento para la generación del CRC es:

- *Paso 1*: Se cargar un registro de 16-bit con FFFF **h**. Llamando a éste el registro CRC.

- *Paso 2:* Se realiza una OR Exclusiva entre el primer byte del mensaje con el byte menos significativo del registro CRC, poniendo este resultado en el registro CRC.

- *Paso 3:* El registro CRC se desplaza hacia la izquierda una posición y se rellena con cero la posición del bit mas significativo.

- *Paso 4:* Se examina el bit que salió, si es igual a cero se repite el paso 3, si es igual a uno, se realiza una OR exclusiva entre el registro CRC y el valor fijo A001 **h** (1010 0000 0000 0001).

- *Paso 5:* Se repite el paso tres y cuatro hasta realizar 8 desplazamientos.

- *Paso 6:* Se repiten los pasos 2….5 para el siguiente byte del mensaje. Esto continúa para todos los bytes del mensaje.

El resultado final del registro CRC contiene el valor del CRC.

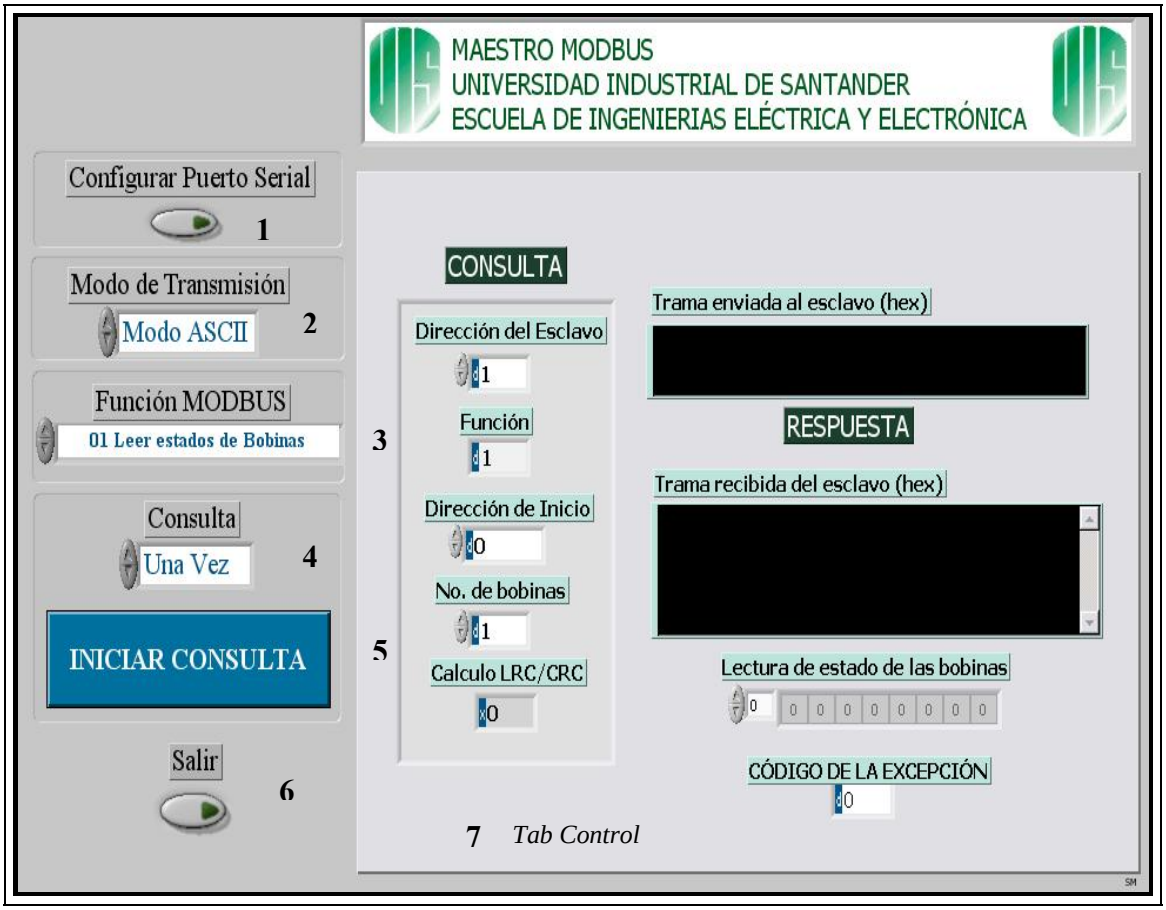
- *Paso 7:* Cuando el CRC es colocado dentro del mensaje, se debe tener en cuenta que se transmite primero el byte de menor orden seguido del byte de mayor orden.

4. IMPLEMENTACIÓN DEL PROTOCOLO MODBUS EN LABVIEW

En este capítulo se explica como se construye la trama Modbus consulta (siendo ésta la función del Maestro de la red) para las principales funciones del Protocolo Modbus y la captura de la trama respuesta (que es entregada por el esclavo consultado) empleando LabVIEW.

4.1 PANEL FRONTAL

Figura 37. Panel Frontal en LabVIEW

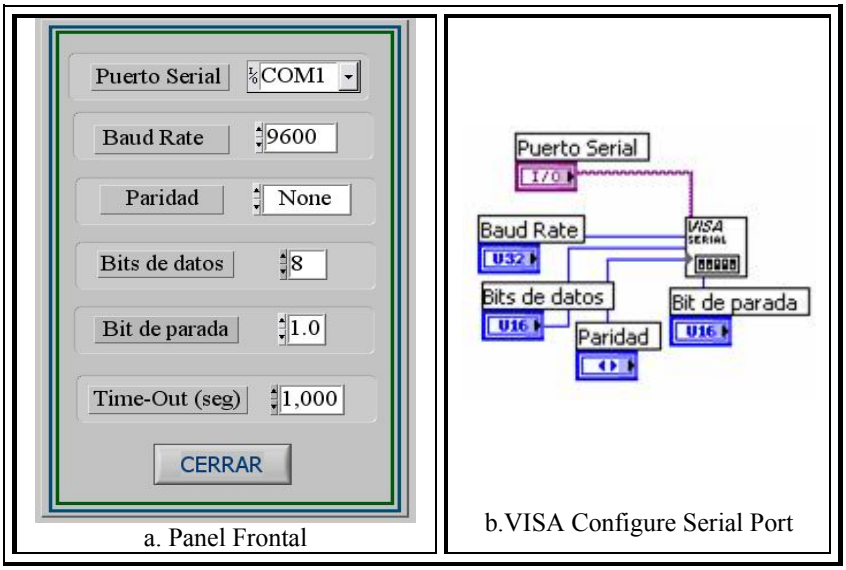


En el panel frontal de la Figura 37 se enumeran los controles donde se eligen los parámetros para iniciar la comunicación con un dispositivo que soporte el protocolo Modbus. A continuación se explican cada uno de ellos:

1) Configuración del puerto serial:

En esta sección del panel frontal se tiene la opción de elegir algunos de los **puertos serie** del PC (com1, com2), la **velocidad del puerto** *(Baud rate)*, la **paridad** (*none* "ninguna", *odd* "impar", *even* "par"), el numero de **bits de datos** *(7 ó 8)*, el **bit de parada** (1, 1.5, 2) y el *time-out*[1], ver figura 38.a.

Figura 38. Configuración del puerto serie



a. Panel Frontal

b.VISA Configure Serial Port

La configuración del puerto se realizó con el vi *VISA Configure Serial Port* ver Figura 38.b; VISA (*Virtual Instrument System Architecture)* es el estándar de arquitectura de los sistemas de instrumentación virtual que define los estándares de software I/O para VXI *(VMEbus eXtensions for Instrumentation),* GPIB *(General Purpose Interface Bus*), Serial, y otras interfaces. Está apoyado por mas de 35 grandes compañías de instrumentación incluyendo *Tektronix, Hewlett-Packard,* y *National Instruments*.

La comunicación en el protocolo Modbus, a nivel físico, es serial y emplea el estándar RS-485. Como se explicó en el capítulo 2, se requiere una línea de control del puerto serie RS-232 del PC para habilitar el integrado ADM485 del conversor RS-232 a RS-485, que

[1] Time-Out: tiempo de llegada de bytes al puerto serial.

permita en un determinado tiempo al PC enviar una trama por la red y en otro tiempo capturar la trama que viaja por la red; por tal razón se utiliza el estándar de arquitectura VISA para controlar la línea RTS del puerto serie.

2) Configuración del Modo de transmisión serial

Se programó un selector donde se elige el modo de transmisión (ASCII o RTU).

3) Elección de la Función Modbus

La función Modbus que se desee utilizar es seleccionada a través del selector llamado **Función Modbus**, al ser seleccionar alguna función se cambia la ventana del *tab control*.

Las funciones Modbus programadas son:

- 01 Leer estado de las "Bobinas".
- 02 Leer estado de las Entradas.
- 03 Leer Registros Internos.
- 04 Leer Registros de Entrada.
- 05 Forzar una "Bobina".
- 06 Fijar un Valor en un Registro.
- 07 Leer estado de la Excepción.
- 08 Diagnóstico.
- 15 Forzar multiples "Bobinas".
- 16 Fijar multiples registros.

4) Consulta

Este selector contiene dos opciones:

- Una Vez: Se envía solo una vez el mensaje consulta y seguidamente se captura la respuesta del esclavo.
- Continua: Esta opción envía el mensaje consulta, captura el mensaje respuesta y realiza esta operación indefinidamente. El usuario tiene la opción de configurar el *Time-Scan*, siendo este un retardo para

volver a enviar el mensaje consulta después de recibir la trama respuesta.

5) El botón **iniciar consulta,** si el programa esta en *RUN* inicia la comunicación con el dispositivo direccionado.

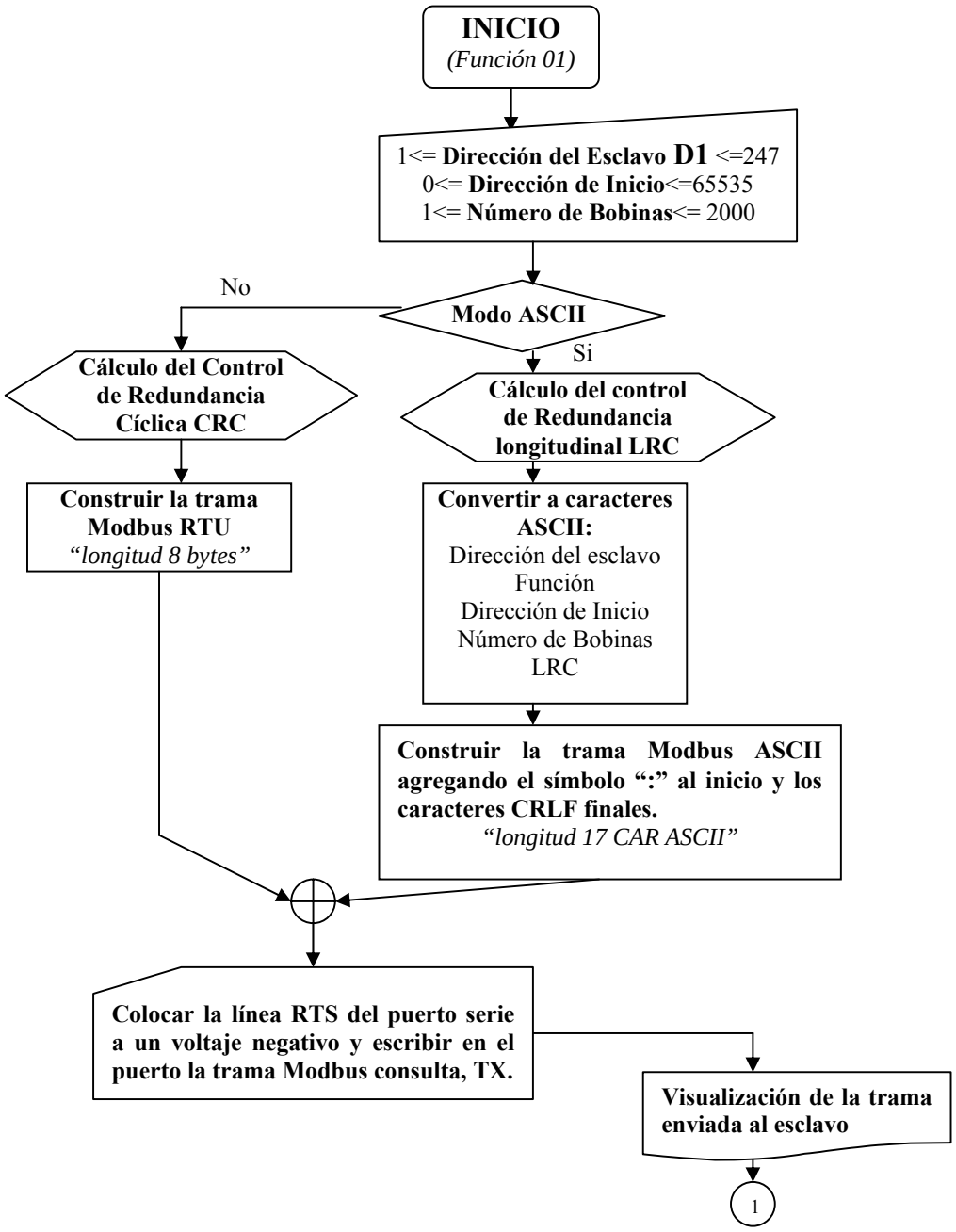
6) El botón **Salir,** cierra LabVIEW.

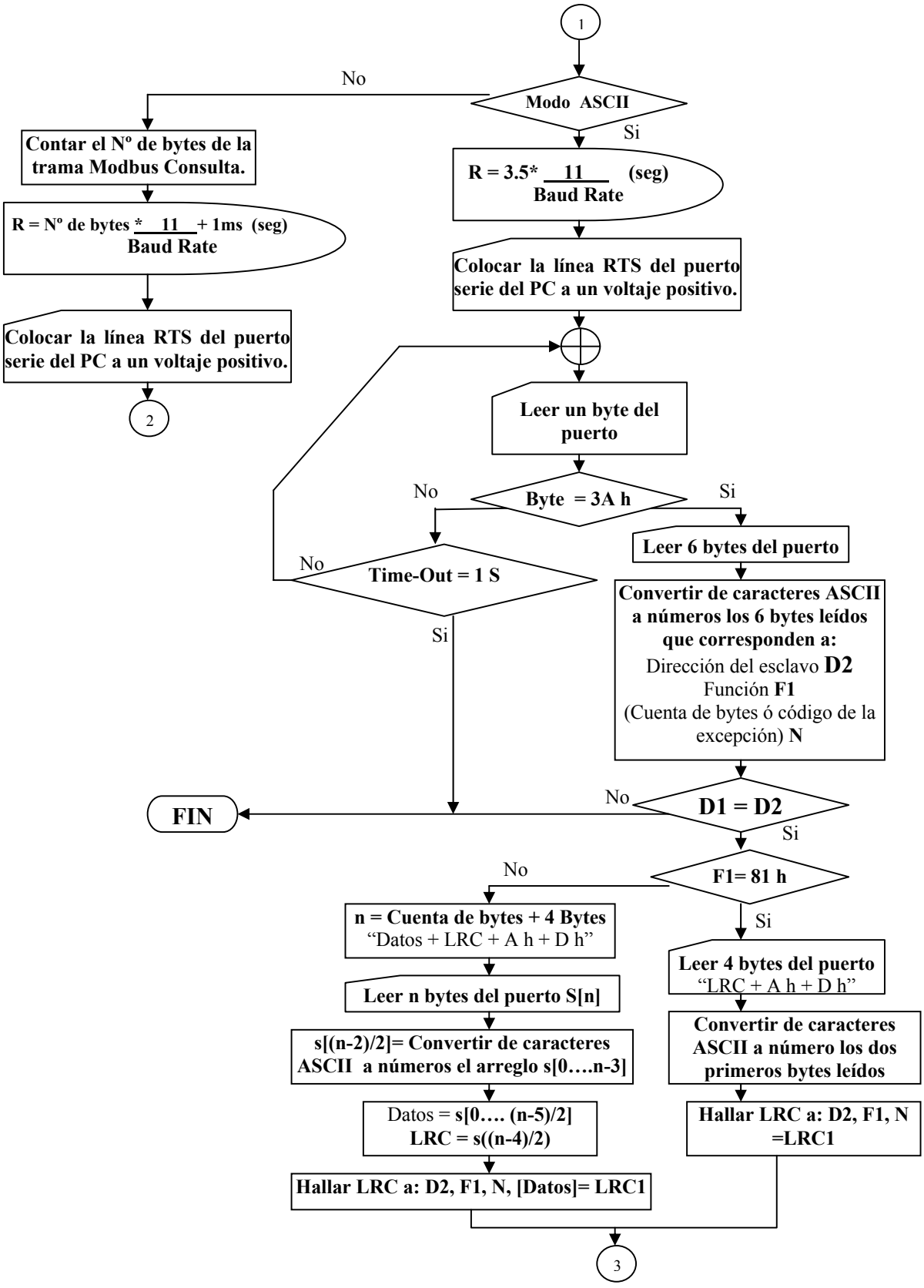
7) El **Tab Control** cambia de ventana a la función elegida a través del control llamado Función Modbus del numeral 3.

4.2 FUNCIÓN 01: LEER ESTADO DE LAS "BOBINAS"

La función 01 del protocolo Modbus corresponde a la lectura del estado ON/OFF de las "bobinas" discretas en un esclavo. En la Figura 39 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 01.

Figura 39. Diagrama de flujo correspondiente a la Función 01




1
No
Modo ASCII
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = 3.5* 11 (seg)
Baud Rate
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
Leer un byte del puerto
No
Byte = 3A h
Si
No
Time-Out = 1 S
Si
Leer 6 bytes del puerto
Convertir de caracteres ASCII a números los 6 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
(Cuenta de bytes ó código de la excepción) N
FIN
No
D1 = D2
Si
No
F1= 81 h
Si
n = Cuenta de bytes + 4 Bytes
"Datos + LRC + A h + D h"
Leer 4 bytes del puerto
"LRC + A h + D h"
Leer n bytes del puerto S[n]
Convertir de caracteres ASCII a número los dos primeros bytes leídos
s[(n-2)/2]= Convertir de caracteres ASCII a números el arreglo s[0….n-3]
Datos = s[0…. (n-5)/2]
LRC = s((n-4)/2)
Hallar LRC a: D2, F1, N =LRC1
Hallar LRC a: D2, F1, N, [Datos]= LRC1
3

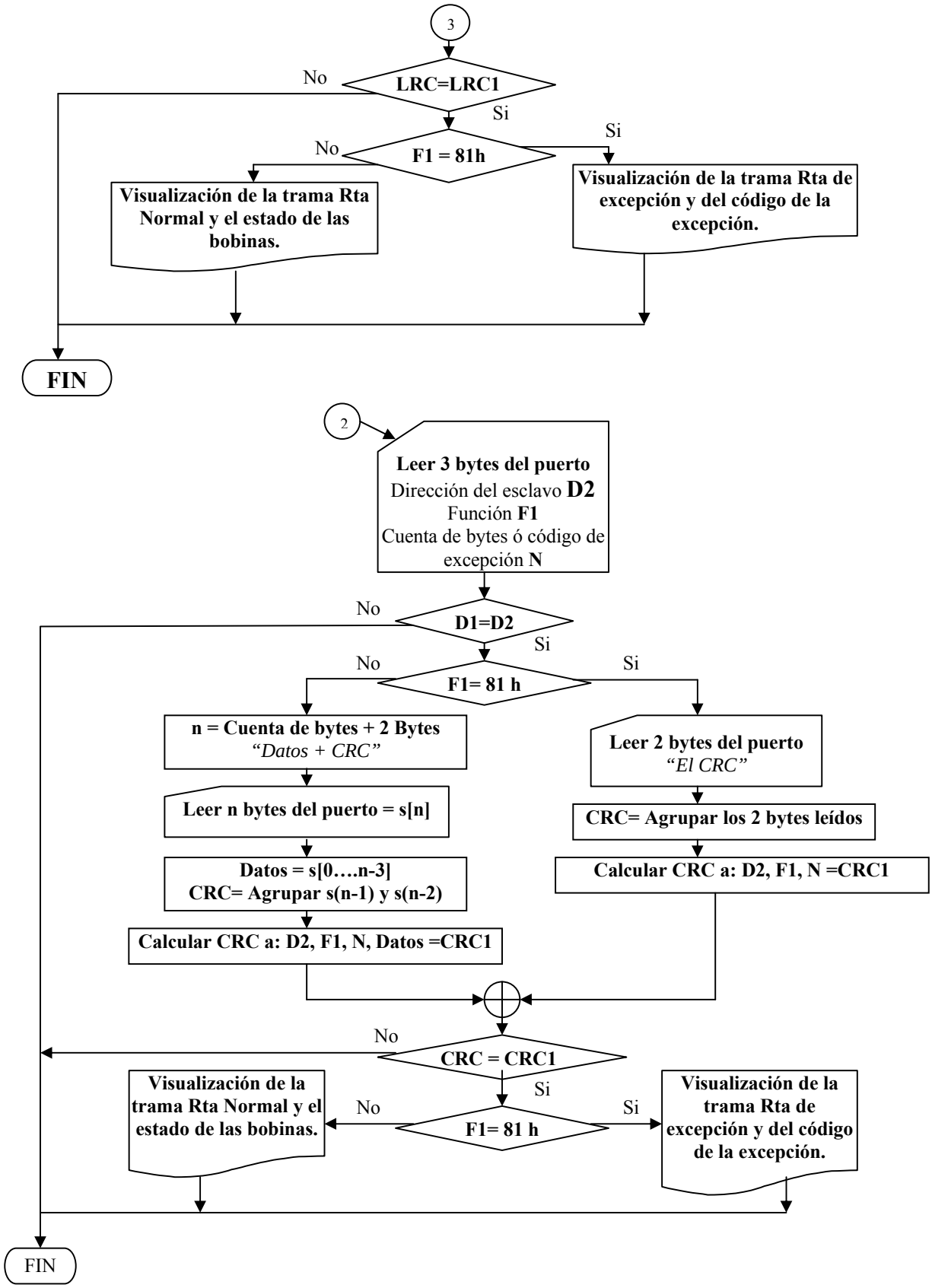

3
No
LRC=LRC1
Si
No
F1 = 81h
Si
Visualización de la trama Rta Normal y el estado de las bobinas.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN
2
Leer 3 bytes del puerto
Dirección del esclavo D2
Función F1
Cuenta de bytes ó código de excepción N
No
D1=D2
Si
No
F1= 81 h
Si
n = Cuenta de bytes + 2 Bytes
"Datos + CRC"
Leer n bytes del puerto = s[n]
Datos = s[0….n-3]
CRC= Agrupar s(n-1) y s(n-2)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 2 bytes del puerto
"El CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, N =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal y el estado de las bobinas.
No
F1= 81 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

Figura 40. Panel frontal correspondiente a la función 1 del protocolo Modbus

En la Figura 40 se observa la sección del panel frontal correspondiente a la función 01; para iniciar la consulta, el usuario, debe ingresar la siguiente información:

1) **Dirección del esclavo:** Es la identificación del dispositivo al que va dirigido el mensaje consulta, (1-247 d).
2) **Función:** código de operación de la función Modbus igual a 1 d (no se introduce información).
3) **Dirección de inicio:** Corresponde a la posición de memoria del dispositivo donde se encuentra la información referente al estado de las bobinas (0-65535 d) y desde la cual se empieza a leer, en el capítulo 3 se encuentran los mapas de memoria para los PLC's KOYO y TRILOGI.

Por ejemplo, la dirección donde se encuentra la información referente a la salida 3 en los PLC's KOYO y TRILOGI es:

KOYO: Salida 0 (Y0) tiene la dirección Nº 2048 d (ver Tabla 18) por lo tanto la salida 3 (Y3) tiene la dirección Nº 2051 d.

Y0	Y1	Y2	Y3	Y4	Y5	Y10	Y11	Y12	Y13	Y14	Y15
2048	2049	2050	**2051**	2052	2053	2054	2055	2056	2057	2058	2059

TRILOGI: Salida 0 tiene la dirección Nº 257 en el mapa de memoria (ver Tabla 17), observe que el posicionamiento empieza en 1, por lo tanto se resta uno a la dirección, obteniendo que la dirección de la salida 0 es la Nº 256 y la de la salida 3 es la Nº 259.

O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
256	257	258	**259**	260	261	262	263	264	265	266	267	268	269	270

4) **Nº. de bobinas:** Número bobinas (bits) consecutivas que se quieren consultar (1-2000 d).
5) **CRC ó LRC:** Cálculo del control de errores, es calculado por el programa con la información que ingrese el usuario en los numerales 1, 2, 3 y 4. El resultado se presenta en hexadecimal.

Al activar el botón iniciar consulta en el panel frontal de la Figura 37, se visualizan los siguientes datos:

6) **Arreglo enviado al esclavo:** En este "*display*" se visualiza el arreglo de caracteres del mensaje consulta enviado por el PC. Se presenta en hexadecimal.

Cuando el esclavo responde se visualiza:

7) **Arreglo recibido del esclavo**: En este "*display*" se visualiza el arreglo de caracteres del mensaje respuesta que envía el esclavo que ha sido consultado. Se presenta en hexadecimal.

8) **Lectura de estado de bobinas**: En este indicador (unos y ceros) se visualiza el estado de las bobinas del esclavo que fueron solicitadas por el usuario en el mensaje consulta. El estado 1 corresponde a ON y el estado 0 a OFF.

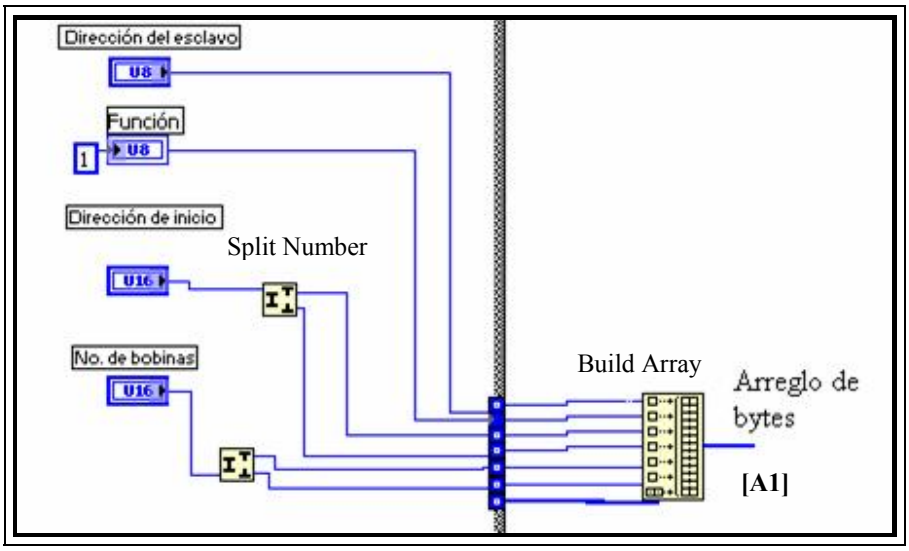
El bit menos significativo del primer byte del campo de información contiene la salida direccionada (dirección de inicio) en el mensaje consulta, ver Figura 41.

Figura 41. Indicador del estado de las bobinas

9) **Código de Excepción:** Este indicador contiene el codigo de la excepción si se recibe una respuesta de excepción.

4.2.1 Construcción de la trama consulta para la Función 01 en modo ASCII

Figura 42. Operación de concatenar el campo del formato del mensaje consulta



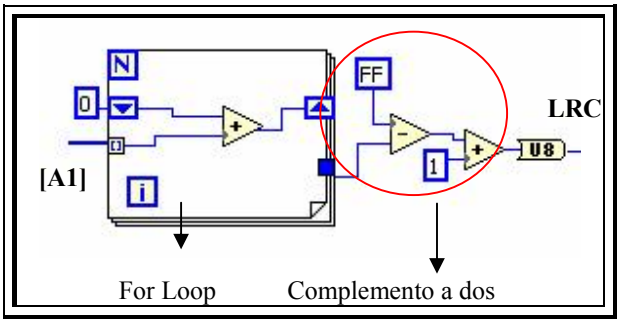
Los elementos del formato del mensaje (dirección del esclavo, función, dirección de inicio y el Nº de bobinas) son concatenados en un arreglo de bytes, con la función *Build Array*,

ver Figura 42. Teniendo los cuatro primeros elementos del mensaje consulta como un arreglo de bytes (llamado [A1]) se procede a calcular el LRC.

❖ **Cálculo del control de redundancia longitudinal (LRC):**

En la Figura 43 se encuentra la rutina desarrollada para el cálculo del LRC. El procedimiento es el siguiente: el arreglo de bytes A1 entra en la estructura *For Loop*, en ésta se realiza una suma binaria con todos los bytes; luego se calcula el complemento a dos al resultado de la suma y se empaqueta en un byte despreciando el acarreo, este nuevo resultado es el LRC.

Figura 43. Calculo del LRC



Ejemplo del cálculo del LRC:

Dirección del esclavo	Función	Dirección de inicio	Número de puntos	LRC
21 d	01d	2048 d	0016 d	**210 d**
15 $_{h}$	01 $_{h}$	08 00 $_{h}$	0010 $_{h}$	**D2 $_{h}$**

1 5		Dirección del esclavo
0 1		Función
0 8	+	Dirección de inicio Hi
0 0		Dirección de inicio Lo
0 0		Número de puntos Hi
1 0		Número de puntos Lo
2 E $_{h}$		

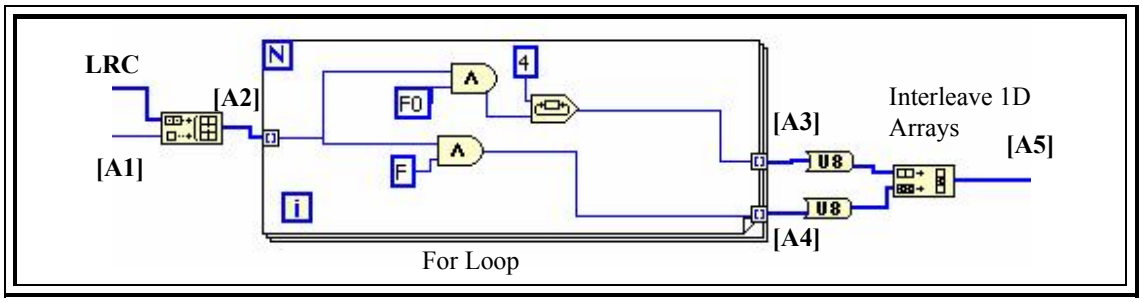
Luego se calcula el complemento a dos:

LRC = FF – **2E** + 0 1 = **D 2 $_{h}$**

En modo ASCII cada byte del mensaje se convierte en dos caracteres ASCII, a continuación se explica por pasos el procedimiento realizado en LabVIEW para hacer la conversión.

- *Paso 1: Se* forma un nuevo arreglo de datos A2 concatenando el arreglo A1 y el LRC, ver Figura 44.
- *Paso 2:* El arreglo A2 ingresa al *For Loop,* ver Figura 44.
- *Paso 3: S*e realiza una operación AND entre cada byte del arreglo A2 y la máscara F0 $_{h}$ (11110000), el resultado de esta operación se rota 4 posiciones, obteniendo un nuevo arreglo A3 que contiene los cuatro bits más significativos de cada byte del arreglo A2. Al mismo tiempo se realiza otra operación AND entre cada byte del arreglo A2 con la máscara 0F $_{h}$ (00001111) para obtener un nuevo arreglo A4 que contiene los cuatro bits menos significativos de cada byte del arreglo A2, ver Figura 44.

Figura 44. Rutina para separar un byte en dos bytes



Ejemplo:

21 = **00010101** = 15 $_{h}$

Rota 4 posiciones

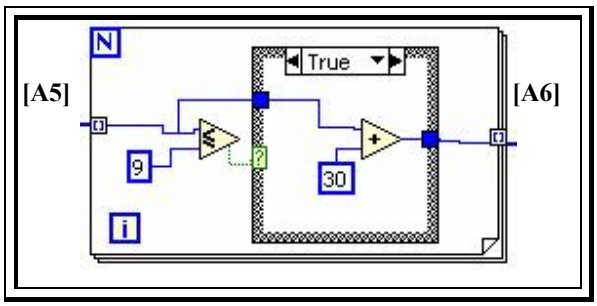
00010101 AND **1111**0000 = **0001**0000 ⟹ 0000**0001** = **01** $_{h}$

00010101 AND 0000**1111** = 0000**0101** = **05** $_{h}$

- *Paso 4:* Los arreglos A3 y A4 se intercalan con la función *Interleave 1D Arrays,* para formar un nuevo arreglo A5, ver Figura 44.

❖ *Paso 5: Se* procede a transformar el arreglo de bytes A5 en una arreglo de caracteres ASCII. Para esto cada byte del arreglo entra a la estructura *For Loop* (ver Figura 45), en esta se pregunta si el número es menor o igual a 09 $_h$, si lo es, al byte se le suma el número 30 $_h$ (48 d), si no, se le suma el número 37 $_h$ (55d). Al final se obtiene un nuevo arreglo de bytes A6 codificado en ASCII.

Figura 45. Conversión a caracteres ASCII 



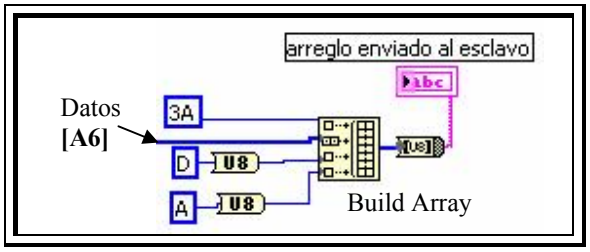
Ejemplo:

Suponga que se tiene el siguiente arreglo en hexadecimal: [07 $_h$ 0C $_h$ 03 $_h$ 0B $_h$ 00 $_h$]

07 $_h$	0C $_h$	03 $_h$	0B $_h$	00 $_h$	
30 $_h$	37 $_h$	30 $_h$	37 $_h$	30 $_h$	+
37 $_h$	43 $_h$	33 $_h$	42 $_h$	30 $_h$	Código ASCII

Y se obtiene arreglo en código ASCII: 37 $_h$ 43 $_h$ 33 $_h$ 42 $_h$ 30 $_h$

Figura 46. Trama enviada al esclavo

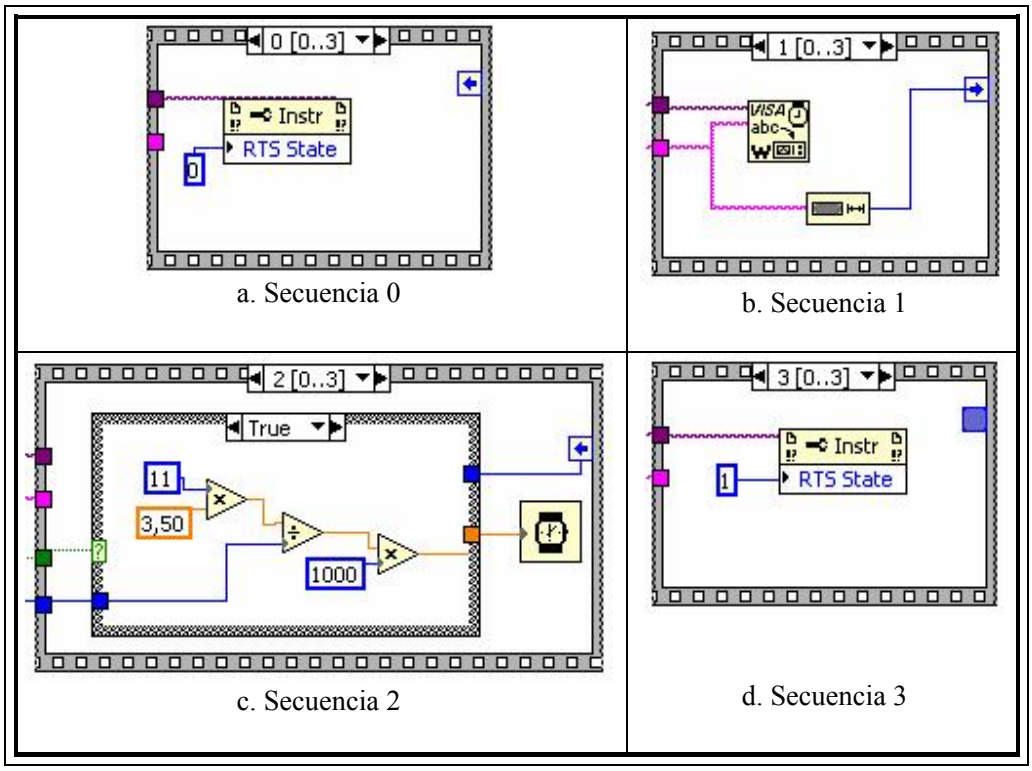


Teniendo los datos como caracteres ASCII, se les agrega el símbolo carácter ASCII dos puntos ":" (3A $_h$ ó 58 d) que indica el inicio de la trama, y los caracteres ASCII finales retorno de carro y salto de línea "CR" y "LF" que corresponde a D $_h$ y A $_h$

respectivamente, la trama completa se concatena y se transforma en un *string* con la función *Byte Array To String* para ser escrita en el puerto, ver Figura 46.

Para escribir la trama en el puerto se realizó la siguiente secuencia, (ver Figura 47):

Figura 47. Secuencia realizada para escribir en el puerto



a. Secuencia 0

b. Secuencia 1

c. Secuencia 2

d. Secuencia 3

- ***Secuencia 0***: Se pone la función de nodo VISA *RTS STATE* en cero, (esto corresponde a un voltaje negativo en la línea RTS del puerto RS-232 del PC) habilitando el conversor RS 232 a RS 485 en modo *Driver*; para que el PC (Maestro de la Red) envie la trama a la red 485, ver Figura 47.a.

- ***Secuencia 1***: Se escribe en el puerto la trama, ver Figura 47.b.

- ***Secuencia 2***: Se genera un retardo igual a:

Tiempo de un carácter serie ASCII

$$R = 3.5 * \frac{11}{\text{Baud Rate}} \quad \text{(seg)}$$

Tiempo de un carácter serie ASCII ≈ (1 bit inicio, 8 bits datos, 1 bit paridad, 1 bit de parada)/Baud Rate

Este retardo garantiza que los datos lleguen a la salida del conversor RS-232 a 485 (específicamente a la salida del integrado ADM 485), antes de cambiar el estado de la línea RTS (continúe en modo *Driver* el conversor), ver Figura 47.c. Observe que el retardo equivale aproximadamente a 3.5 el tiempo de un carácter serie ASCII, este valor se obtuvo realizando pruebas a diferentes velocidades.

- ***Secuencia 3***: Se pone la función de nodo VISA *RTS STATE* en uno, (esto corresponde a un voltaje positivo para la línea RTS del puerto RS-232 del PC) habilitando el conversor RS 232 a 485 en modo *receiver*, para que el PC lea la repuesta del esclavo consultado ver Figura 47.d.

4.2.2 Captura de la trama Respuesta de un esclavo consultado con la Función 01 en modo ASCII

Después que el programa ha enviado el mensaje a los esclavos de la red se procede a capturar el mensaje respuesta del esclavo direccionado, que contiene la información solicitada, el tamaño de la trama respuesta depende del número de bobinas solicitadas por el usuario.

Longitud de la trama = 11 CAR ASCII + 2* N CAR (ver Figura 48); donde N corresponde a la cuenta de bytes.

Figura 48. El formato del mensaje respuesta de la función 01 en modo ASCII

Carácter inicial	Dirección del esclavo	Función	Cuenta de bytes = N	Datos	LRC	Caracteres finales
1 CAR ASCII **:**	**2** CAR ASCII	**2** CAR ASCII	**2** CAR ASCII	**2*N** CAR ASCII	**2** CAR ASCII	**2** CAR ASCII **CR LF**

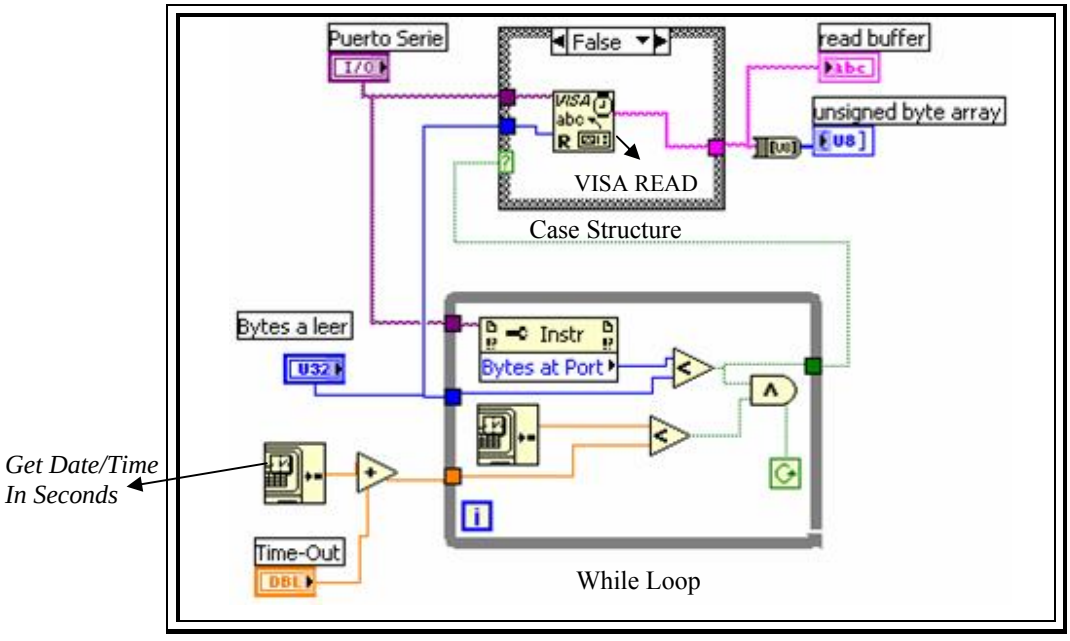
Primero se explica el subvi para leer el puerto serie, seguidamente la subrutina para leer el símbolo ASCII ':' y finalmente el programa para leer el resto de la trama.

SUBVI leer puerto serie

El subvi de la Figura 49 se utiliza en el programa cada vez que se requiere leer un número específico de bytes del puerto serie. Las funciones utilizadas en el subvi son:

- *Bytes at Port*: Cuenta los bytes del buffer del puerto serie.
- *Get Date/Time In Seconds*: Esta función retorna un valor en segundos que depende de la hora y fecha del PC, y se incrementa en segundos a medida que corre el programa.
- *VISA Read:* Lee un número específico de bytes del puerto serie.

Figura 49. Subvi leer puerto. 



Los datos que ingresan al subvi son:

- Puerto serie: contiene el número del puerto serie del PC (com1, com2……com9).
- Bytes a leer: este control contiene el número de bytes que se quieren leer del buffer del puerto, según las necesidades del programa.

- Time-Out: este control contiene un valor en segundos que representa un tiempo de espera para la llegada de bytes al puerto.

En la estructura *While Loop* se determina si el número del *Bytes at Port* es menor al dato que se encuentra en el control bytes a leer; y si el valor instantáneo del *Get Date/Time In Seconds* es menor al resultado de la suma entre el número del *Get Date/Time In Seconds* y el *time-out*. Los resultados lógicos de las comparaciones hechas en el *while loop* entran a una compuerta AND; si el resultado de la operación AND es verdadero el subdiagrama de la estructura *While Loop* se repetirá, indicando que:

- Los bytes contados en el puerto son menor a los bytes a leer ó
- No ha trascurrido el *time-out*, es decir que la suma de tiempos hecha fuera de la estructura es menor que el valor del *Get Date/Time In Second* dentro del *While Loop.*

Si no, se termina el ciclo del *While Loop;* ocurriendo alguna de estas posibilidades:

- El números de bytes contados es mayor a los bytes a leer; y se ingresa al caso *False* (Falso) de la estructura *Case Structure* en donde se procede a leer el número de bytes en el puerto serie con la función *VISA Read*.
- El tiempo del *time-out* se termina, y se ingresa a la condición *True* (verdad) de la estructura *Case Structure,* como no hay bytes suficientes para leer se entrega un *string* vacío.

Subrutina leer simbolo ASCII

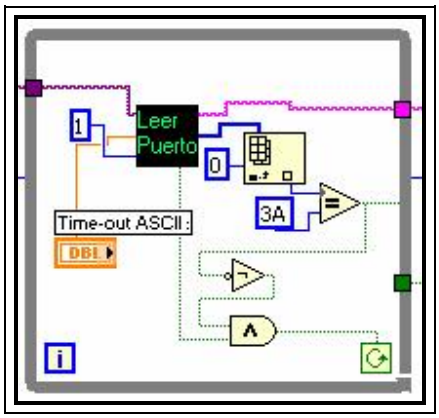
En modo ASCII la trama inicia con el símbolo ASCII dos puntos ':', en la Figura 50 se encuentra la subrutina para leer este símbolo, siendo el procedimiento el siguiente:

Dentro de la estructura *While Loop* se encuentra el subvi Leer Puerto 

explicado anteriormente, con este subvi se lee un byte del buffer y seguidamente se pregunta si este es igual al número 3A **h** (valor del símbolo ASCII :), si es verdad se sale de la estructura

While Loop indicando que hay una trama en modo ASCII para leer, si no se continúa preguntando por un tiempo hasta que se finalice el time-out.

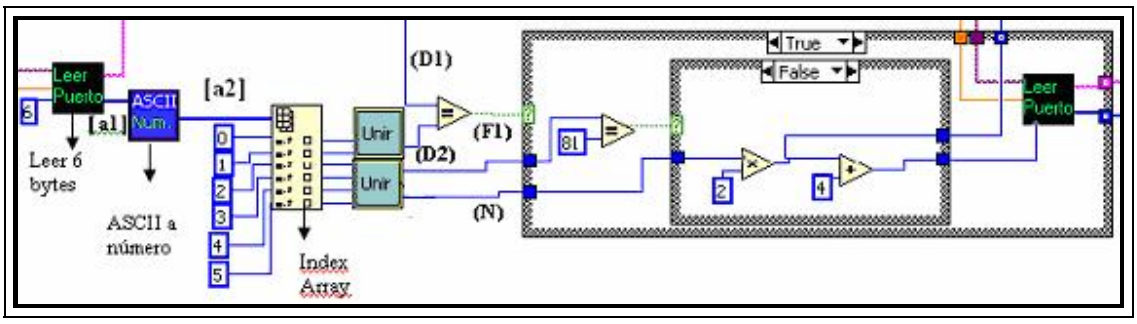
Figura 50. Leer el símbolo ASCII ":"



LEER TRAMA RESPUESTA

Luego de leer el símbolo ASCII ':' se procede a leer la trama respuesta, en la Figura 51 se encuentra esta sección del programa y se explicará por pasos:

Figura 51. Lectura de la trama respuesta



Paso 1: Leer 6 bytes del puerto, esto se hace con el subvi 

. Este arreglo de bytes **[a1]** corresponde a:

Dirección del esclavo (D2)	Función (F1)	Cuenta de bytes (N)
2 CAR ASCII	**2** CAR ASCII	**2** CAR ASCII

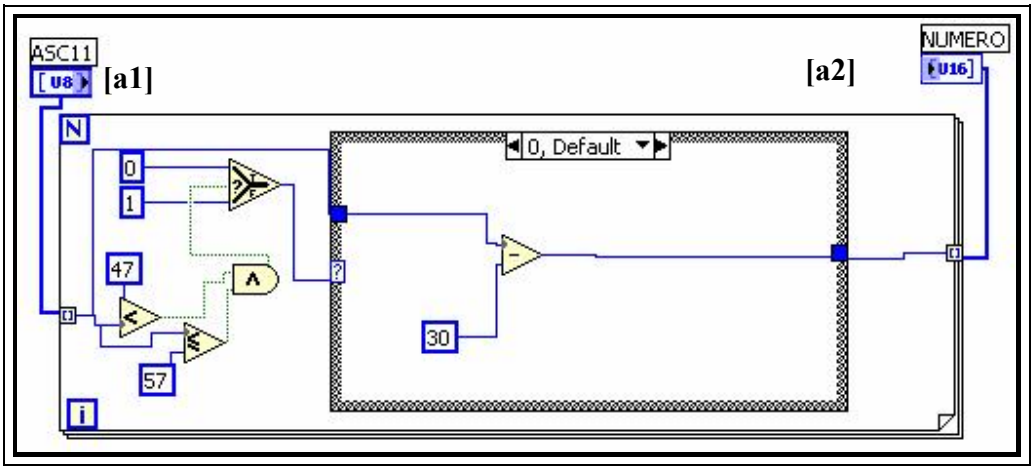
Paso 2: El arreglo de bytes **[a1]** debe ser decodificado de caracteres en código ASCII a números, en la Figura 52 se presenta el subvi que realiza esta función. El arreglo de bytes **[a1]** en ASCII entran a la estructura *For Loop* uno por uno, a cada uno se le pregunta si el dato es menor o igual que 39_h (57 d) si lo es, se selecciona el caso 0 del *Case Structure* y en éste al dato se le resta el número 30_h (48 d).

La otra opción es que el dato sea menor que $2F_h$ (47 d) y mayor o igual que 39_h (57 d), si es así se entra al caso 1 del *Case Structure* y al dato se le resta el número 37_h (55 d).

Ejemplo: Supongamos que se tiene la siguiente trama:

	38_h	45_h	30_h	33_h	44_h	35_h	31_h	Código ASCII
−	30_h	37_h	30_h	30_h	37_h	30_h	30_h	
	08_h	$0E_h$	00_h	03_h	$0D_h$	05_h	01_h	

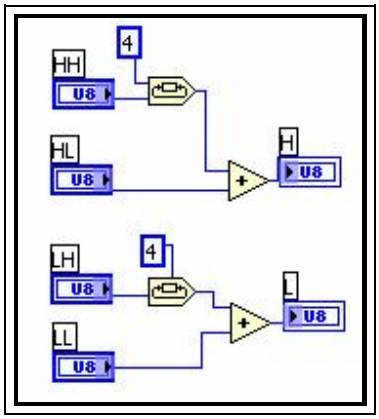
Figura 52. Decodificación de carácter ASCII a número



Paso 3: Con la función *Index Array* se extraen los 6 bytes del arreglo **[a2]**, ver Figura 51.

Paso 4: Note que cuando se codifica en ASCII cada byte de información se separa en dos bytes. Con el subvi de la Figura 53 se hace la operación inversa, es decir cada pareja de bytes se une en un byte. Para eso se toma los datos extraídos del arreglo **[a2]** en parejas de bytes; el byte más significativo se desplaza cuatro posiciones y se le suma el byte menos significativo.

Figura 53.Subvi Unir 



Ejemplo:

Paso 1: **[a1]** = 31_h 35_h 30_h 32_h 30_h 33_h código ASCII; Arreglo leído del puerto.

Paso 2: **[a2]** = 01_h 05_h 00_h 02_h 00_h 03_h; Decodificación ASCII/num.

Paso 4: 15_h 02_h 03_h ; Unión

Dirección del Esclavo (D2)	Función (F1)	Cuenta de bytes (N)
15_h	02_h	03_h

Paso 5: Se pregunta si la dirección del esclavo **(D2)** leída del puerto es igual a la que se envió en el mensaje consulta (digitada por el usuario **D1** = Dirección del esclavo), si es el esclavo correcto se entra al caso True de *Case Structure,* si no se entrega una cuenta de bytes igual a cero y por lo tanto no se lee el resto del mensaje, ver Figura 51. Esto se hace para verificar si el esclavo consultado es el que está respondiendo.

Paso 6: Si es el esclavo correcto, observe Figura 51, se pregunta si el código de operación (función Modbus) es igual a 81_h (Respuesta de excepción para la Función 01), si no lo es se tiene un respuesta normal y se entra en el caso falso del *Case Structure*, en donde se multiplica la cuenta de bytes por dos (dado que el campo de datos es el doble en el formato ASCII) y se le suma 4 bytes ya que para completar la trama se necesita el cálculo del LRC (2 CAR ASCII) y los caracteres finales A_h y D_h (2 CAR ASCII), ver Figura 54.

Figura 54. Tamaño del campo de datos, LRC y los caracteres finales en modo ASCII para la función 01

Datos	LRC	Caracteres finales
2*N CAR ASCII	**2** CAR ASCII	**2** CAR **CR LF**

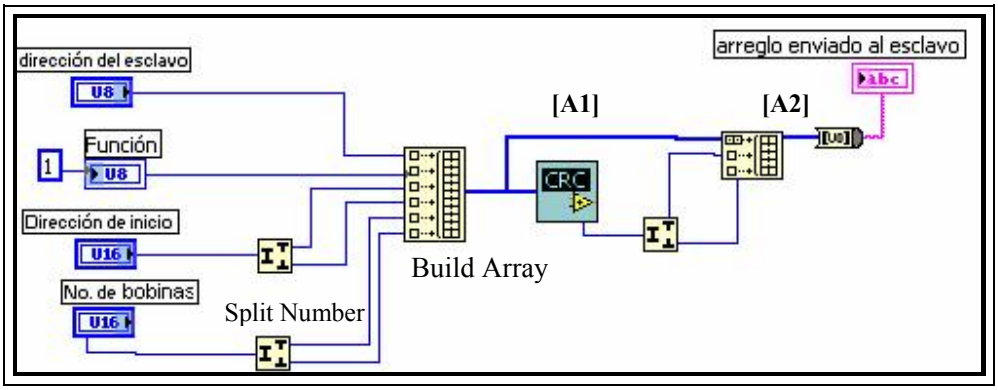
Paso 7: Finalmente con esta nueva cuenta de bytes (2*N + 4) se procede a leer el puerto; ver Figura 51.

El proceso descrito anteriormente se realiza cuando la respuesta es normal, en el caso que se produzca una respuesta de excepción (función **(F1)** = 81_h) ver Figura 51, el dato que se adquirió como cuenta de bytes corresponde ahora al código de excepción, y solo se necesitan 4 bytes para completar la trama que corresponden al LRC y a los caracteres finales (A_h y D_h).

Paso 8: Se verifica el LRC, y si es correcto se visualiza al usuario el estado de las bobinas.

4.2.3 Construcción de la trama consulta para la Función 01 en modo RTU

Figura 55. Construcción trama Modbus RTU

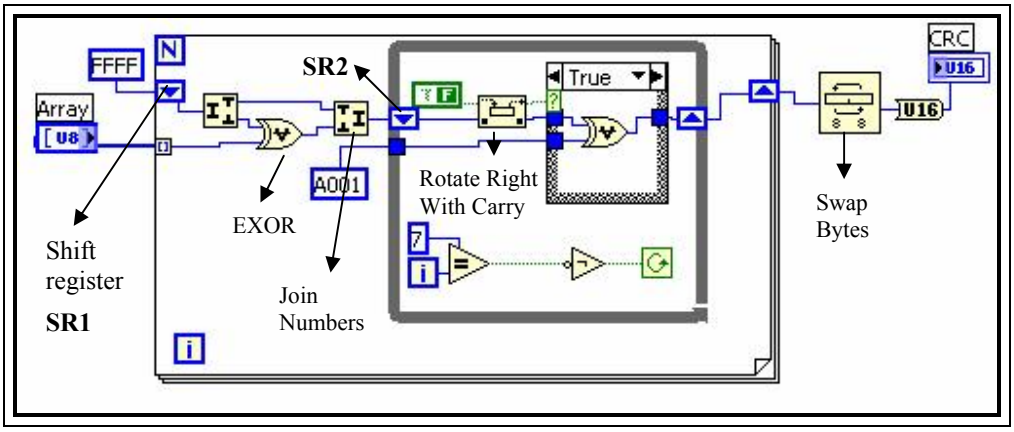


En modo RTU los datos que debe digitar el usuario son los mismos que en modo ASCII (dirección del esclavo, función, dirección de inicio y Nº de Bobinas); estos datos se

concatenan con la función *Build Array* para formar el arreglo A1, seguidamente a este arreglo se le calcular el CRC, ver Figura 55.

❖ **Cálculo de control de redundancia cíclica CRC**

Figura 56. Cálculo del CRC



El cálculo del CRC (ver Figura 56) se obtiene de la siguiente manera:

1. Se carga el S*hift Register* (registro temporal) **SR1** de la estructura *For Loop* con la palabra FFFF $_{h}$.

2. Dentro de la estructura *For Loop* se realiza una OR Exclusiva entre el primer byte de la trama (dirección del esclavo, en la primera iteración) con el byte menos significativo del *Shift Register* **SR1.**

3. Se agrupan el byte más significativo del *Shift Register* **SR1** con el resultado de la OR exclusiva a través de la función *Join Numbers* y se pone este dato en el S*hift Register* **SR2** de la estructura *For While.*

4. Dentro de la estructura *For While,* se desplaza el dato del S*hift Register* **SR2** hacia el bit menos significativo y se rellena con cero la posición del bit más significativo esta operación se realiza con la función *Rotate Right With Carry*

5. Se examina el LSB que salió, si es igual a cero se desplaza el dato del S*hift Register* **SR2** y se vuelve a preguntar; si es igual a uno, se realiza una OR exclusiva entre el S*hift Register* **SR2** y el valor fijo A001 $_{h}$ (1010 0000 0000 0001). Se repite hasta realizar 8 desplazamientos.

6. El dato final del S*hift Register* **SR2** se carga en el S*hift Register* **SR1.**

7. Se toma el siguiente byte de la trama y se realizan los pasos del 2 al 7 hasta completar toda la trama.

8. El resultado final del S*hift Register* **SR1** contiene el valor del CRC.

9. Finalmente con la función *Swap Bytes* se cambia el byte más significativo por el byte menos significativo; debido a que en la transmisión se envía primero el byte menos significativo seguido del byte más significativo.

Teniendo el cálculo del CRC se concatenan el arreglo A1 con el resultado del CRC, obteniendo un arreglo llamado A2 (ver Figura 55); este nuevo arreglo se convierte en una cadena de datos con la función *Byte Array To String,* estando lista la trama para ser escrita en el puerto serie.

Para escribir la trama en el puerto se realiza la siguiente secuencia:

- ***Secuencia 0***. Se pone la función de nodo VISA *RTS STATE* en cero, (esto corresponde a un voltaje negativo en la línea RTS del puerto RS-232 del PC) habilitando el conversor RS 232 a RS 485 en modo *Driver*; para que el PC (Maestro de la Red) sea el que envia la trama a la red 485, ver Figura 57.a.

- ***Secuencia 1***. Se escribe en el puerto la trama, y se cuenta el número de bytes de ésta con la función *String Length* , ver Figura 57.b.

- ***Secuencia 2.*** Se genera un retardo igual a:

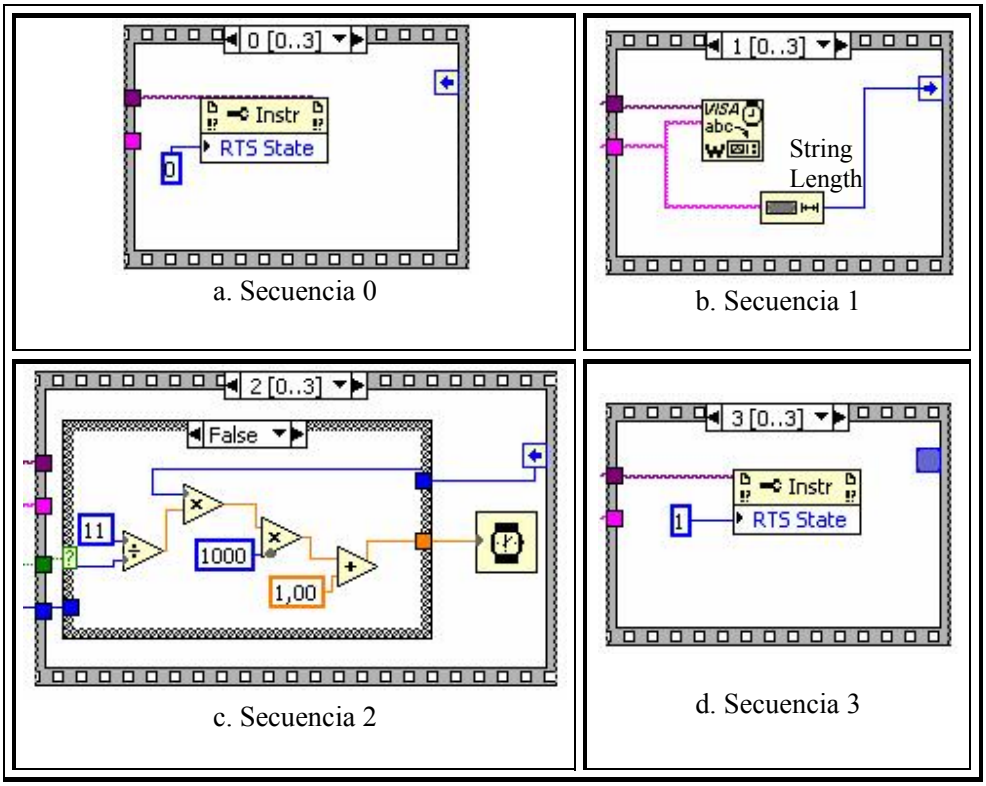
$$R = \text{Nº de bytes} * \frac{11}{\text{Baud Rate}} + 1\text{ms} \quad \text{(seg)}$$

Tiempo de un carácter serie RTU ≈ (1 bit inicio, 8 bits datos, 1 bit paridad, 1 bit de parada)/Baud Rate

Este retardo garantiza que los datos lleguen a la salida del conversor RS-232 a 485 (específicamente a la salida del integrado ADM 485), antes de cambiar el estado de la línea RTS (continue en modo *Driver* el conversor), ver Figura 57.c. Observe que el retardo equivale aproximadamente a N (número de bytes contados en el puerto) veces el tiempo de un carácter serie ASCII más 1ms, este valor se obtuvo realizando pruebas a diferentes velocidades.

- ***Secuencia 3***. Se pone la función de nodo VISA *RTS STATE* en uno (esto corresponde a un voltaje positivo para la línea RTS del puerto RS-232 del PC) habilitando el conversor RS 232 a 485 en modo *receiver*, para que lea la repuesta del esclavo consultado ver Figura 57.d.

Figura 57. Secuencia para escribir en el puerto en modo RTU



a. Secuencia 0

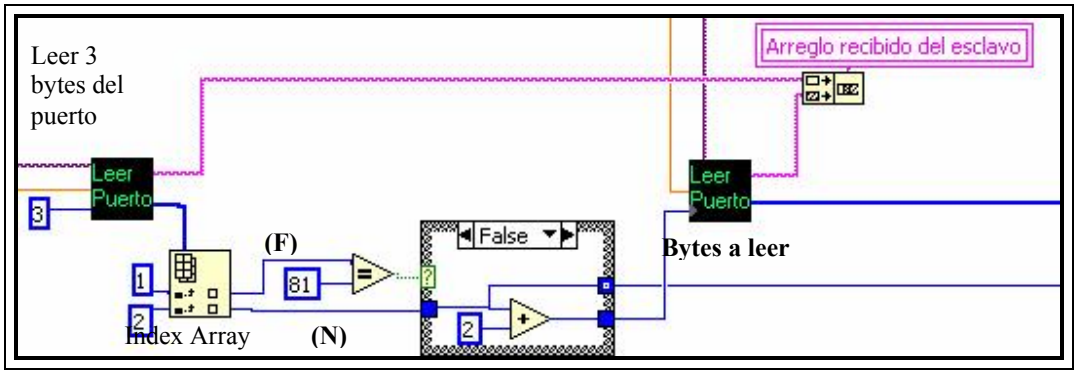
b. Secuencia 1

c. Secuencia 2

d. Secuencia 3

4.2.4 Captura de la trama respuesta de un esclavo consultado con la Función 01 en modo RTU

Figura 58. Lectura de la trama RTU para la función 01



El procedimiento de lectura de la trama respuesta enviada por un esclavo es el siguiente (ver Figura 58):

- Se leen los tres primeros bytes del puerto que corresponden a la ***dirección del esclavo***, el código de la ***función*** Modbus y la ***cuenta de bytes.***
- Se extrae del arreglo de bytes con la función *Index Array* el código de función **F** y la cuenta de bytes **N**.
- Se pregunta si el código de función es igual a 81_h ,si no lo es, se calcula el número de bytes a leer en el puerto para completar la trama, que corresponde a:

Bytes a leer **n** =N (cuenta de bytes del mensaje) + 2 bytes (corresponden al CRC)

Si es una respuesta de excepción solo resta por capturar los bytes que corresponden al CRC. **n** = 2 bytes

- Se leen los **n** bytes del puerto.
- Se verifica el CRC de la trama capturada.
- Finalmente se visualiza el estado de las bobinas pedidas por el usuario.

4.3 FUNCIÓN 02: LEER ESTADO DE LAS ENTRADAS

La función 02 del protocolo Modbus permite la lectura de los estados ON/OFF de las entradas discretas de un esclavo. En la Figura 59 se observa la sección del panel frontal correspondiente a la función 02.

Figura 59. Sección del panel frontal correspondiente a la función 02 del protocolo Modbus

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica la dirección en memoria (**dirección de inicio**) donde se almacena el estado de la primera entrada y el **número de entradas** consecutivas que se quieren leer.

El direccionamiento de las entradas en los PLC's KOYO y TRILOGI es el siguiente:

KOYO tiene un módulo que contienen 16 entradas y su direccionamiento (en decimal) es:

X0	X1	X2	X3	X4	X5	X6	X7	X10	X11	X12	X13	X14	X15	X16	X17
2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063

TRILOGI contienen un sistema integrado de 16 entradas. La entrada 0 tiene la dirección Nº 1 en el mapa de memoria (ver Tabla 17); dado que el posicionamiento empieza en 1 según el protocolo Modbus, se debe restar uno a la dirección, obteniendo que la dirección Modbus de la entrada 0 es la Nº 0 d y de la entrada 15 es la Nº 15 d.

I0	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

En la Figura 60 se presenta el formato del mensaje consulta para la función 02, se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

Figura 60. Formato del Mensaje Consulta Función 02

	Dirección del esclavo	Función	Dirección de inicio	Nº de Salidas	CRC/LRC		
T1-T2-T3-T4	1 byte	02 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 32 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

Formato del Mensaje Respuesta

En el mensaje respuesta, el estado de cada entrada corresponde a un bit del campo de información. El estado 1 corresponde a ON y el estado 0 a un OFF, el bit menos significativo del primer byte del campo de información contiene la entrada direccionada en la consulta (**dirección de inicio).**

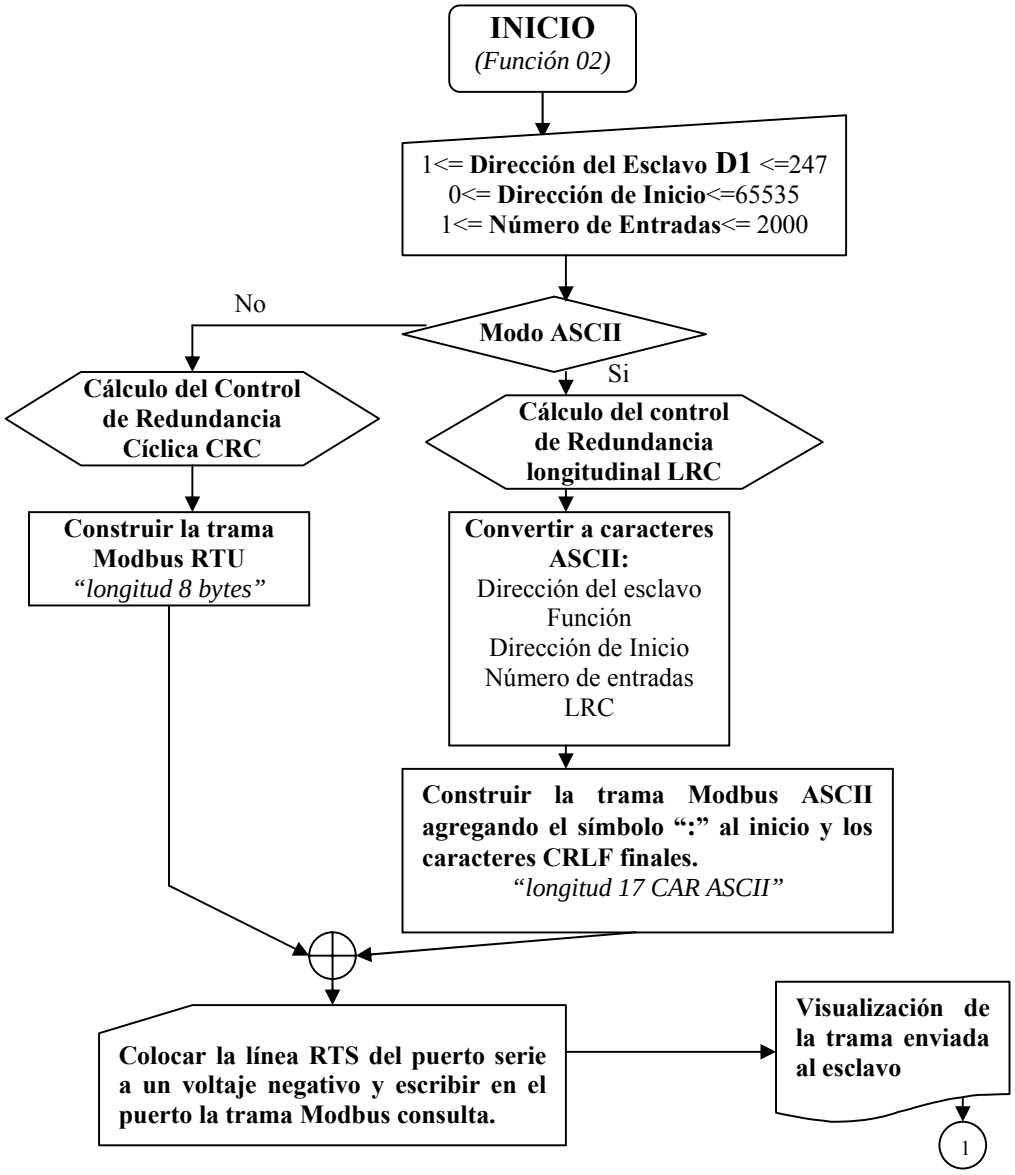
En la Figura 61 se presenta el formato del mensaje respuesta en los dos modos de transmisión y el tamaño de cada campo del mensaje.

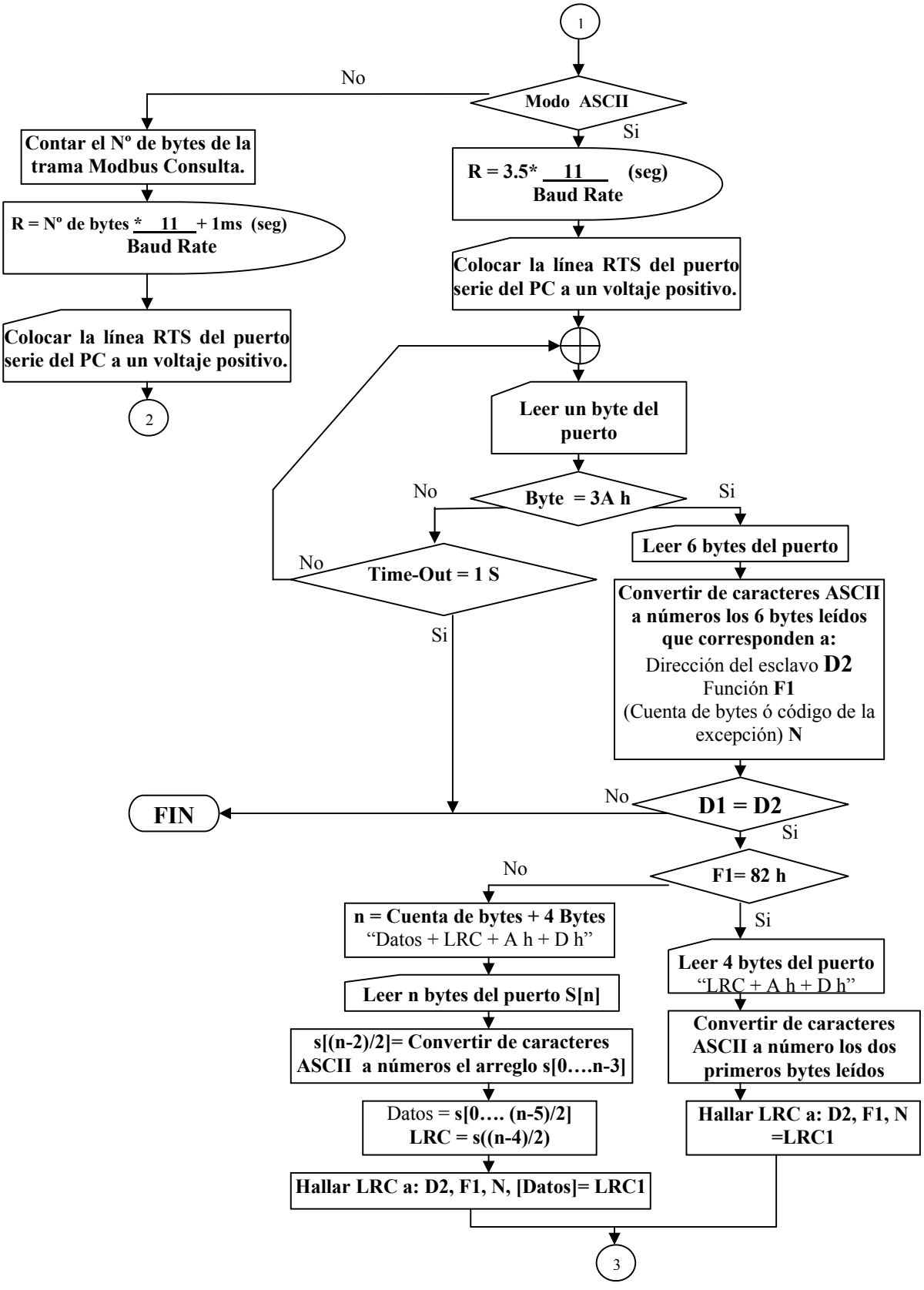
Figura 61. Formato del Mensaje Respuesta Función 02

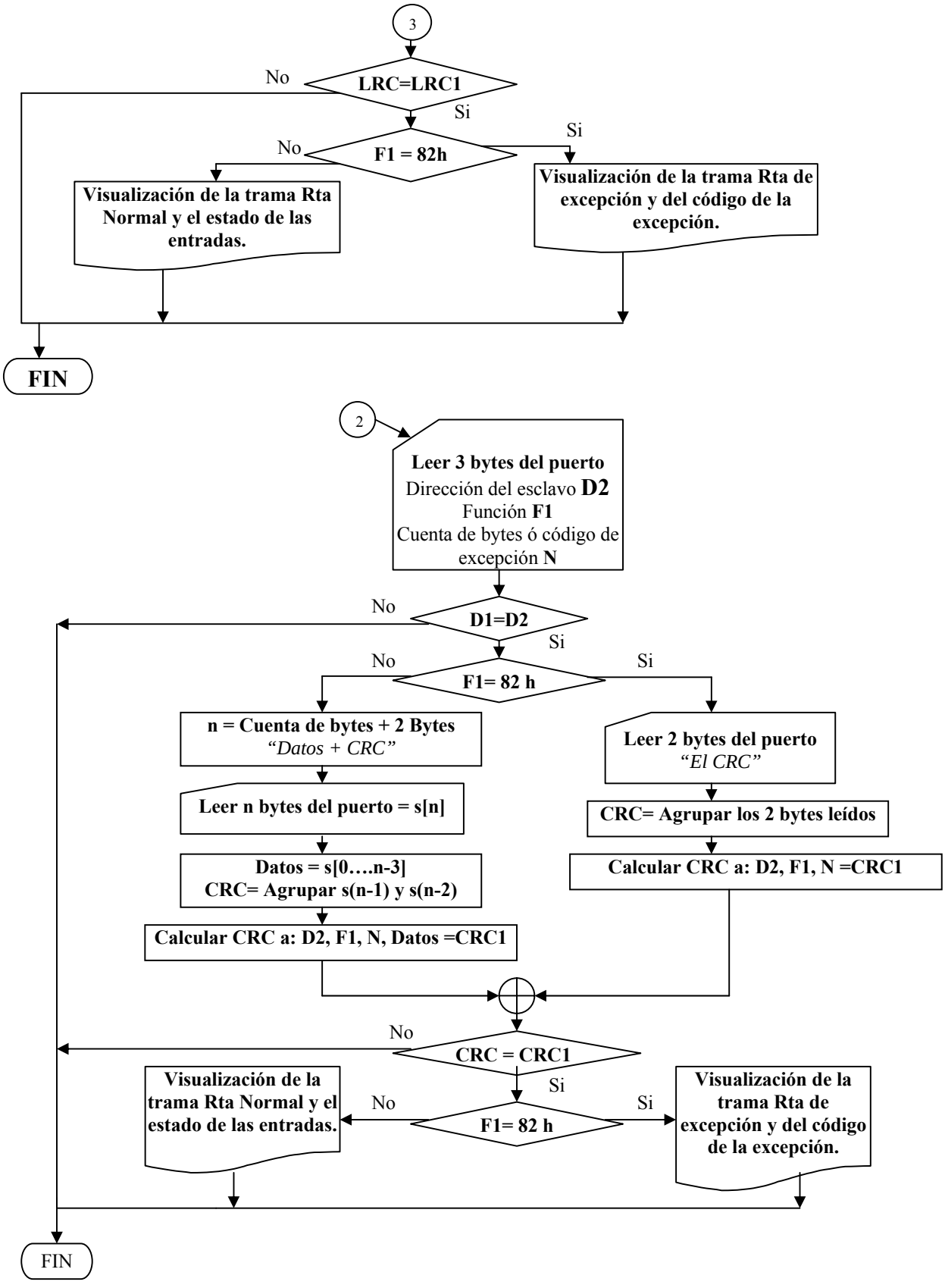
	Dirección del esclavo	Función	Campo de Información: Cuenta de Bytes N	Estados de las Entradas	CRC/LRC		
T1-T2-T3-T4	1 byte	02 h	1 byte	N byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 32 h	2 CAR ASCII	2*N CAR ASCII	2 CAR ASCII	D h A h	ASCII

En la Figura 62 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 02.

Figura 62. Diagrama de flujo correspondiente a la función 02




1
No
Modo ASCII
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = 3.5* 11 (seg)
Baud Rate
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
Leer un byte del puerto
No
Byte = 3A h
Si
Leer 6 bytes del puerto
No
Time-Out = 1 S
Convertir de caracteres ASCII a números los 6 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
(Cuenta de bytes ó código de la excepción) N
Si
FIN
No
D1 = D2
Si
No
F1= 82 h
Si
n = Cuenta de bytes + 4 Bytes
"Datos + LRC + A h + D h"
Leer 4 bytes del puerto
"LRC + A h + D h"
Leer n bytes del puerto S[n]
Convertir de caracteres ASCII a número los dos primeros bytes leídos
s[(n-2)/2]= Convertir de caracteres ASCII a números el arreglo s[0….n-3]
Datos = s[0…. (n-5)/2]
LRC = s((n-4)/2)
Hallar LRC a: D2, F1, N =LRC1
Hallar LRC a: D2, F1, N, [Datos]= LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 82h
Si
Visualización de la trama Rta Normal y el estado de las entradas.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN
2
Leer 3 bytes del puerto
Dirección del esclavo D2
Función F1
Cuenta de bytes ó código de excepción N
No
D1=D2
Si
No
F1= 82 h
Si
n = Cuenta de bytes + 2 Bytes
"Datos + CRC"
Leer n bytes del puerto = s[n]
Datos = s[0….n-3]
CRC= Agrupar s(n-1) y s(n-2)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 2 bytes del puerto
"El CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, N =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal y el estado de las entradas.
No
F1= 82 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.4 FUNCIÓN 03: LEER REGISTROS INTERNOS

La función 03 del protocolo Modbus permite la lectura de los registros internos en un esclavo. En la Figura 63 se observa la sección del panel frontal correspondiente a la función 03.

Figura 63. Sección del panel frontal correspondiente a la función 03 del protocolo Modbus

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica el registro inicial (**dirección de inicio**) y la cantidad de registros consecutivos a leer.

El direccionamiento de las entradas en los PLC's KOYO y TRILOGI es el siguiente:
KOYO: Al registro interno ubicado en la posición de memoria V1400 (octal) le corresponde la dirección Modbus 768 d, y así sucesivamente.

Dirección Modbus	**V1400**	**……**	**V7377**	**V10000**	**……**	**V17777**	**Palabras de datos (Octal)**
	768 d	……	3839 d	4096 d	……	8191 d	

TRILOGI: Al registro 41001 de la memoria de datos le corresponde la dirección Modbus 1000; para hallar la dirección Modbus se le resta, a la dirección del registro que se encuentra en la Tabla 17, el número 40001.

Ejemplo: 41001 - **40001** = 1000
40129 – **40001** = 128

En la Figura 64 se presenta el formato del mensaje consulta para la función 03, se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

Figura 64. Formato del Mensaje Consulta Función 03

	Dirección del esclavo	Función	Dirección de inicio	Nº de Registros	CRC/LRC		
T1-T2-T3-T4	**1 byte**	**03 h**	**2 byte**	**2 byte**	**2byte**	T1-T2-T3-T4	**RTU**
3A h	**2 CAR ASCII**	**30 h 33 h**	**4 CAR ASCII**	**4 CAR ASCII**	**2 CAR ASCII**	**D h A h**	**ASCII**

Formato del Mensaje Respuesta

En el mensaje respuesta, la información de los registros se empaqueta en dos bytes por registro; el primer byte contiene los bits más significativos y el segundo byte contiene los bits menos significativos.

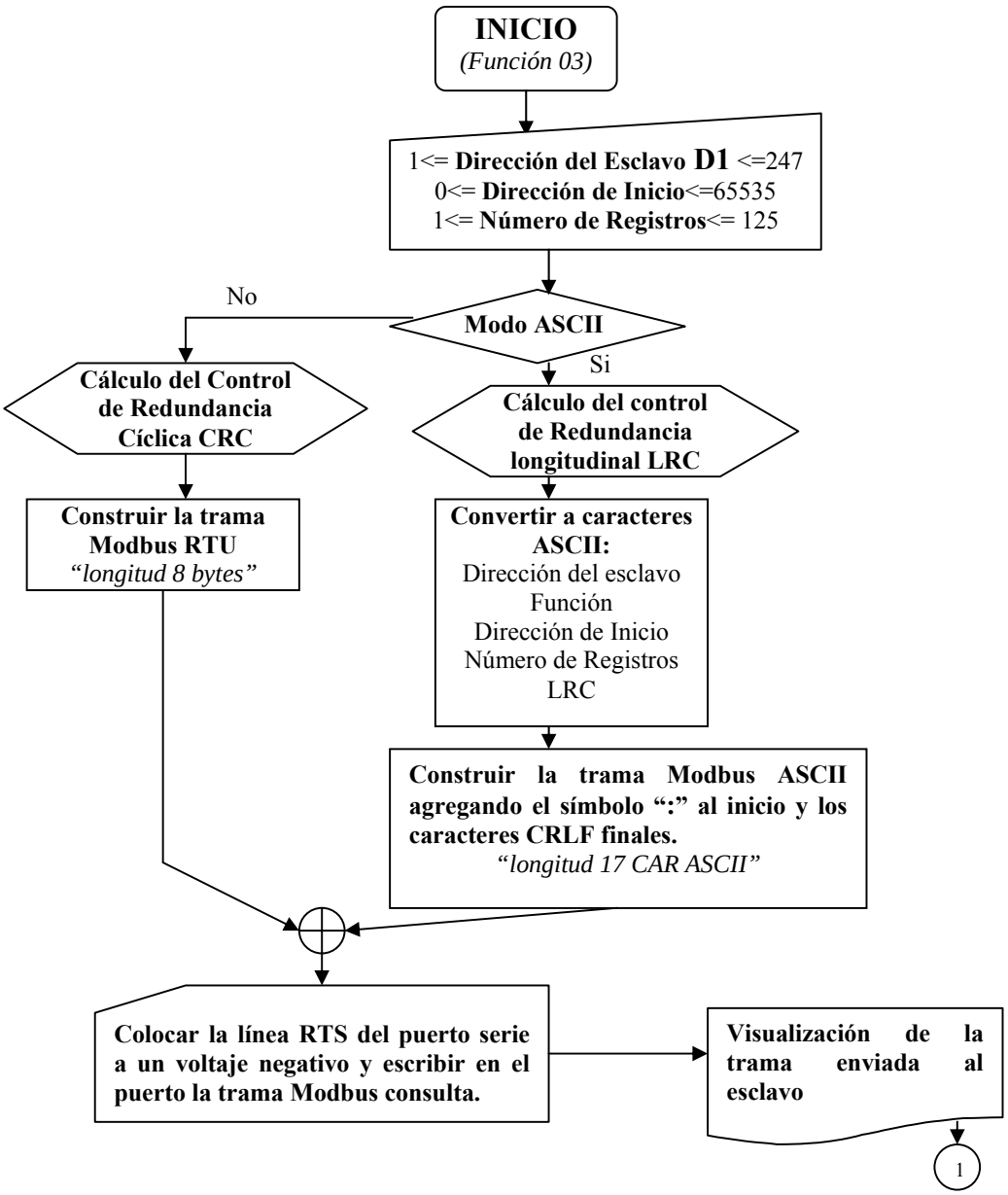
En la Figura 65 se presenta el formato del mensaje respuesta en los dos modos de transmisión y el tamaño de cada campo del mensaje.

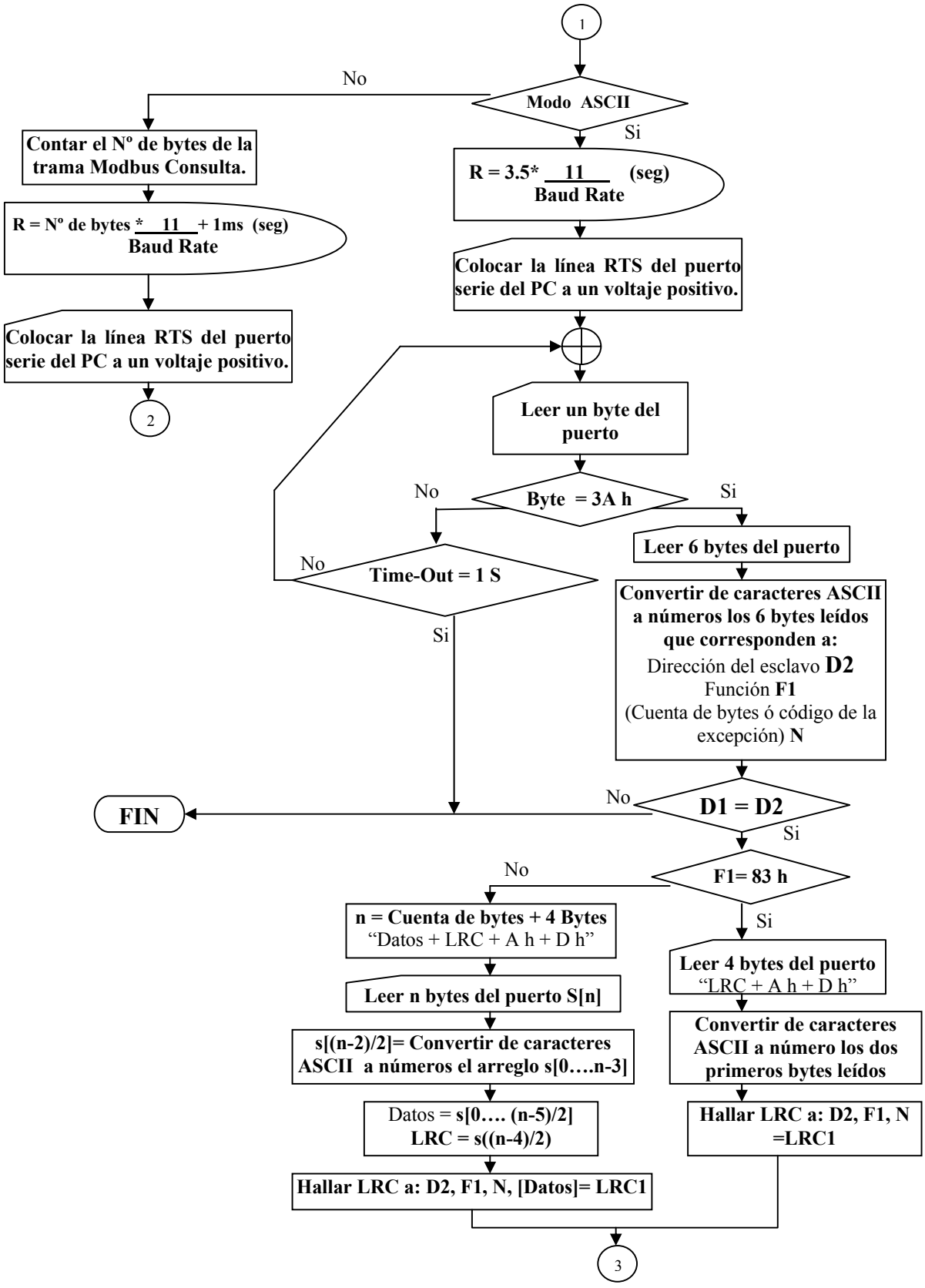
Figura 65. Formato del Mensaje Respuesta Función 03

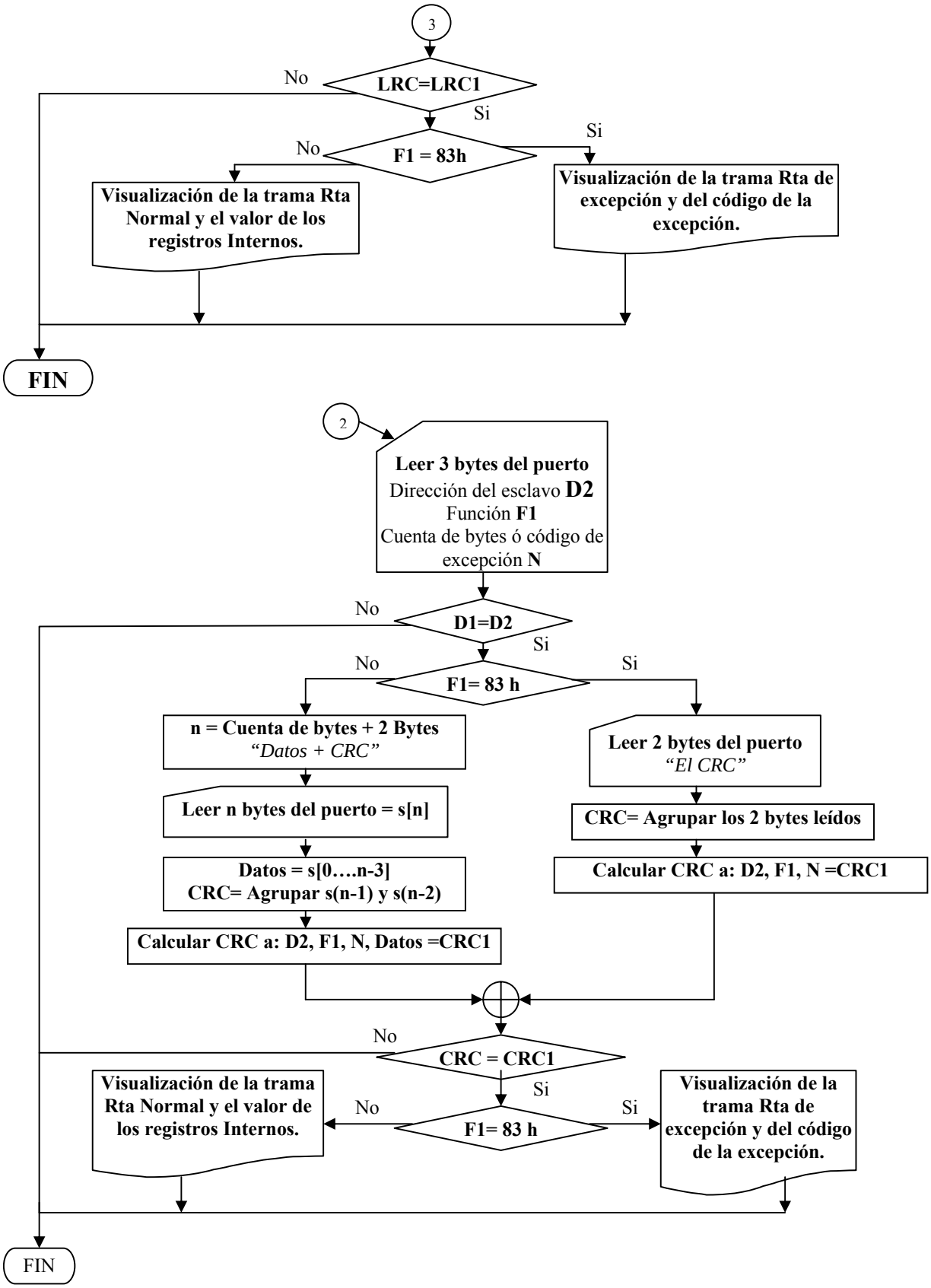
			Campo de Información				
	Dirección del esclavo	Función	Cuenta de Bytes N	Valor de los Registros	CRC/LRC		
T1-T2-T3-T4	**1 byte**	**03 h**	**1 byte**	**N byte**	**2byte**	T1-T2-T3-T4	**RTU**
3A h	**2 CAR ASCII**	**30 h 33 h**	**2 CAR ASCII**	**2*N CAR ASCII**	**2 CAR ASCII**	**D h A h**	**ASCII**

En la Figura. 66 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 03.

Figura 66. Diagrama de flujo correspondiente a la Función 03.




1
Modo ASCII
No
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
R = 3.5* 11 (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Leer un byte del puerto
Byte = 3A h
No
Si
Time-Out = 1 S
No
Si
Leer 6 bytes del puerto
Convertir de caracteres ASCII a números los 6 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
(Cuenta de bytes ó código de la excepción) N
FIN
D1 = D2
No
Si
F1= 83 h
No
Si
n = Cuenta de bytes + 4 Bytes
"Datos + LRC + A h + D h"
Leer n bytes del puerto S[n]
s[(n-2)/2]= Convertir de caracteres ASCII a números el arreglo s[0….n-3]
Datos = s[0…. (n-5)/2]
LRC = s((n-4)/2)
Hallar LRC a: D2, F1, N, [Datos]= LRC1
Leer 4 bytes del puerto
"LRC + A h + D h"
Convertir de caracteres ASCII a número los dos primeros bytes leídos
Hallar LRC a: D2, F1, N =LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 83h
Si
Visualización de la trama Rta Normal y el valor de los registros Internos.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN
2
Leer 3 bytes del puerto
Dirección del esclavo D2
Función F1
Cuenta de bytes ó código de excepción N
No
D1=D2
Si
No
F1= 83 h
Si
n = Cuenta de bytes + 2 Bytes
"Datos + CRC"
Leer n bytes del puerto = s[n]
Datos = s[0….n-3]
CRC= Agrupar s(n-1) y s(n-2)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 2 bytes del puerto
"El CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, N =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal y el valor de los registros Internos.
No
F1= 83 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.5 FUNCIÓN 04: LEER REGISTROS DE ENTRADA

La función 04 del protocolo Modbus permite la lectura de los registros de entrada en un esclavo. En la Figura 67 se observa la sección del panel frontal correspondiente a la función 04.

Figura 67. Sección del panel frontal correspondiente a la función 04 del protocolo Modbus

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica el registro inicial (**dirección de inicio**) y la cantidad de registros a leer.

En la Figura 68 se presenta el formato del mensaje consulta para la función 04 y se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

Figura 68. Formato del Mensaje Consulta Función 04

	Dirección del esclavo	Función	Dirección de inicio	Nº de Registros	CRC/LRC		
T1-T2-T3-T4	1 byte	04 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 34 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

Formato del Mensaje Respuesta

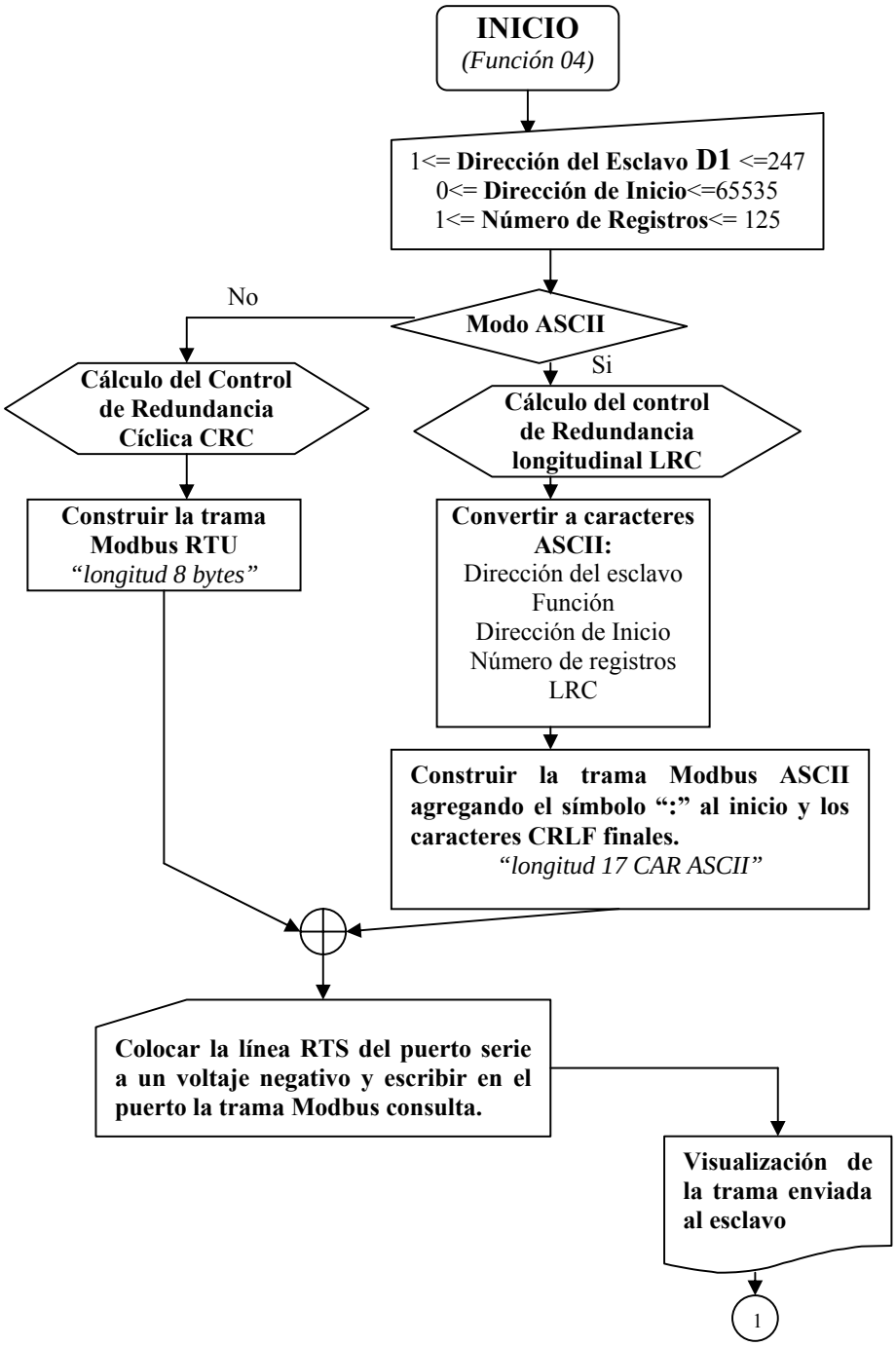
En el mensaje respuesta, la información de los registros se empaqueta en dos bytes por registro; el primer byte contiene los bits más significativos y el segundo byte contiene los bits menos significativos. En la Figura 69 se presenta el formato del mensaje respuesta en los dos modos de transmisión y el tamaño de cada campo del mensaje.

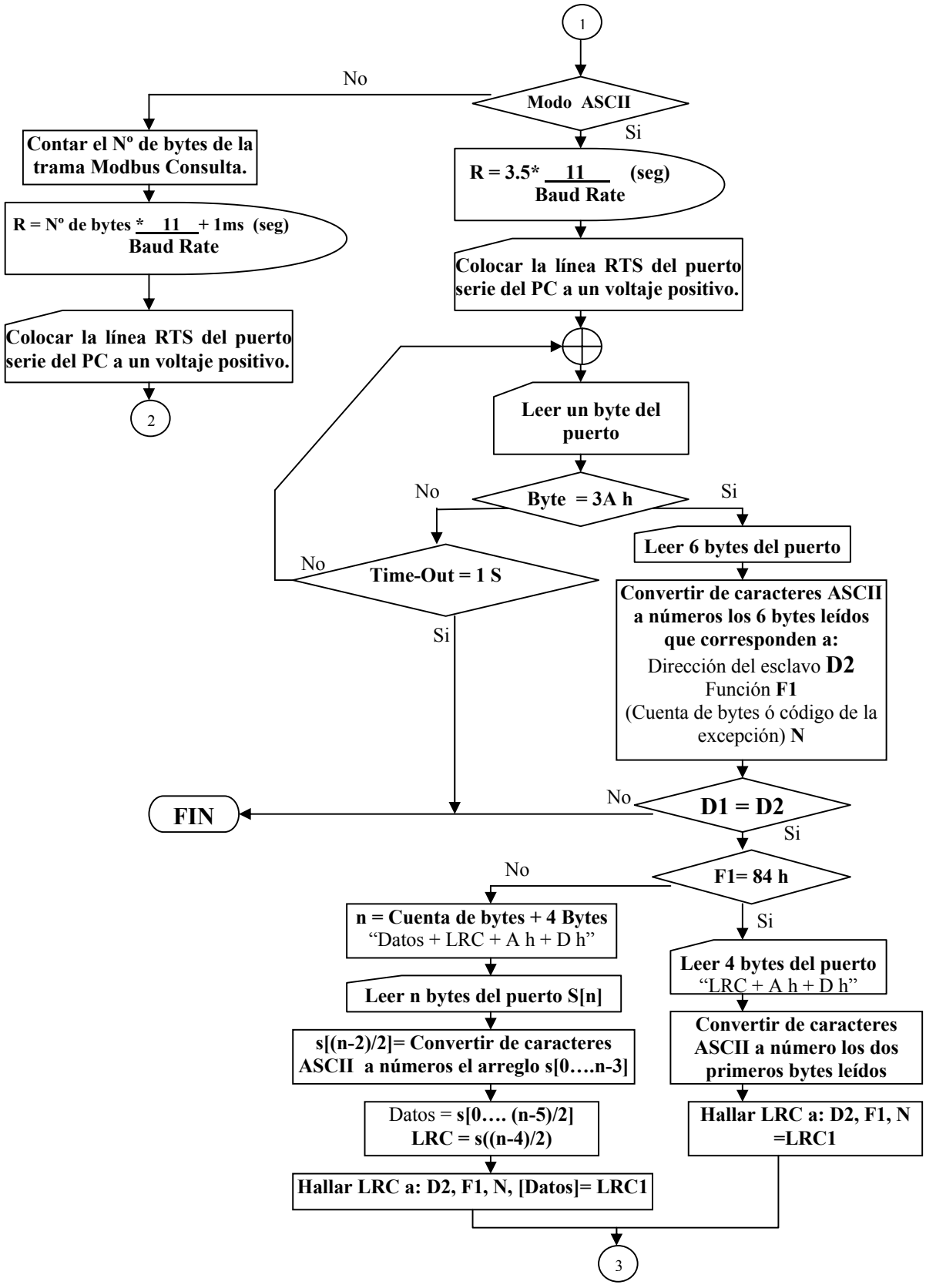
Figura 69. Formato del Mensaje Respuesta Función 04

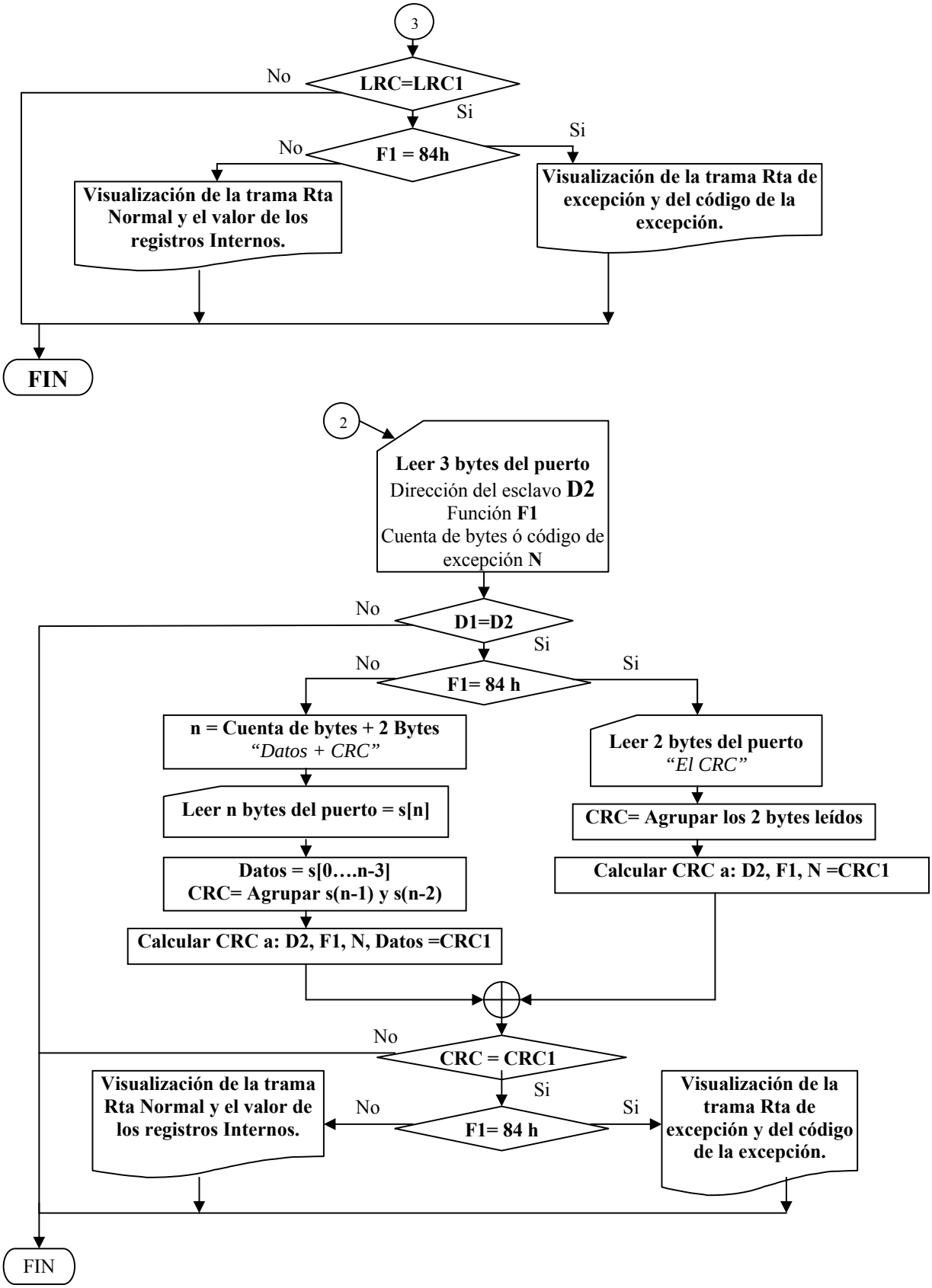
			Campo de Información				
	Dirección del esclavo	Función	Cuenta de Bytes N	Valor de los Registros	CRC/LRC		
T1-T2-T3-T4	1 byte	04 h	1 byte	N byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 34 h	2 CAR ASCII	2*N CAR ASCII	2 CAR ASCII	D h A h	ASCII

En la Figura 70 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 04.

Figura 70. Diagrama de flujo correspondiente a la función 04




1
No
Modo ASCII
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = 3.5* 11 (seg)
Baud Rate
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
Leer un byte del puerto
No
Byte = 3A h
Si
No
Time-Out = 1 S
Leer 6 bytes del puerto
Si
Convertir de caracteres ASCII a números los 6 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
(Cuenta de bytes ó código de la excepción) N
FIN
No
D1 = D2
Si
No
F1= 84 h
Si
n = Cuenta de bytes + 4 Bytes
"Datos + LRC + A h + D h"
Leer 4 bytes del puerto
"LRC + A h + D h"
Leer n bytes del puerto S[n]
Convertir de caracteres ASCII a número los dos primeros bytes leídos
s[(n-2)/2]= Convertir de caracteres ASCII a números el arreglo s[0….n-3]
Datos = s[0…. (n-5)/2]
LRC = s((n-4)/2)
Hallar LRC a: D2, F1, N =LRC1
Hallar LRC a: D2, F1, N, [Datos]= LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 84h
Si
Visualización de la trama Rta Normal y el valor de los registros Internos.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN
2
Leer 3 bytes del puerto
Dirección del esclavo D2
Función F1
Cuenta de bytes ó código de excepción N
No
D1=D2
Si
No
F1= 84 h
Si
n = Cuenta de bytes + 2 Bytes
"Datos + CRC"
Leer n bytes del puerto = s[n]
Datos = s[0….n-3]
CRC= Agrupar s(n-1) y s(n-2)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 2 bytes del puerto
"El CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, N =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal y el valor de los registros Internos.
No
F1= 84 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.6 FUNCIÓN 05: FORZAR UNA "BOBINA"

La función 05 del protocolo Modbus forza el estado de una "bobina" (ON ó OFF). En la Figura 71 se observa la sección del panel frontal correspondiente a la función 05.

Figura 71. Sección del panel frontal correspondiente a la función 05 del protocolo Modbus

La bobina permanece en el estado forzado siempre y cuando no esté programada en la lógica del dispositivo; si la bobina está programada cambiará al estado que indique la lógica del dispositivo.

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica la dirección de la bobina (**dirección de inicio**) y el estado a forzar (**dato a forzar**) se consigue incluyendo alguna de las siguientes constantes en el mensaje establecidas en el protocolo Modbus:

- **FF00 h** para el estado **ON**
- **0000 h** para el estado **OFF**

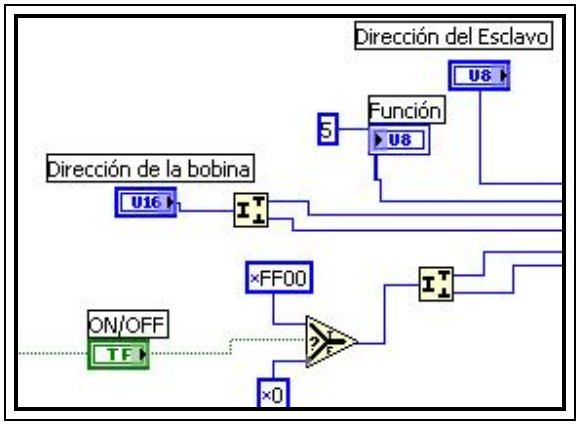
En la Figura 72 se presenta el formato del mensaje consulta para la función 05 y se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

Figura 72. Formato del Mensaje Consulta Función 05

	Dirección del esclavo	Función	Dirección de la bobina	Estado a Forzar (FF00 h ó 0000 h)	CRC/LRC		
T1-T2-T3-T4	1 byte	05 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 35 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

En la Figura 73 se presenta la construcción del mensaje consulta para la función 05 en LabVIEW, observe que dependiendo del Estado a forzar la bobina (T "1" ó F "0") se elige la constante que irá en el campo de información del mensaje consulta. El procedimiento es igual que el de la función 01, después de concatenar los datos se calcula el control de errores (LRC ó CRC) y se escribe la trama en el puerto.

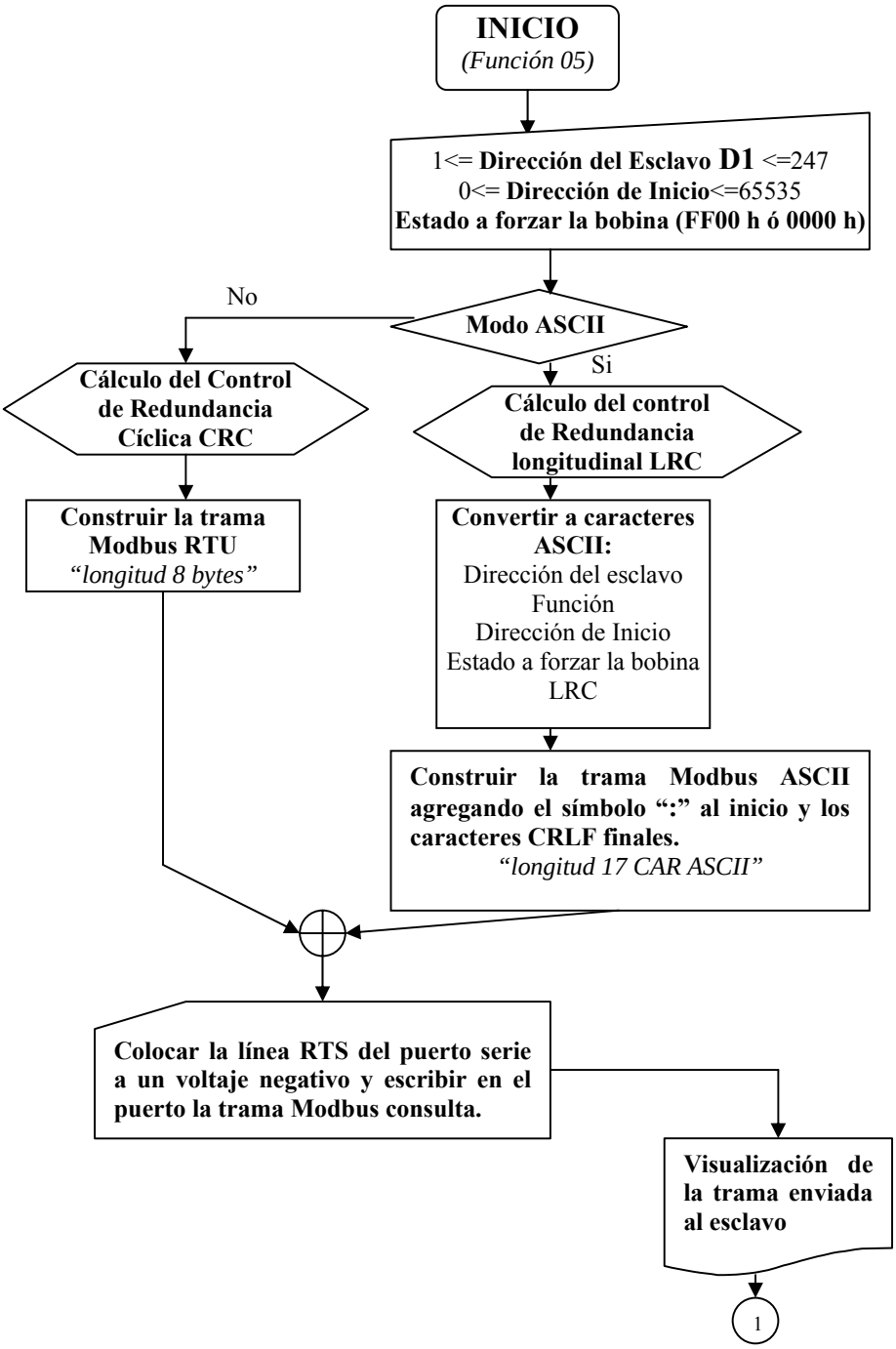
Figura 73. Construcción del mensaje consulta para la función 05

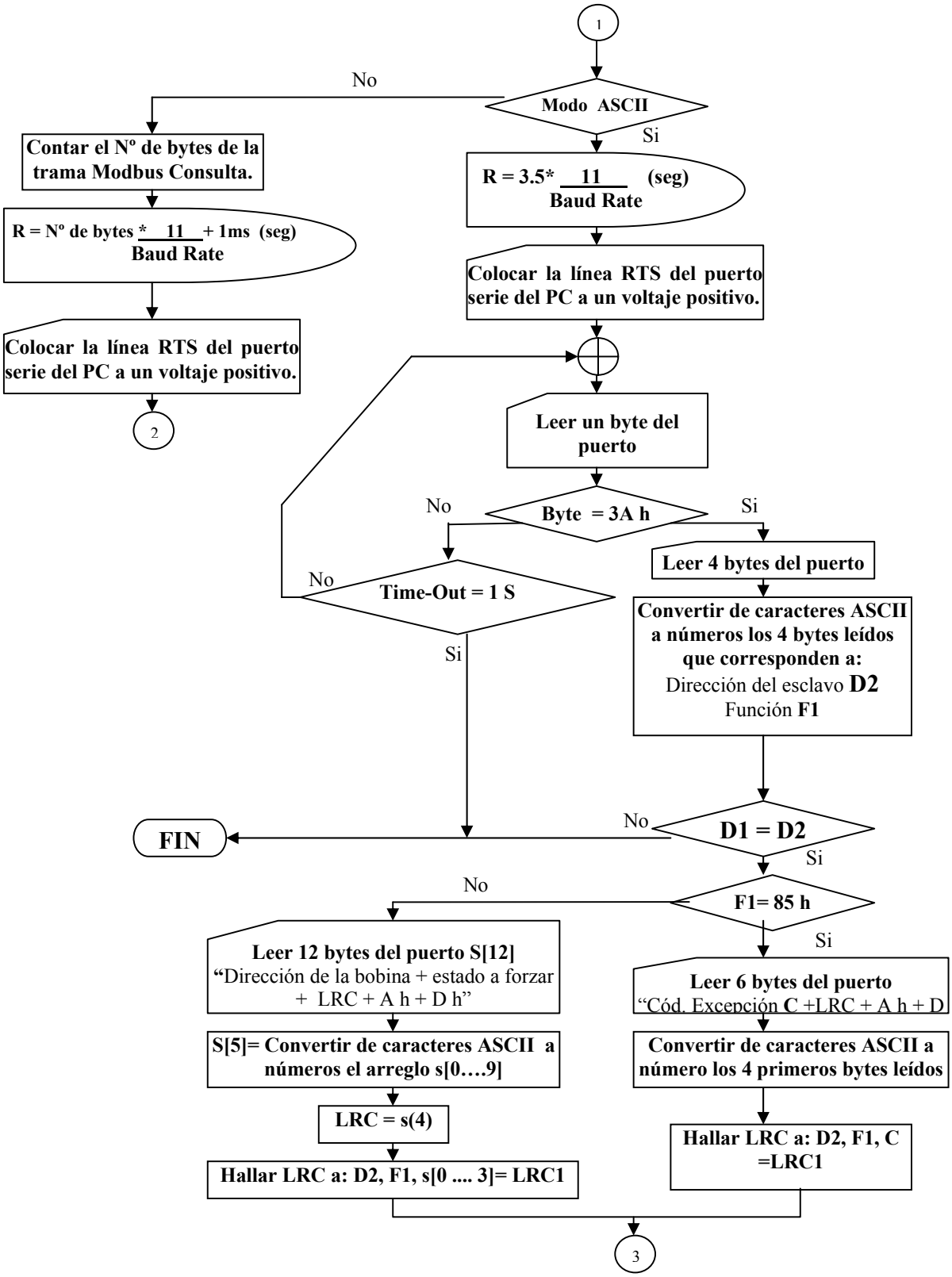


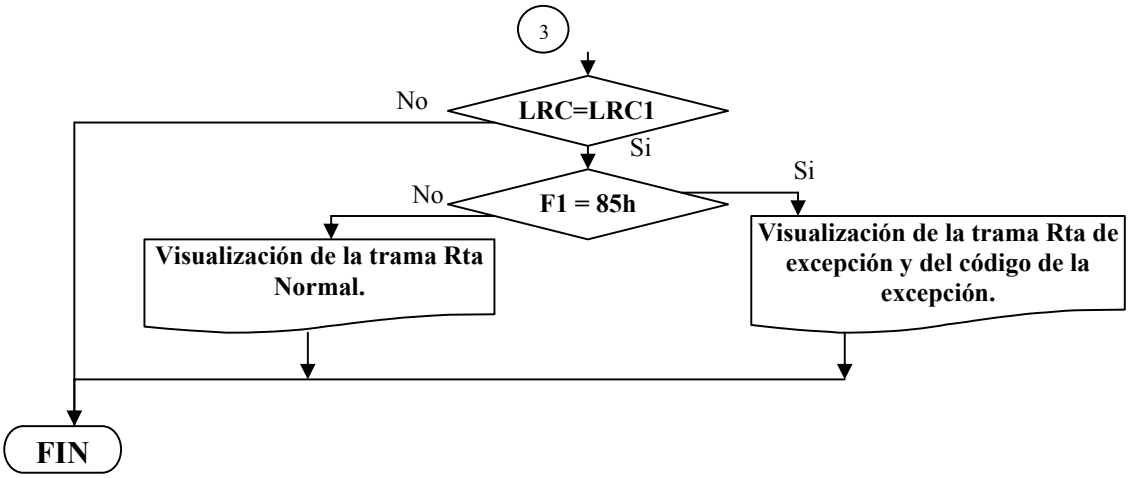
Cuando el esclavo responde (en el caso que no sea un mensaje de difusión) devuelve como respuesta una copia del mensaje consulta, ver Figura 71.

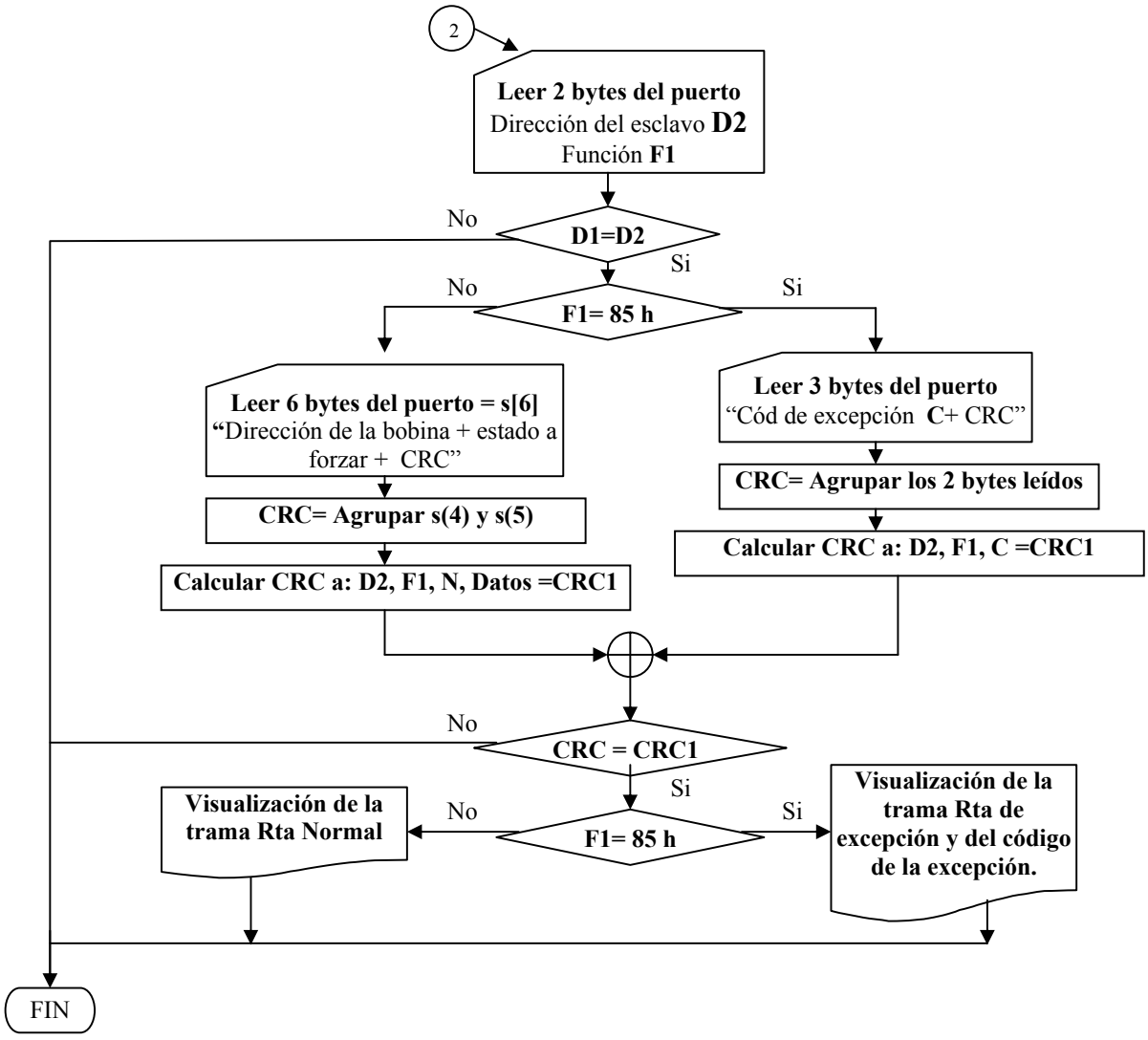
En la Figura 74 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 05.

Figura 74. Diagrama de flujo correspondiente a la función 05




1
Modo ASCII
No
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
R = 3.5* 11 (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Leer un byte del puerto
Byte = 3A h
No
Si
Time-Out = 1 S
No
Si
Leer 4 bytes del puerto
Convertir de caracteres ASCII a números los 4 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
FIN
D1 = D2
No
Si
F1= 85 h
No
Si
Leer 12 bytes del puerto S[12]
"Dirección de la bobina + estado a forzar + LRC + A h + D h"
S[5]= Convertir de caracteres ASCII a números el arreglo s[0….9]
LRC = s(4)
Hallar LRC a: D2, F1, s[0 .... 3]= LRC1
Leer 6 bytes del puerto
"Cód. Excepción C +LRC + A h + D
Convertir de caracteres ASCII a número los 4 primeros bytes leídos
Hallar LRC a: D2, F1, C =LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 85h
Si
Visualización de la trama Rta Normal.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN


2
Leer 2 bytes del puerto
Dirección del esclavo D2
Función F1
No
D1=D2
Si
No
F1= 85 h
Si
Leer 6 bytes del puerto = s[6]
"Dirección de la bobina + estado a forzar + CRC"
CRC= Agrupar s(4) y s(5)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 3 bytes del puerto
"Cód de excepción C+ CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, C =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal
No
F1= 85 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.7 FUNCIÓN 06: FIJAR UN VALOR EN UN REGISTRO

La función 06 del protocolo Modbus fija un valor en un registro interno. El registro permanece en el valor forzado siempre y cuando no esté programado en la lógica del dispositivo; si el registro está programado cambiará al valor que indique la lógica del dispositivo. En la Figura 75 se observa la sección del panel frontal correspondiente a la función 06.

Figura 75. Sección del panel frontal correspondiente a la función 06 del protocolo Modbus

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica la **Dirección del Registro** y el valor a fijar en el registro.

En la Figura 76 se presenta el formato del mensaje consulta para la función 06 y se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

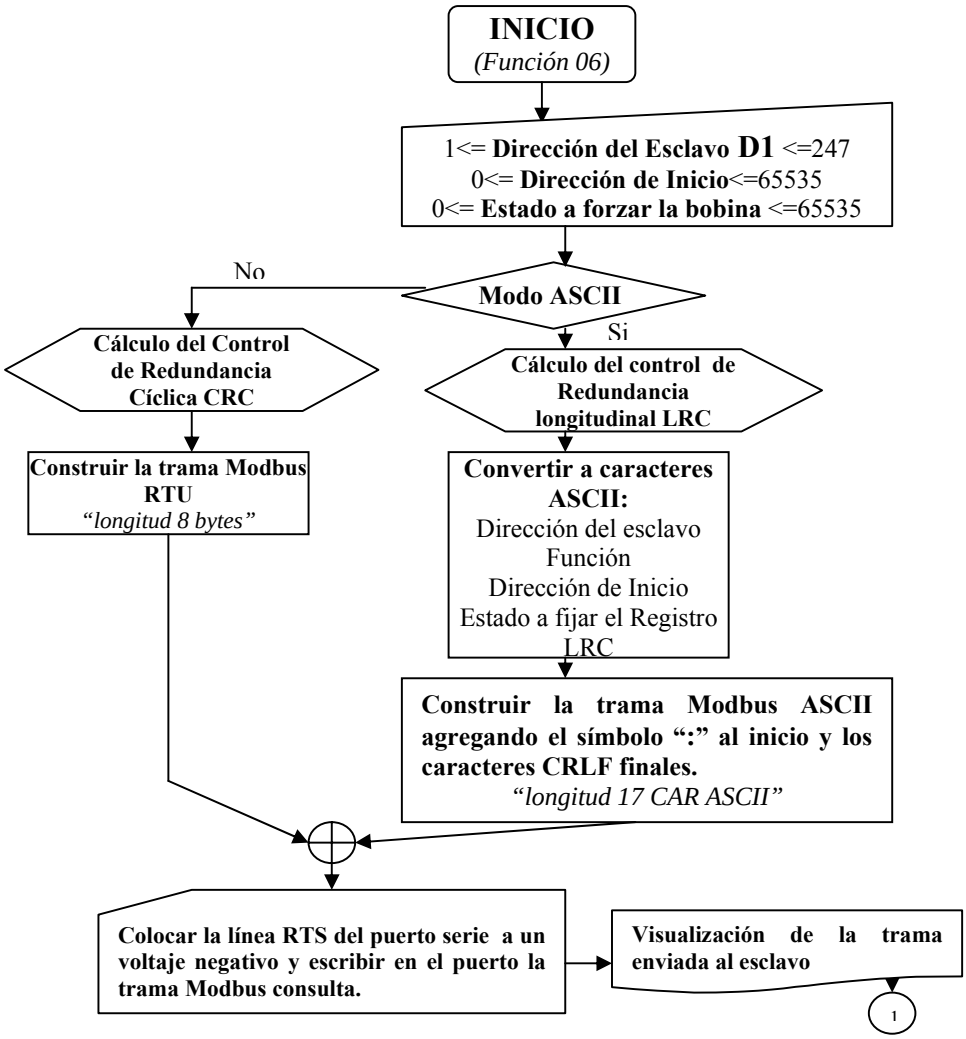
Figura 76. Formato del Mensaje Consulta Función 06

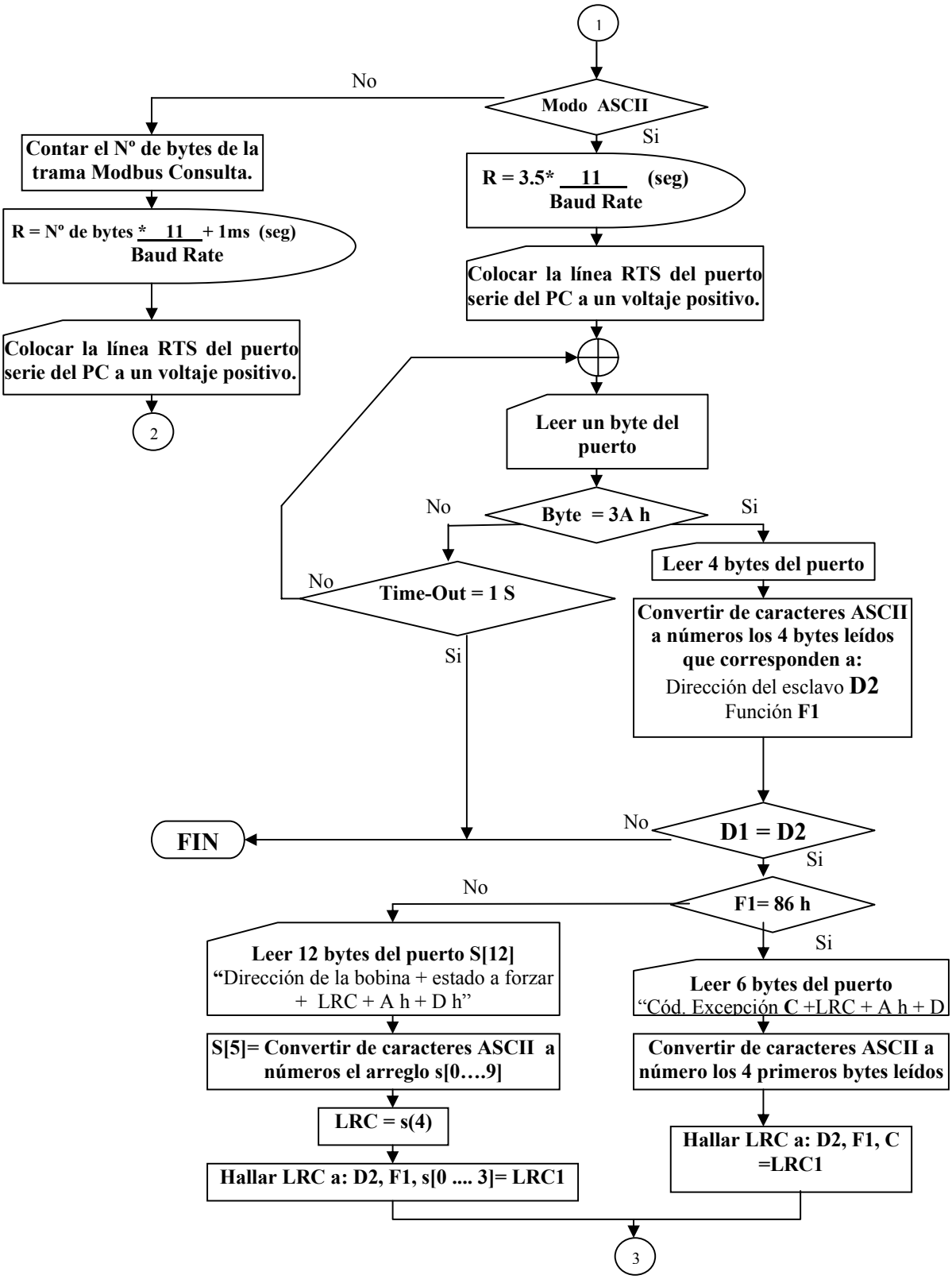
	Dirección del esclavo	Función	Dirección de la bobina	Dato a fijar en el registro	CRC/LRC		
T1-T2-T3-T4	1 byte	06 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 36 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

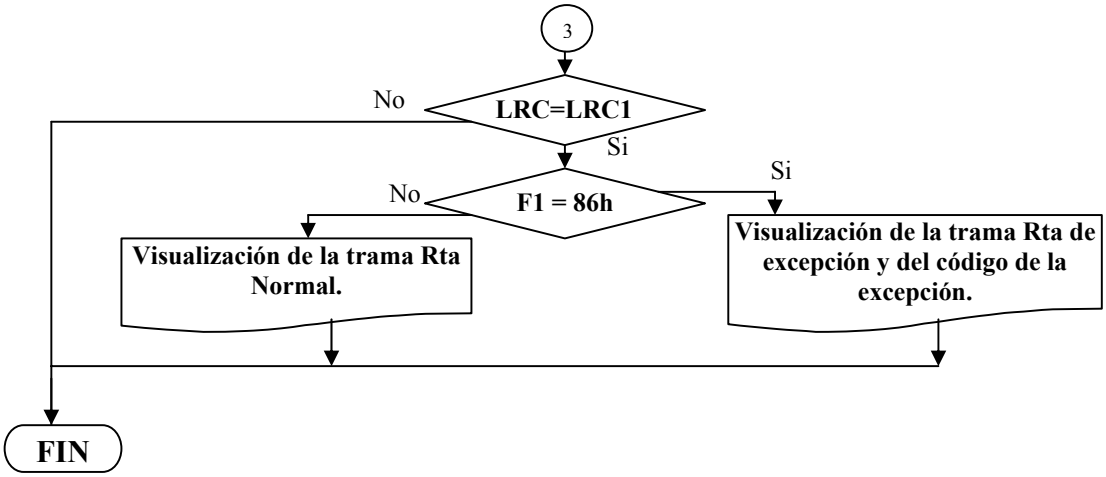
Cuando el esclavo responde devuelve como respuesta una copia del mensaje consulta.

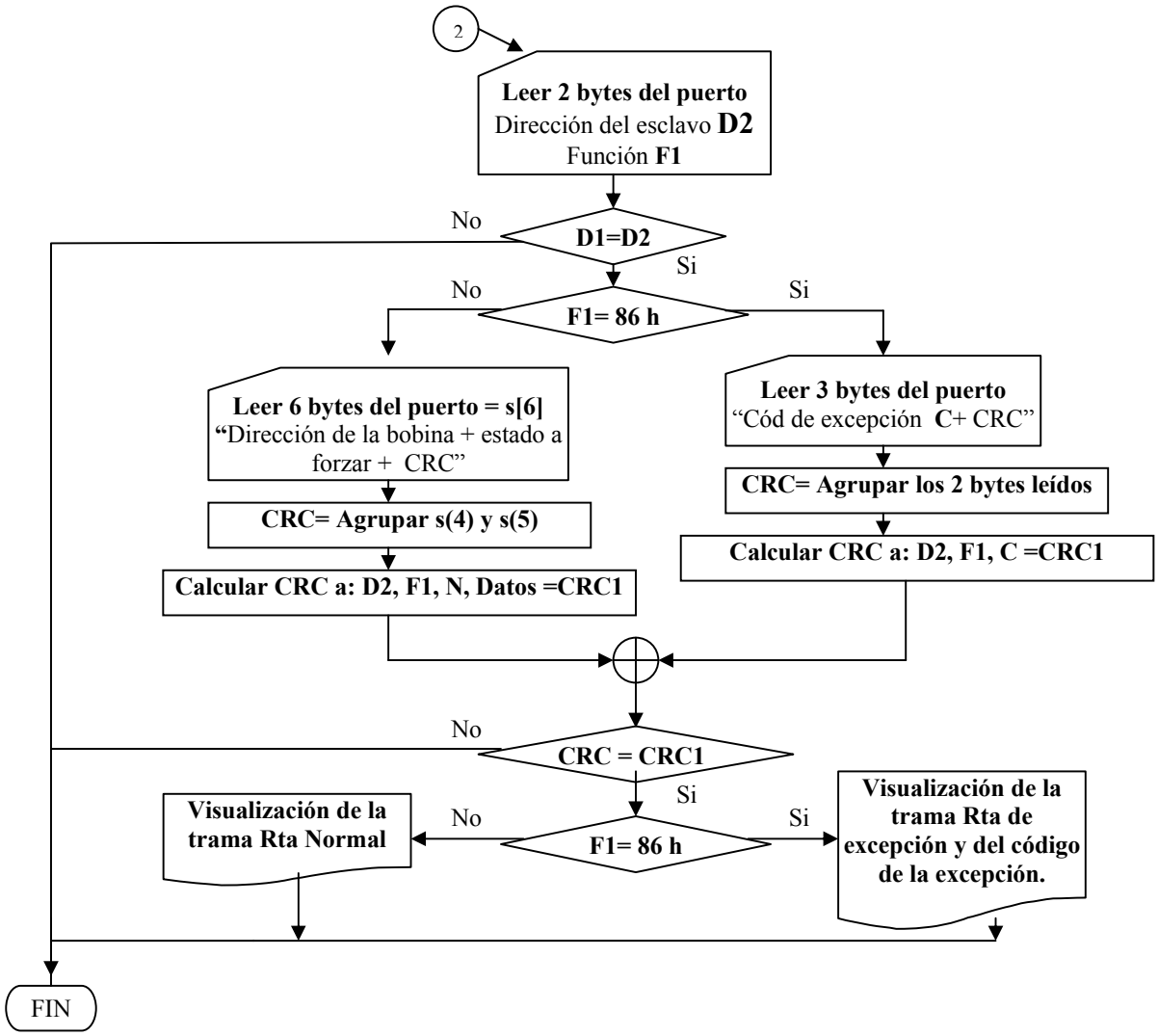
En la Figura 77 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 06.

Figura 77. Diagrama de flujo correspondiente a la función 06




1
Modo ASCII
No
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
R = 3.5* 11 (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Leer un byte del puerto
Byte = 3A h
No
Si
Time-Out = 1 S
No
Si
Leer 4 bytes del puerto
Convertir de caracteres ASCII a números los 4 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
FIN
D1 = D2
No
Si
F1= 86 h
No
Si
Leer 12 bytes del puerto S[12]
"Dirección de la bobina + estado a forzar + LRC + A h + D h"
S[5]= Convertir de caracteres ASCII a números el arreglo s[0….9]
LRC = s(4)
Hallar LRC a: D2, F1, s[0 .... 3]= LRC1
Leer 6 bytes del puerto
"Cód. Excepción C +LRC + A h + D
Convertir de caracteres ASCII a número los 4 primeros bytes leídos
Hallar LRC a: D2, F1, C =LRC1
3


3
LRC=LRC1
No
Si
F1 = 86h
No
Si
Visualización de la trama Rta Normal.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN


2
Leer 2 bytes del puerto
Dirección del esclavo D2
Función F1
D1=D2
No
Si
F1= 86 h
No
Si
Leer 6 bytes del puerto = s[6]
"Dirección de la bobina + estado a forzar + CRC"
CRC= Agrupar s(4) y s(5)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 3 bytes del puerto
"Cód de excepción C+ CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, C =CRC1
CRC = CRC1
No
Si
Visualización de la trama Rta Normal
F1= 86 h
No
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.8 FUNCIÓN 07: PETICIÓN DE LECTURA DE ESTADOS DE EXCEPCIÓN

Lee el contenido de ocho bobinas de Condición de Excepción en un esclavo. En la Figura 78 se observa la sección del panel frontal correspondiente a la función 07.

Figura 78. Sección del panel frontal correspondiente a la función 07 del protocolo Modbus

En la Figura 79 se presenta el formato del mensaje consulta para la función 07 para los modos RTU y ASCII.

Figura 79. Formato del Mensaje Consulta Función 07

	Dirección del esclavo	Función	CRC/LRC		
T1-T2-T3-T4	1 byte	07 h	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 37 h	2 CAR ASCII	D h A h	ASCII

Formato del Mensaje Respuesta

En el mensaje respuesta, la información de las 8 bobinas de Condición de excepción se empaqueta en un byte; una bobina (el estado ON/OFF) por bit.

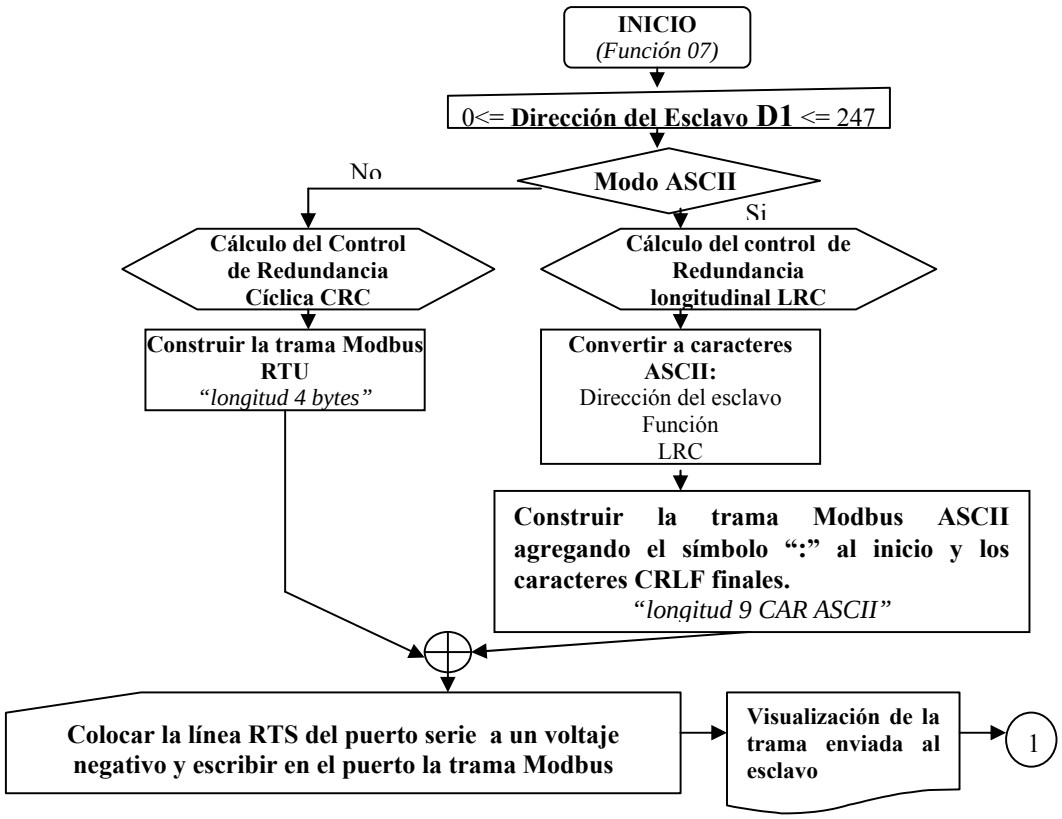
En la Figura 80 se presenta el formato del mensaje respuesta en los dos modos de transmisión.

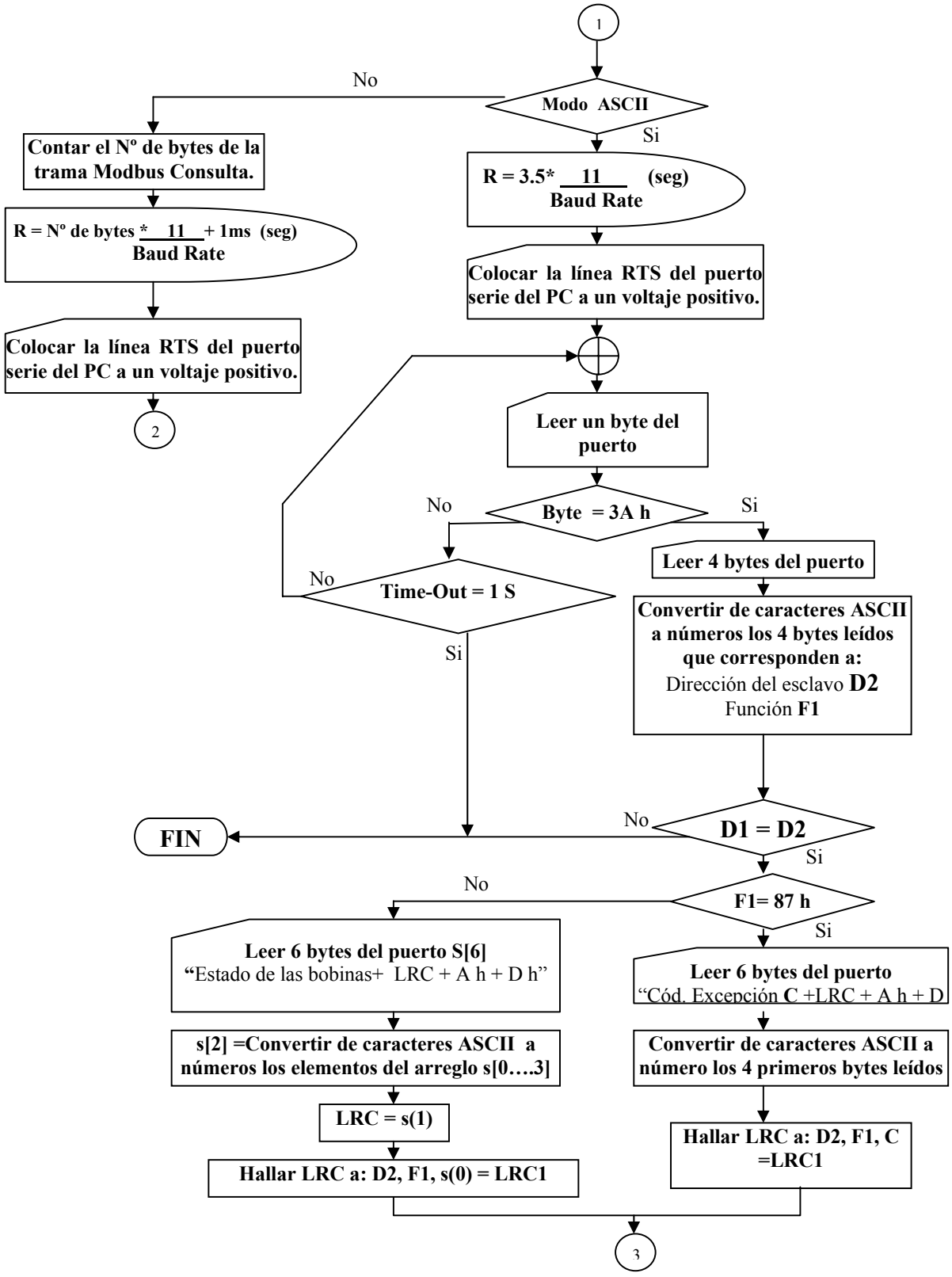
Figura 80. Formato del Mensaje Respuesta Función 07

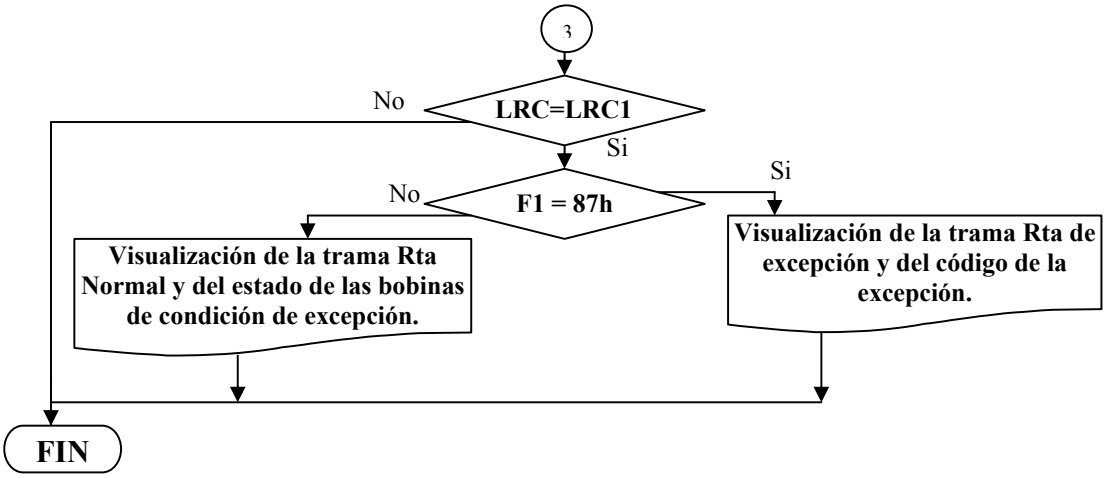
	Dirección del esclavo	Función	Datos de las bobinas	CRC/LRC		
T1-T2-T3-T4	1 byte	07 h	1 byte	2byte	T1-T2-T3-T4	RTU
3ª h	2 CAR ASCII	30 h 37 h	2 CAR ASCII	2 CAR ASCII	D h A h	ASCII

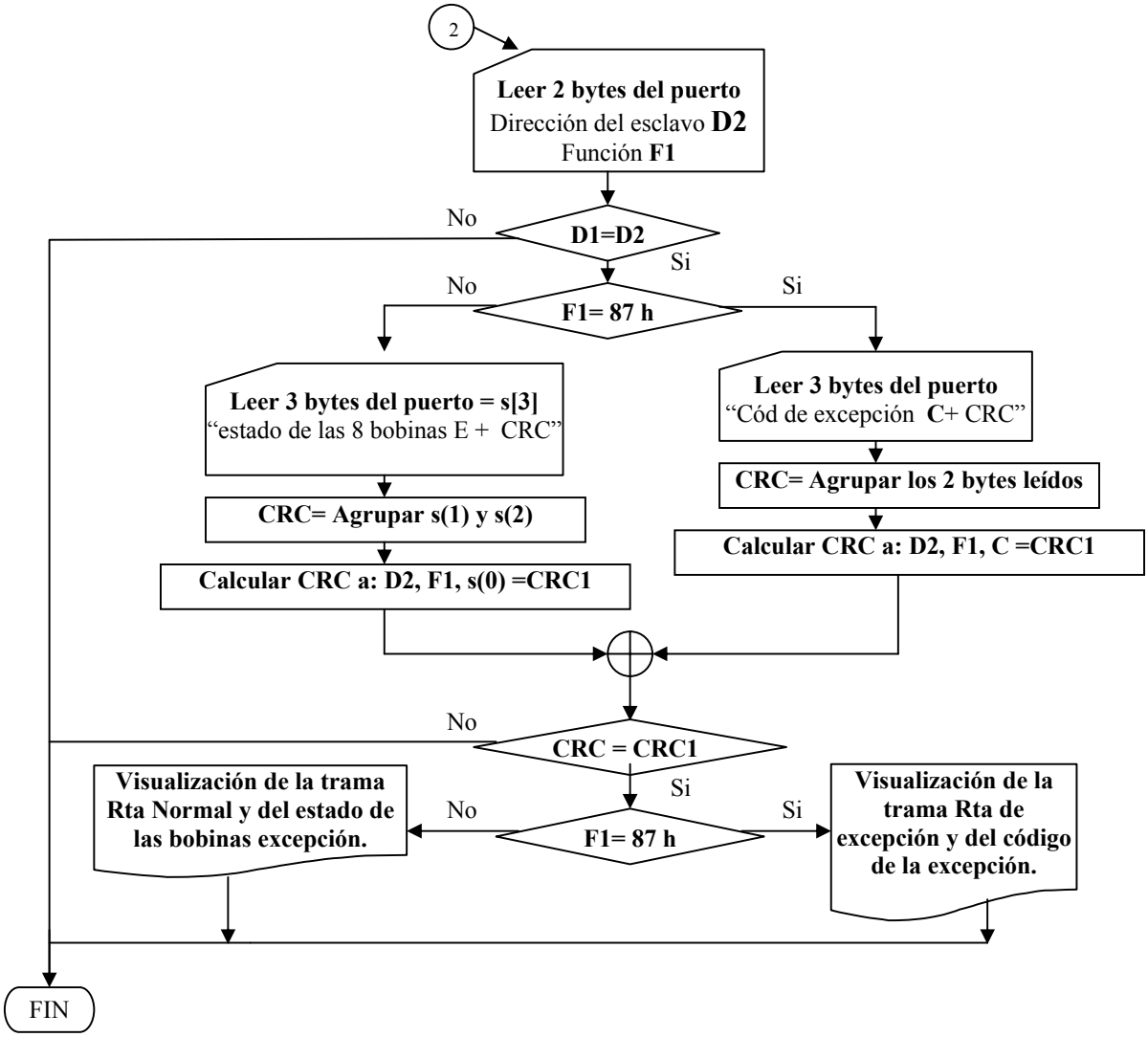
En la Figura 81 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 07.

Figura 81. Diagrama de flujo correspondiente a la función 07.




1
No
Modo ASCII
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = 3.5* 11 (seg)
Baud Rate
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
Leer un byte del puerto
No
Byte = 3A h
Si
No
Time-Out = 1 S
Si
Leer 4 bytes del puerto
Convertir de caracteres ASCII a números los 4 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
FIN
No
D1 = D2
Si
No
F1= 87 h
Si
Leer 6 bytes del puerto S[6]
"Estado de las bobinas+ LRC + A h + D h"
Leer 6 bytes del puerto
"Cód. Excepción C +LRC + A h + D
s[2] =Convertir de caracteres ASCII a números los elementos del arreglo s[0….3]
Convertir de caracteres ASCII a número los 4 primeros bytes leídos
LRC = s(1)
Hallar LRC a: D2, F1, s(0) = LRC1
Hallar LRC a: D2, F1, C =LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 87h
Si
Visualización de la trama Rta Normal y del estado de las bobinas de condición de excepción.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN


2
Leer 2 bytes del puerto
Dirección del esclavo D2
Función F1
No
D1=D2
Si
No
F1= 87 h
Si
Leer 3 bytes del puerto = s[3]
"estado de las 8 bobinas E + CRC"
CRC= Agrupar s(1) y s(2)
Calcular CRC a: D2, F1, s(0) =CRC1
Leer 3 bytes del puerto
"Cód de excepción C+ CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, C =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal y del estado de las bobinas excepción.
No
F1= 87 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

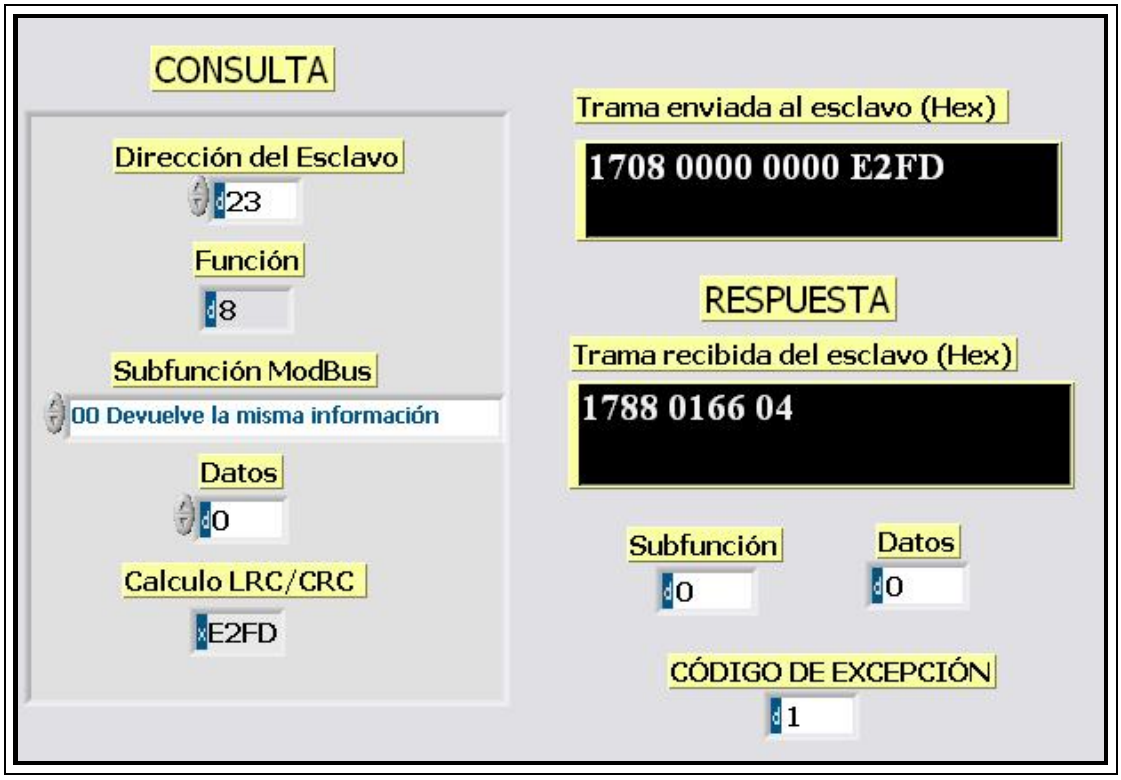
4.9 FUNCIÓN 08: DIAGNÓSTICO

La función 08 del protocolo Modbus permite:

- Comprobar el sistema de comunicación entre maestro y esclavo.
- Y saber estadisticas.

En el capítulo 3 (Tabla 15) se encuentran los códigos de las subfunciones de diagnóstico y la descripción de cada una de ellas. En la Figura 82 se observa la sección del panel frontal correspondiente a la función 08.

Figura 82. Sección del panel frontal correspondiente a la función 08 del protocolo Modbus



Formato del Mensaje Consulta

En el mensaje consulta se especifica el **código de subfunción** que define el tipo de prueba a realizar, el campo de información se utiliza para enviar información o control de diagnóstico al esclavo.

En la Figura 83 se presenta el formato del mensaje consulta para la función 08 y se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

Figura 83. Formato del Mensaje Consulta Función 08

			Campo de Información				
	Dirección del esclavo	Función	Subfunción	Dato	CRC/LRC		
T1-T2-T3-T4	1 byte	08 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 38 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

Formato del Mensaje Respuesta

En el mensaje respuesta se devuelve un dato que corresponde a la información pedida en el código de subfunción. En la Figura 84 se presenta el formato del mensaje respuesta en los dos modos de transmisión y el tamaño de cada campo del mensaje.

Figura 84. Formato del Mensaje Respuesta Función 08

			Campo de Información				
	Dirección del esclavo	Función	Subfunción	Dato	CRC/LRC		
T1-T2-T3-T4	1 byte	08 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 38 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

Observe que el formato del mensaje consulta es igual al formato del mensaje respuesta. En la Tabla 19 se define para cada subfunción de diagnóstico el campo de información del mensaje consulta y del mensaje respuesta.

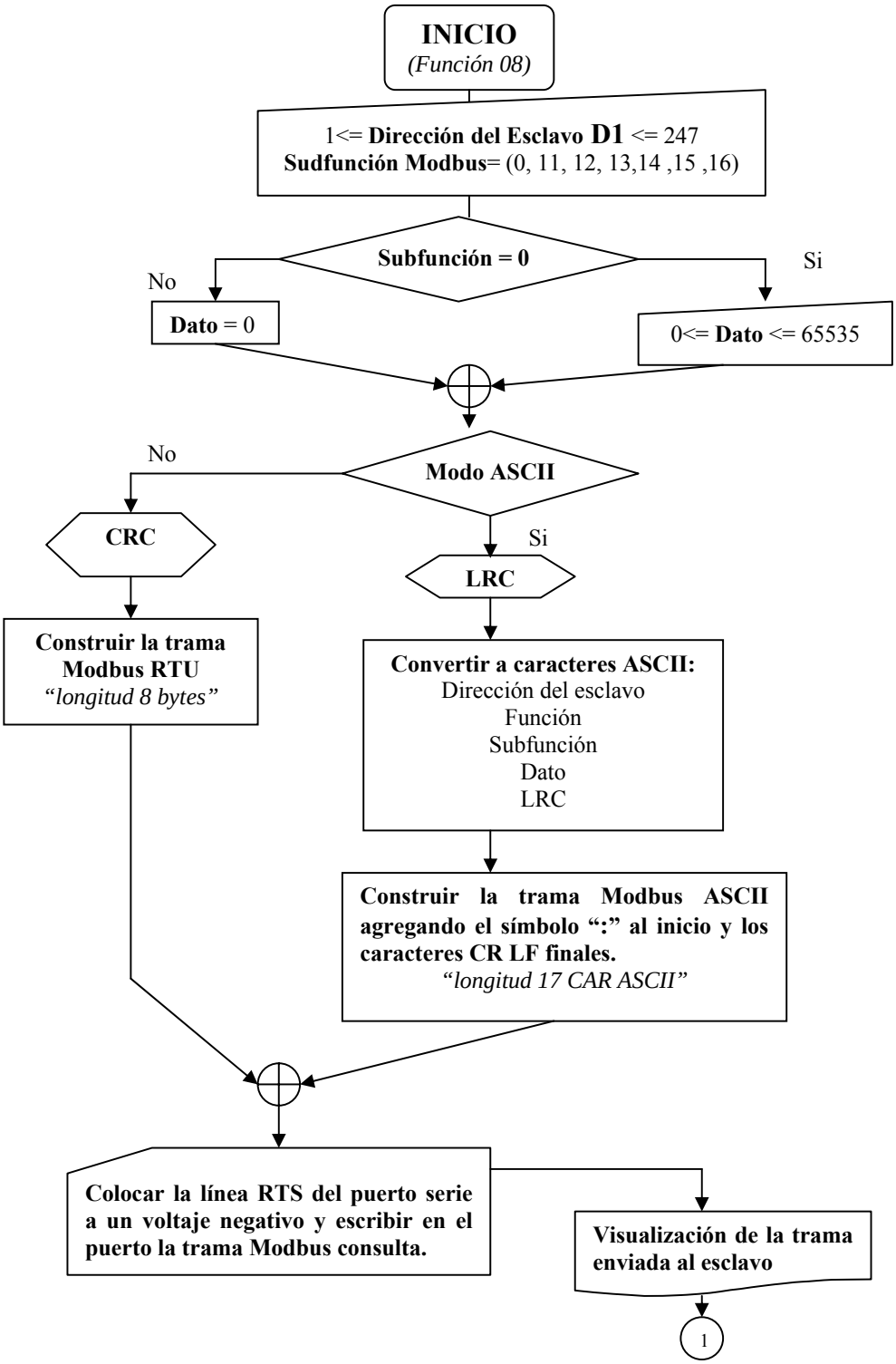
Tabla 19.Códigos de subfunción

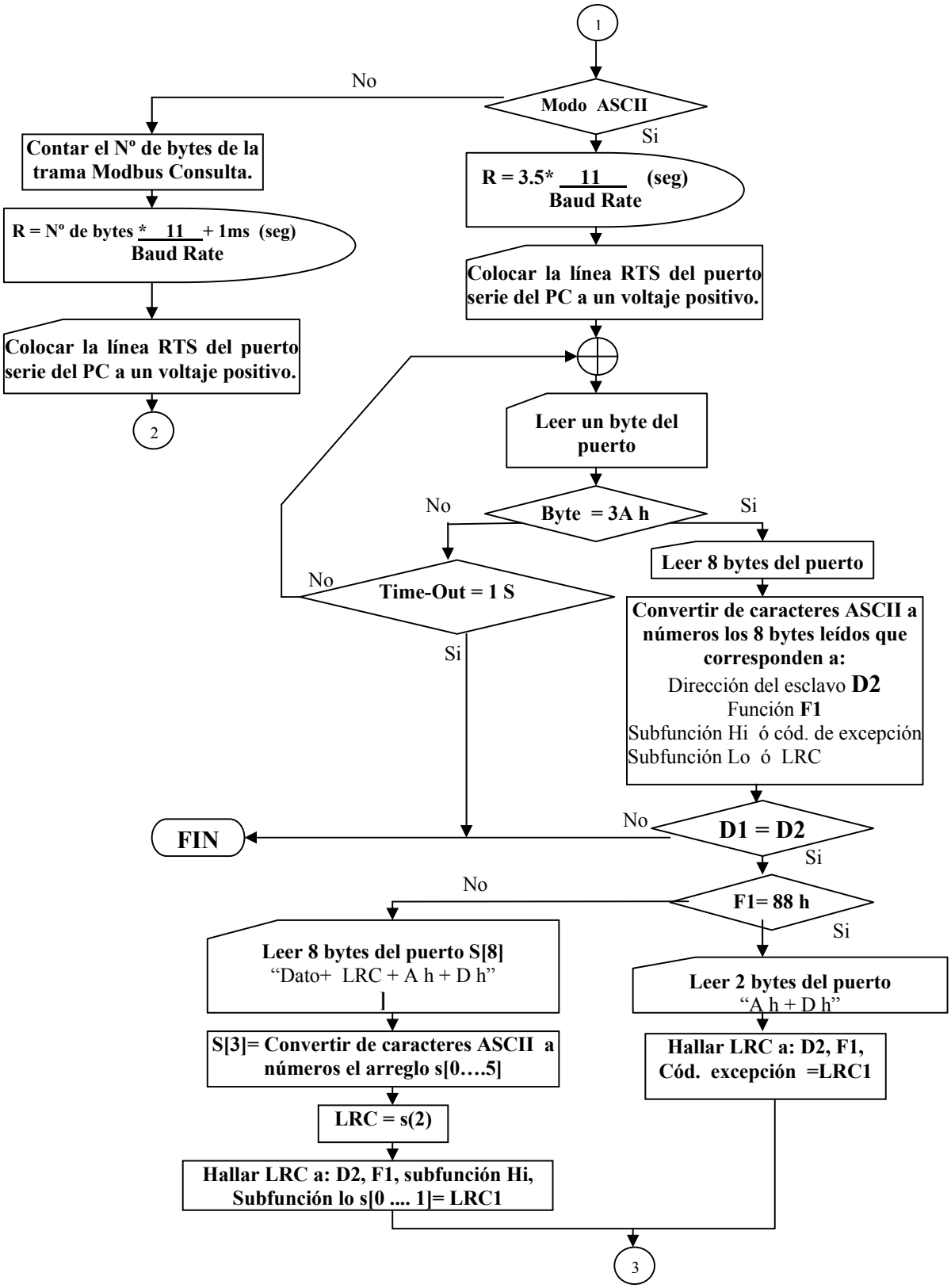
Código de Subfunción		**Nombre**	"Campo de información" **Mensaje Consulta**	"Campo de información" **Mensaje Respuesta**
00 d	**00 h**	**Devuelve la misma información**	xx xx h	xx xx h
11 d	**0B h**	**Devuelve contador de mensaje de bus.**	00 00 h	Número de mensajes detectados por el esclavo.
12 d	**0C h**	**Devuelve contador de error de bus de comunicación.**	00 00 h	Número de errores CRC detectados por el esclavo.
13 d	**0D h**	**Devuelve contador de error de excepción en el bus.**	00 00 h	Número de respuestas de excepción devueltas por el esclavo.
14 d	**0E h**	**Devuelve contador de mensajes del esclavo.**	00 00 h	Numero de mensajes procesados por el esclavo.
15 d	**0F h**	**Devuelve contador no respuesta del esclavo.**	00 00 h	Número de mensajes que el esclavo no contestó.
16 d	**10 h**	**Devuelve contador NAK del esclavo.**	00 00 h	Número de mensajes de respuesta de excepción NAK.

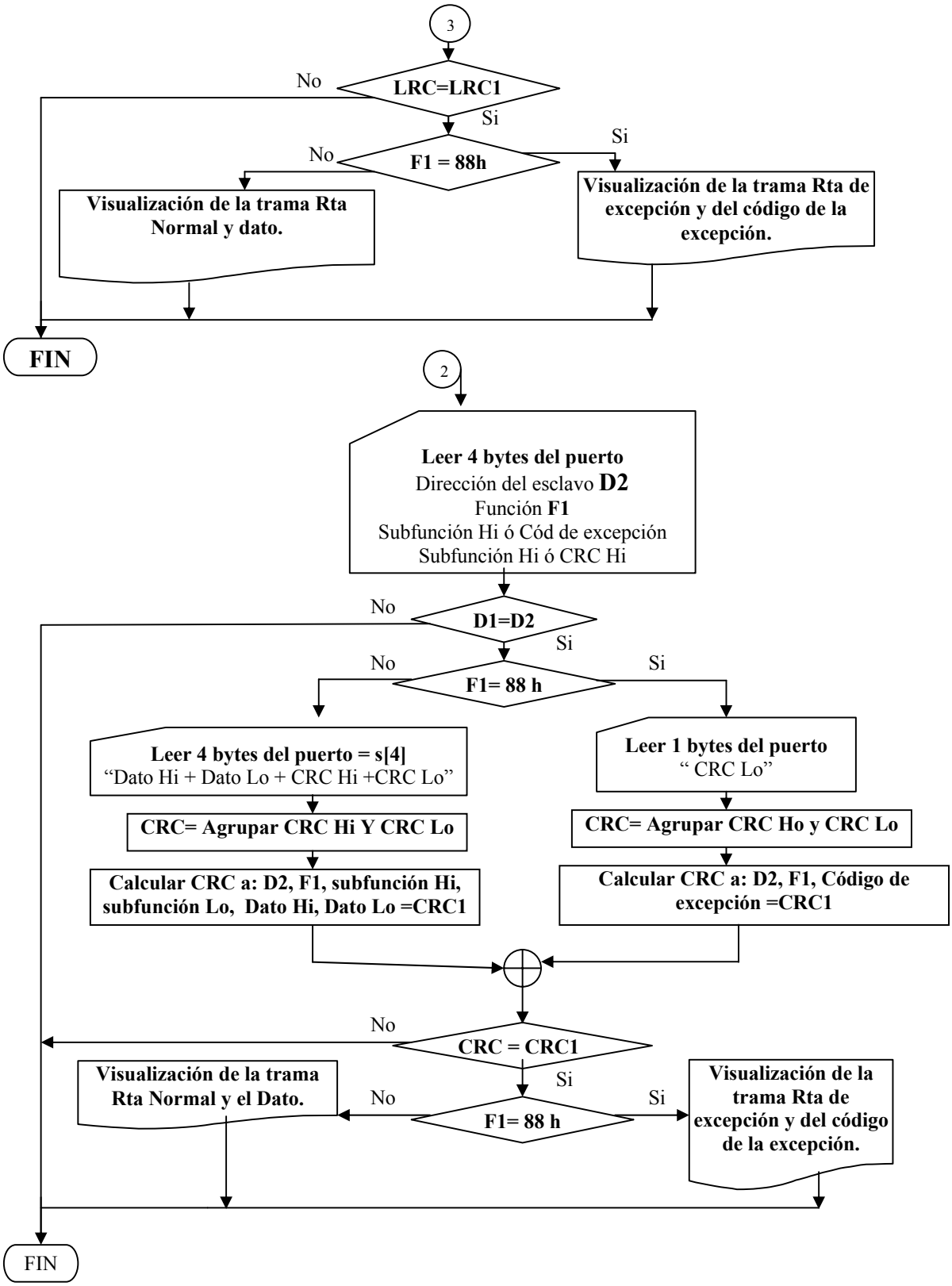
Donde xx xx h es cualquier número con una longitud de 2 bytes.

En la Figura 85 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 08.

Figura 85. Diagrama de flujo correspondiente a la función 08



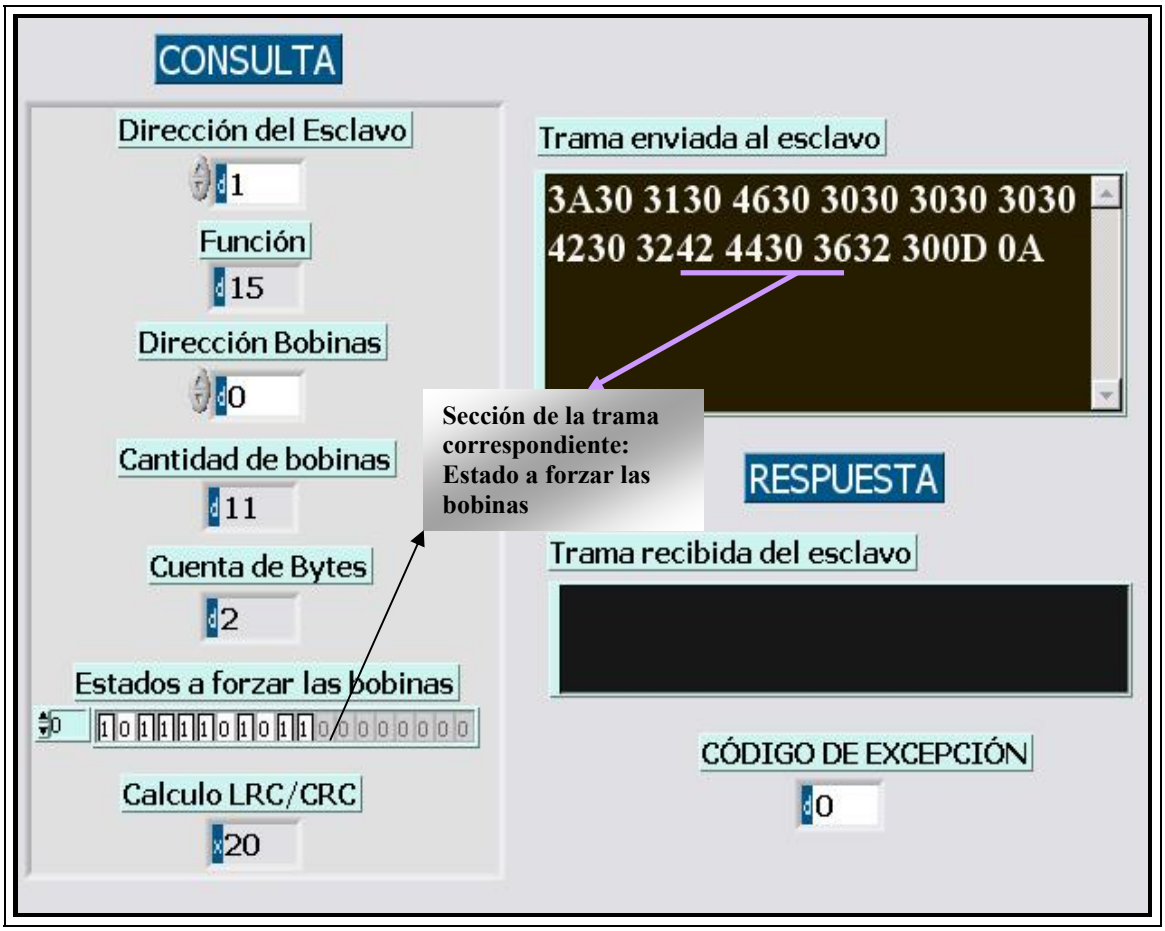

1
No
Modo ASCII
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = 3.5* 11 (seg)
Baud Rate
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
Leer un byte del puerto
No
Byte = 3A h
Si
No
Time-Out = 1 S
Si
Leer 8 bytes del puerto
Convertir de caracteres ASCII a números los 8 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
Subfunción Hi ó cód. de excepción
Subfunción Lo ó LRC
FIN
No
D1 = D2
Si
No
F1= 88 h
Si
Leer 8 bytes del puerto S[8]
"Dato+ LRC + A h + D h"
Leer 2 bytes del puerto
"A h + D h"
S[3]= Convertir de caracteres ASCII a números el arreglo s[0….5]
Hallar LRC a: D2, F1, Cód. excepción =LRC1
LRC = s(2)
Hallar LRC a: D2, F1, subfunción Hi, Subfunción lo s[0 .... 1]= LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 88h
Si
Visualización de la trama Rta Normal y dato.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN
2
Leer 4 bytes del puerto
Dirección del esclavo D2
Función F1
Subfunción Hi ó Cód de excepción
Subfunción Hi ó CRC Hi
No
D1=D2
Si
No
F1= 88 h
Si
Leer 4 bytes del puerto = s[4]
"Dato Hi + Dato Lo + CRC Hi +CRC Lo"
CRC= Agrupar CRC Hi Y CRC Lo
Calcular CRC a: D2, F1, subfunción Hi, subfunción Lo, Dato Hi, Dato Lo =CRC1
Leer 1 bytes del puerto
" CRC Lo"
CRC= Agrupar CRC Ho y CRC Lo
Calcular CRC a: D2, F1, Código de excepción =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal y el Dato.
No
F1= 88 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.10 FUNCIÓN 15: FORZAR MÚLTIPLES "BOBINAS"

La función 15 del protocolo Modbus permite forzar varias "bobinas" (ON ó OFF) consecutivas. En la Figura 86 se observa la sección del panel frontal correspondiente a la función 15.

Figura 86. Sección del panel frontal correspondiente a la función 15 del protocolo Modbus



Para iniciar la consulta, el usuario, debe ingresar la siguiente información:

- Dirección del esclavo (decimal)
- Dirección bobinas (decimal)
- Estados a forzar las bobinas (decimal)

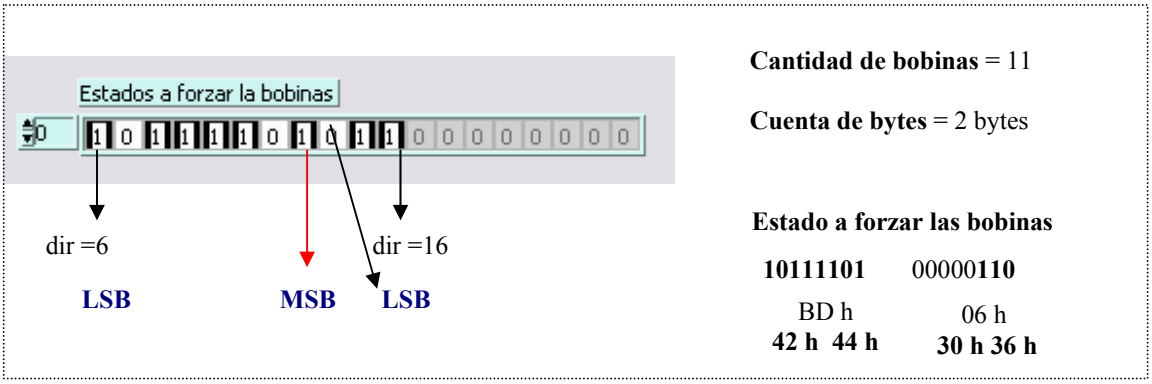
Y el programa calcula los siguientes campos del mensaje consulta para completar la trama:

- Cantidad de bobinas (decimal)
- Cuenta de bytes (decimal)
- CRC/LRC (hexadecimal)

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica la dirección de la primera bobina (**Dirección bobinas**) a forzar, la **Cantidad de Bobinas**, la **Cuenta de Bytes** (una bobina por bit), y los **Estados a Forzar las Bobinas.**

Ejemplo:



En la Figura 87 se presenta el formato del mensaje consulta para la función 15 y se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

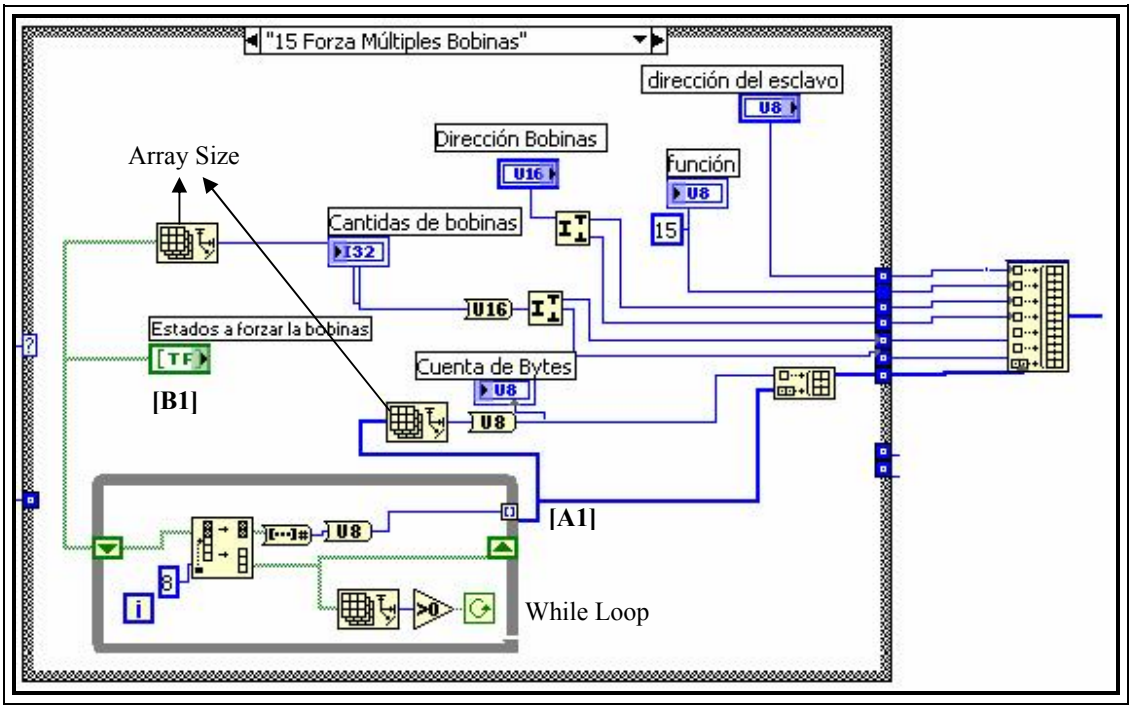
Figura 87. Formato del Mensaje Consulta Función 15

	Dirección del esclavo	Función	Dirección de bobinas	Cantidad de bobinas	Cuenta de bytes (N)	Estado a forzar Las bobinas	CRC LRC	
T1-T2-T3-T4	1 byte	0F h	2 byte	2 byte	1 byte	N bytes	2byte	T1-T2-T3-T4
3A h	2 CAR ASCII	30 h 46 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	2*N CAR ASCII	2 CAR ASCII	D h A h

En la Figura 88 se presenta la construcción del mensaje consulta para la función 15. Observe que el arreglo de bits (que contiene los **Estados a forzar las "bobinas"**) **B1** ingresa a la estructura *While Loop*, allí se agrupan en paquetes de bytes formando un arreglo de bytes llamado **A1.**

La función *Array Size* cuenta el número de bytes del arreglo **A1** (**Cuenta de Bytes**) y el número de bits del arreglo **B1** (**Cantidad de bobinas**). Al igual que la función 01, se concatenan todos los campos del mensaje, seguidamente se calcula el control de errores (LRC ó CRC), se construye la trama completa y se escribe en el puerto.

Figura 88. Construcción del mensaje consulta para la función 15



Formato del Mensaje Respuesta

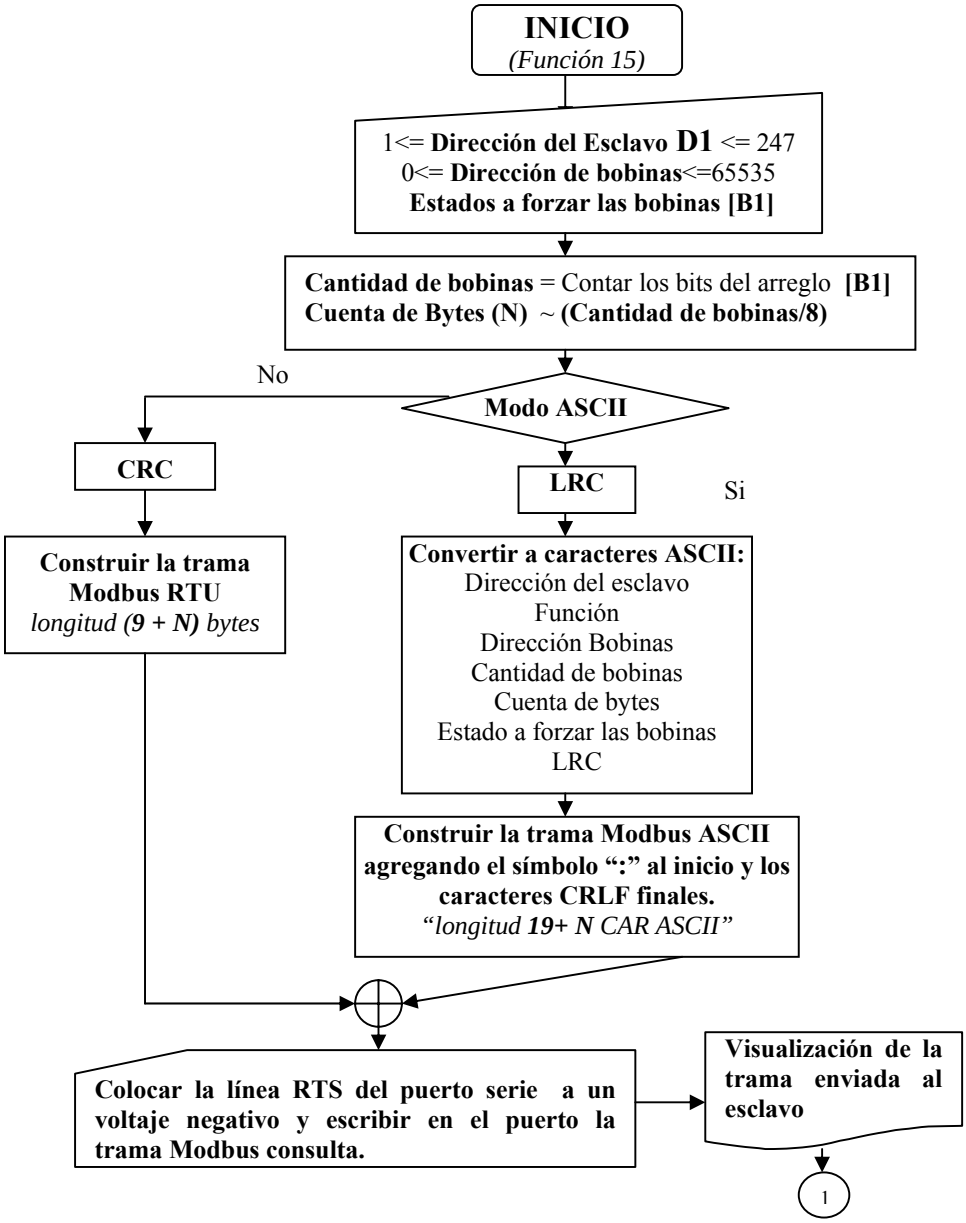
En el mensaje respuesta, el dispositivo retorna la dirección del esclavo, el código de operación, la dirección de bobinas y la cantidad de bobinas. En la Figura 89 se presenta el formato del mensaje respuesta en los dos modos de transmisión y el tamaño de cada campo del mensaje.

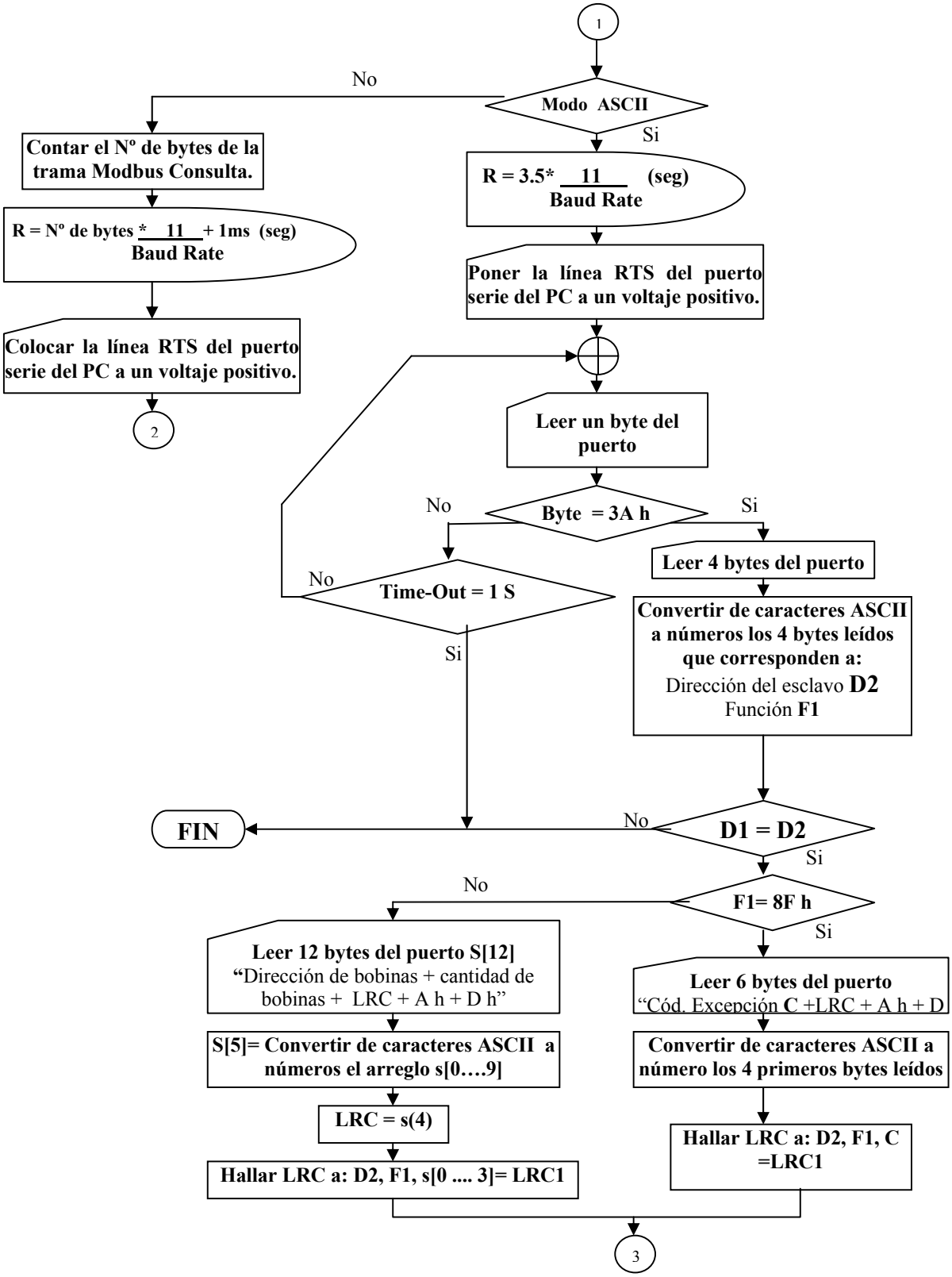
Figura 89. Formato del Mensaje Respuesta Función 15

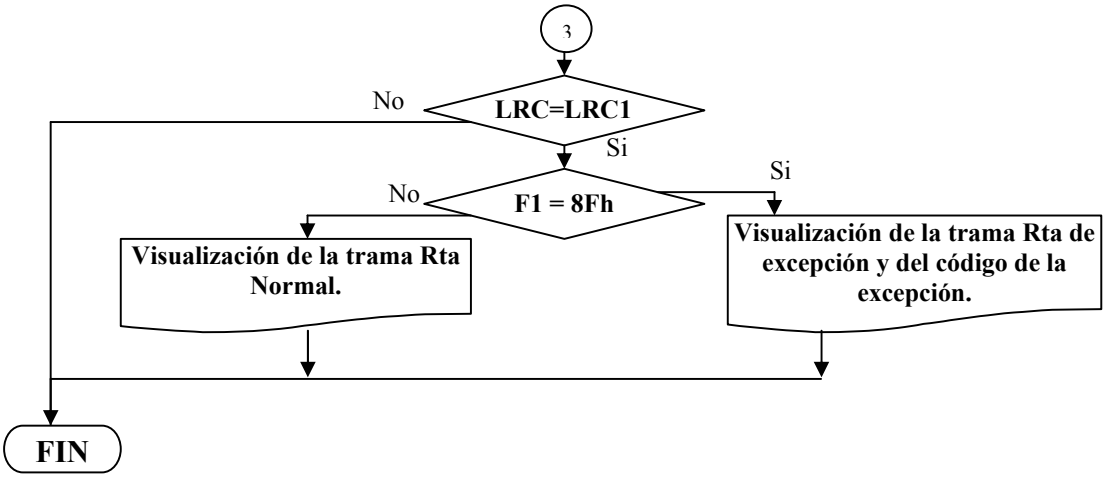
	Dirección del esclavo	Función	Dirección de Bobinas	Cantidad de Bobinas	CRC/LRC		
T1-T2-T3-T4	1 byte	0F h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	30 h 46 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

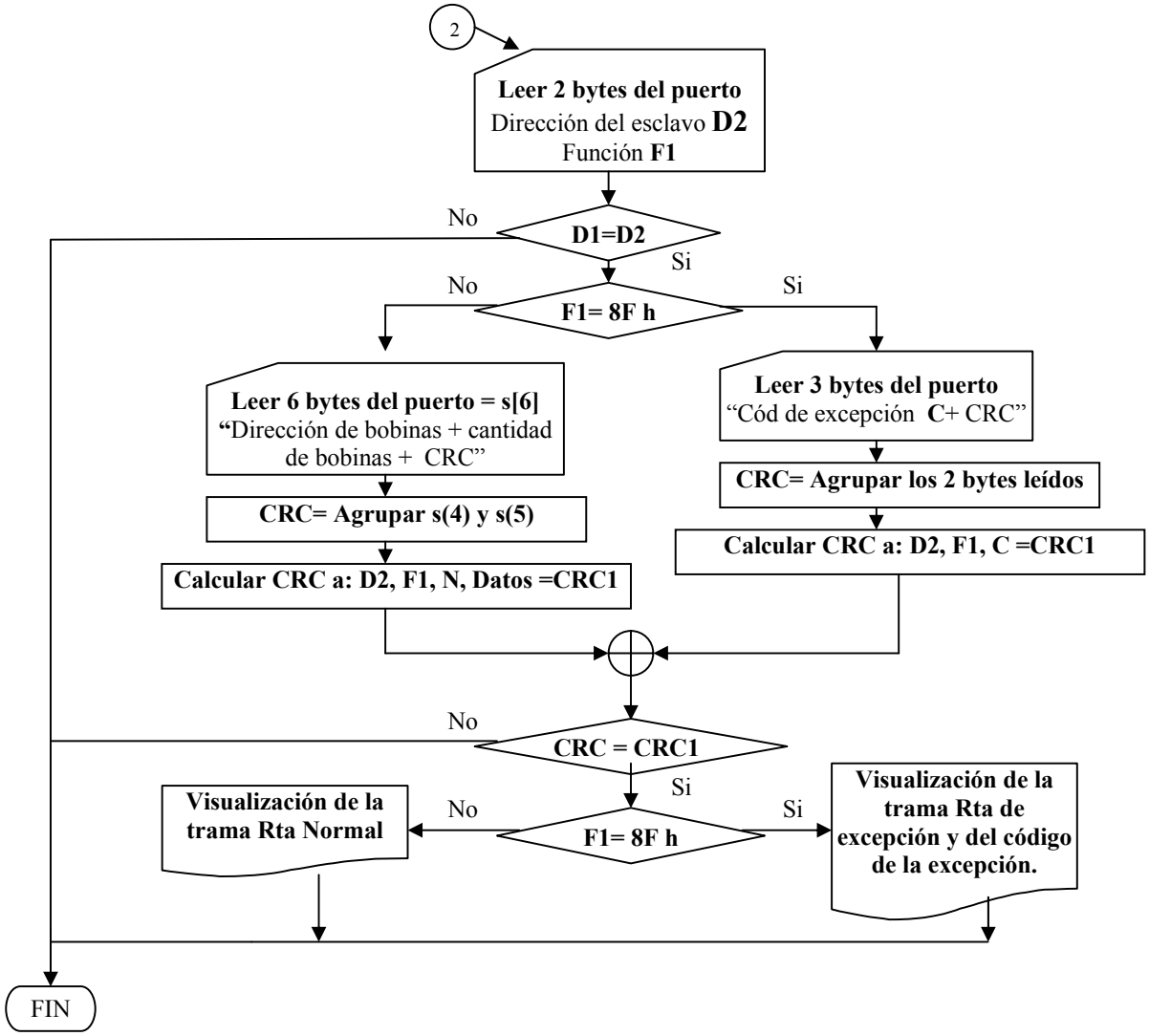
En la Figura 90 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 15.

Figura 90. Diagrama de flujo correspondiente a la función 15




1
Modo ASCII
No
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
R = 3.5* 11 (seg)
Baud Rate
Poner la línea RTS del puerto serie del PC a un voltaje positivo.
Leer un byte del puerto
Byte = 3A h
No
Si
Time-Out = 1 S
No
Si
Leer 4 bytes del puerto
Convertir de caracteres ASCII a números los 4 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
FIN
D1 = D2
No
Si
F1= 8F h
No
Si
Leer 12 bytes del puerto S[12]
"Dirección de bobinas + cantidad de bobinas + LRC + A h + D h"
S[5]= Convertir de caracteres ASCII a números el arreglo s[0….9]
LRC = s(4)
Hallar LRC a: D2, F1, s[0 .... 3]= LRC1
Leer 6 bytes del puerto
"Cód. Excepción C +LRC + A h + D
Convertir de caracteres ASCII a número los 4 primeros bytes leídos
Hallar LRC a: D2, F1, C =LRC1
3

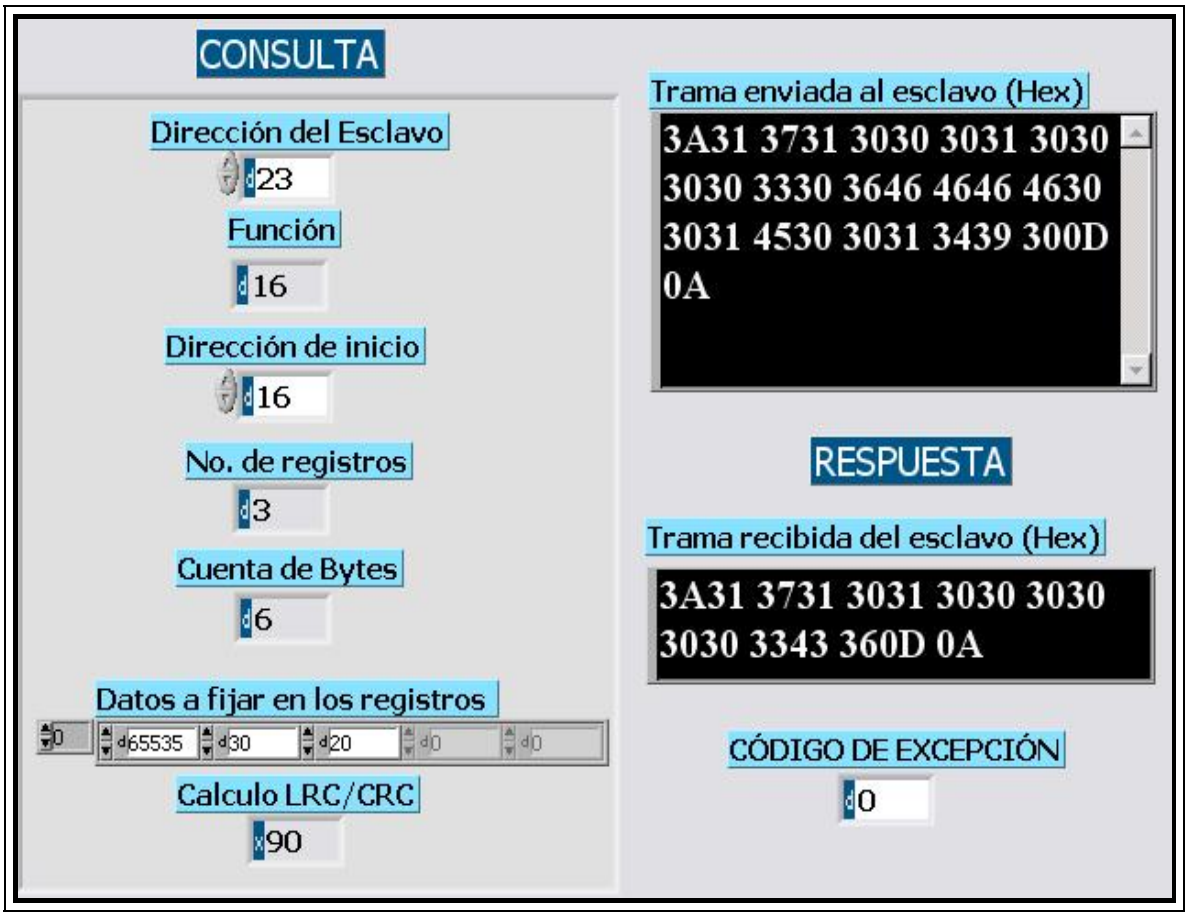

3
No
LRC=LRC1
Si
No
F1 = 8Fh
Si
Visualización de la trama Rta Normal.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN


2
Leer 2 bytes del puerto
Dirección del esclavo D2
Función F1
No
D1=D2
Si
No
F1= 8F h
Si
Leer 6 bytes del puerto = s[6]
"Dirección de bobinas + cantidad de bobinas + CRC"
CRC= Agrupar s(4) y s(5)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 3 bytes del puerto
"Cód de excepción C+ CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, C =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal
No
F1= 8F h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

4.11 FUNCIÓN 16: FORZAR MÚLTIPLES REGISTROS

La función 16 del protocolo Modbus permite forzar varios registros internos consecutivos; en la Figura 91 se observa la sección del panel frontal correspondiente a la función 16.

Figura 91. Sección del panel frontal correspondiente a la función 16 del protocolo Modbus



Para iniciar la consulta, el usuario, debe ingresar la siguiente información:

- Dirección del esclavo (decimal)
- Dirección de inicio (decimal)
- Datos a fijar en el registro (decimal)

Y el programa calcula los siguientes campos del mensaje consulta para completar la trama:

- Número de Registros (decimal)

- Cuenta de bytes (decimal)
- CRC/LRC (hexadecimal)

Formato del Mensaje Consulta

En el campo de información del mensaje consulta se especifica la dirección del primer registro (**Dirección bobinas**) a forzar, el **Número de registros**, la **Cuenta de Bytes** (2 Bytes por registro), y **los Datos a fijas en los registros.** En la Figura 92 se presenta el formato del mensaje consulta para la función 16 y se incluye el tamaño de cada campo del mensaje para los modos RTU y ASCII.

Figura 92. Formato del Mensaje Consulta Función 16

	Dirección del esclavo	Función	Dirección de inicio	Nº de Registros	Cuenta de bytes (N)	Datos a fijar en los registros	CRC LRC	
T1-T2-T3-T4	1 byte	10 h	2 byte	2 byte	1 byte	N bytes	2byte	T1-T2-T3-T4
3A h	2 CAR ASCII	31 h 30 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	2*N CAR ASCII	2 CAR ASCII	D h A h

Formato del Mensaje Respuesta

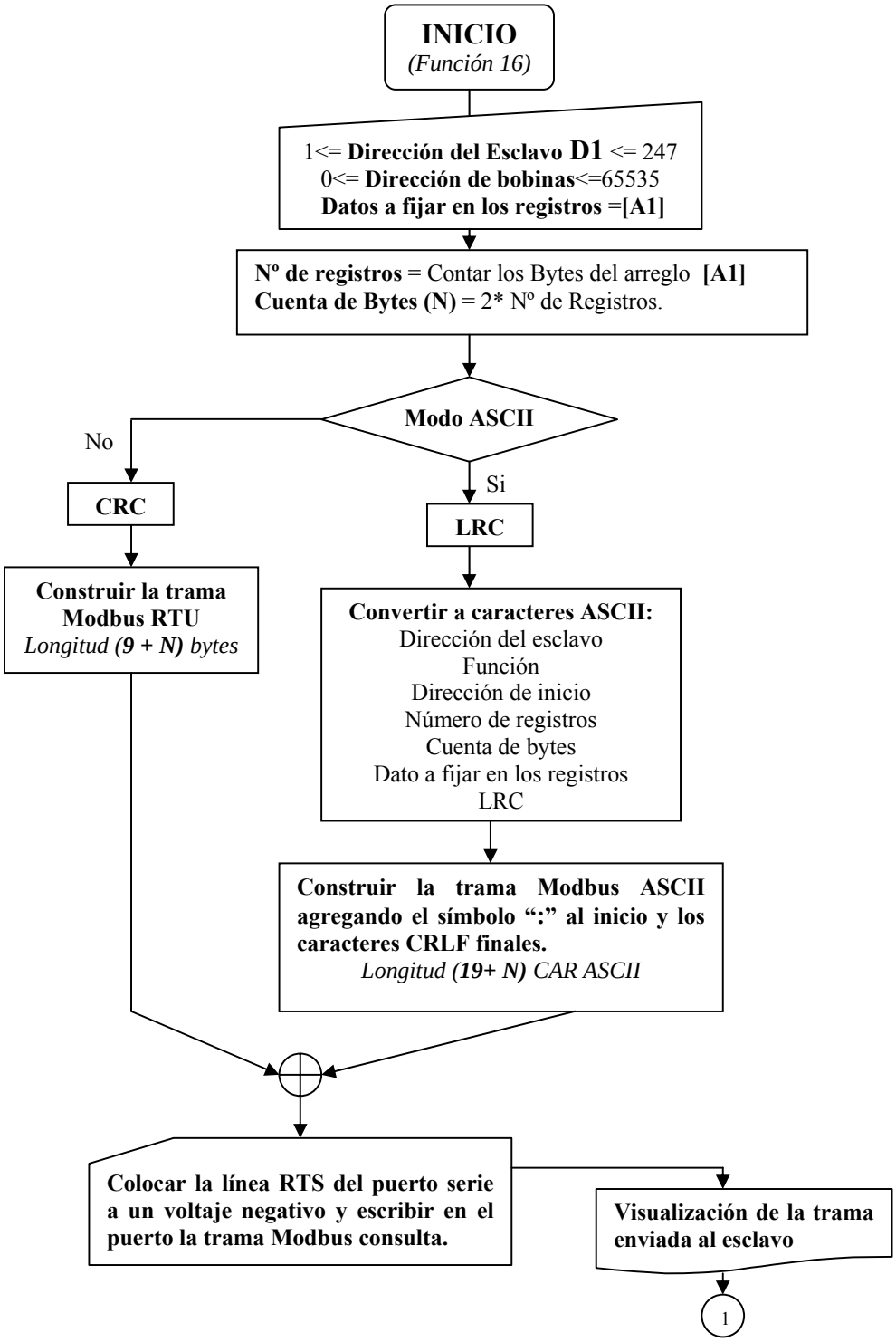
En el mensaje respuesta, para una respuesta normal el dispositivo retorna la dirección del esclavo, el código de operación, la dirección de inicio y el número de registros. En la Figura 93 se presenta el formato del mensaje respuesta en los dos modos de transmisión y el tamaño de cada campo del mensaje.

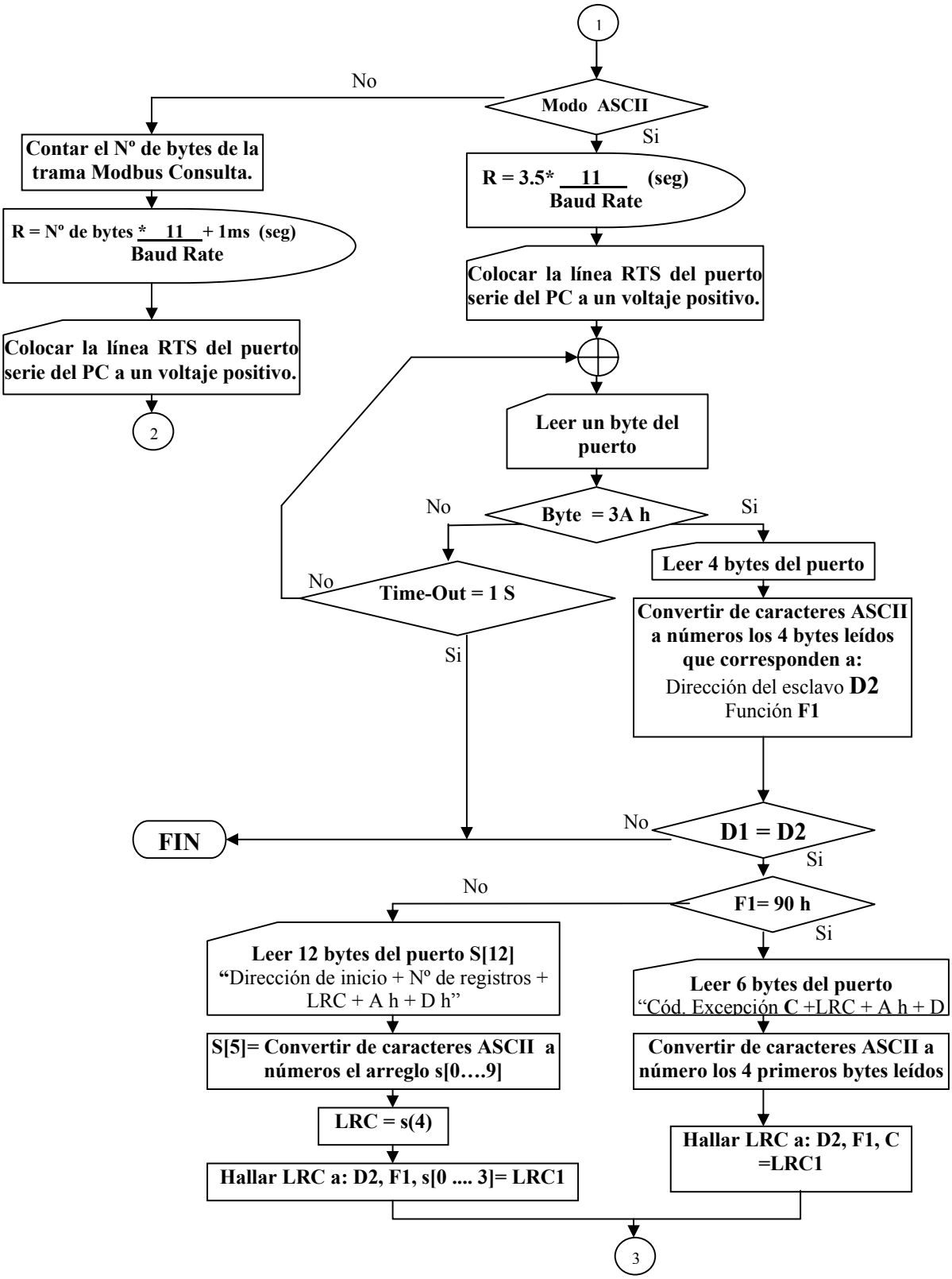
Figura 93. Formato del Mensaje Respuesta Función 16

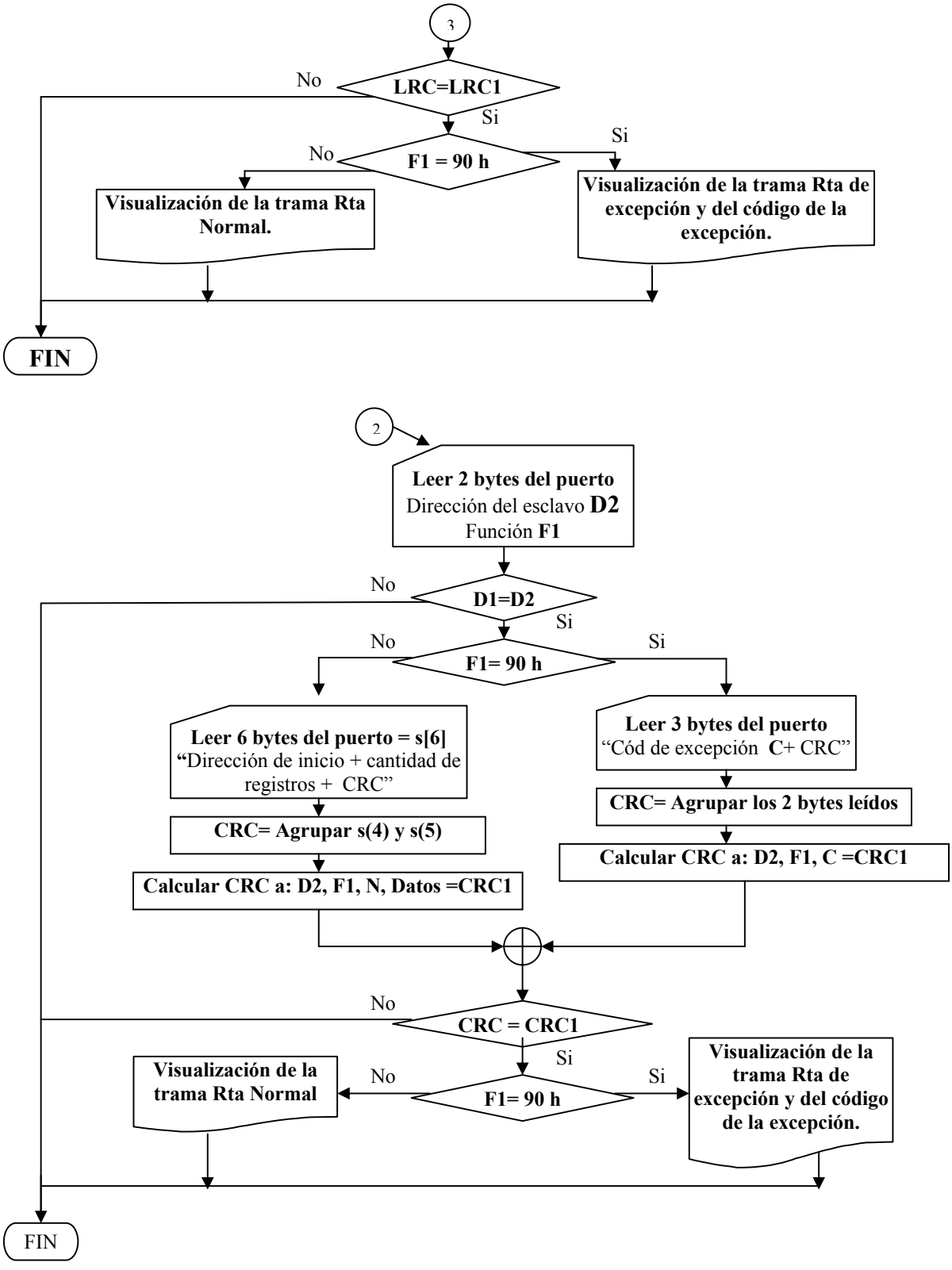
	Dirección del esclavo	Función	Dirección de inicio	Número de Registros	CRC/LRC		
T1-T2-T3-T4	1 byte	10 h	2 byte	2 byte	2byte	T1-T2-T3-T4	RTU
3A h	2 CAR ASCII	31 h 30 h	4 CAR ASCII	4 CAR ASCII	2 CAR ASCII	D h A h	ASCII

En la Figura 94 se presenta el diagrama de flujo correspondiente a la programación realizada en LabVIEW para la función 16.

Figura 94. Diagrama de flujo correspondiente a la función 16




1
No
Modo ASCII
Si
Contar el Nº de bytes de la trama Modbus Consulta.
R = 3.5* 11 (seg)
Baud Rate
R = Nº de bytes * 11 + 1ms (seg)
Baud Rate
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
Colocar la línea RTS del puerto serie del PC a un voltaje positivo.
2
Leer un byte del puerto
No
Byte = 3A h
Si
No
Time-Out = 1 S
Leer 4 bytes del puerto
Convertir de caracteres ASCII a números los 4 bytes leídos que corresponden a:
Dirección del esclavo D2
Función F1
Si
FIN
No
D1 = D2
Si
No
F1= 90 h
Si
Leer 12 bytes del puerto S[12]
"Dirección de inicio + Nº de registros + LRC + A h + D h"
Leer 6 bytes del puerto
"Cód. Excepción C +LRC + A h + D
S[5]= Convertir de caracteres ASCII a números el arreglo s[0….9]
Convertir de caracteres ASCII a número los 4 primeros bytes leídos
LRC = s(4)
Hallar LRC a: D2, F1, s[0 .... 3]= LRC1
Hallar LRC a: D2, F1, C =LRC1
3


3
No
LRC=LRC1
Si
No
F1 = 90 h
Si
Visualización de la trama Rta Normal.
Visualización de la trama Rta de excepción y del código de la excepción.
FIN
2
Leer 2 bytes del puerto
Dirección del esclavo D2
Función F1
No
D1=D2
Si
No
F1= 90 h
Si
Leer 6 bytes del puerto = s[6]
"Dirección de inicio + cantidad de registros + CRC"
CRC= Agrupar s(4) y s(5)
Calcular CRC a: D2, F1, N, Datos =CRC1
Leer 3 bytes del puerto
"Cód de excepción C+ CRC"
CRC= Agrupar los 2 bytes leídos
Calcular CRC a: D2, F1, C =CRC1
No
CRC = CRC1
Si
Visualización de la trama Rta Normal
No
F1= 90 h
Si
Visualización de la trama Rta de excepción y del código de la excepción.
FIN

5. SNIFFER MODBUS: PROGRAMADO EN LABVIEW

En la Figura 95 esta el panel frontal del SNIFFER Modbus; este programa captura la información que viaja por la red Modbus durante un intervalo de tiempo (Tiempo de adquisición) definido por el usuario y seguidamente visualiza los mensajes consulta-respuesta de un esclavo específico. Los parámetros que el usuario debe configurar son:

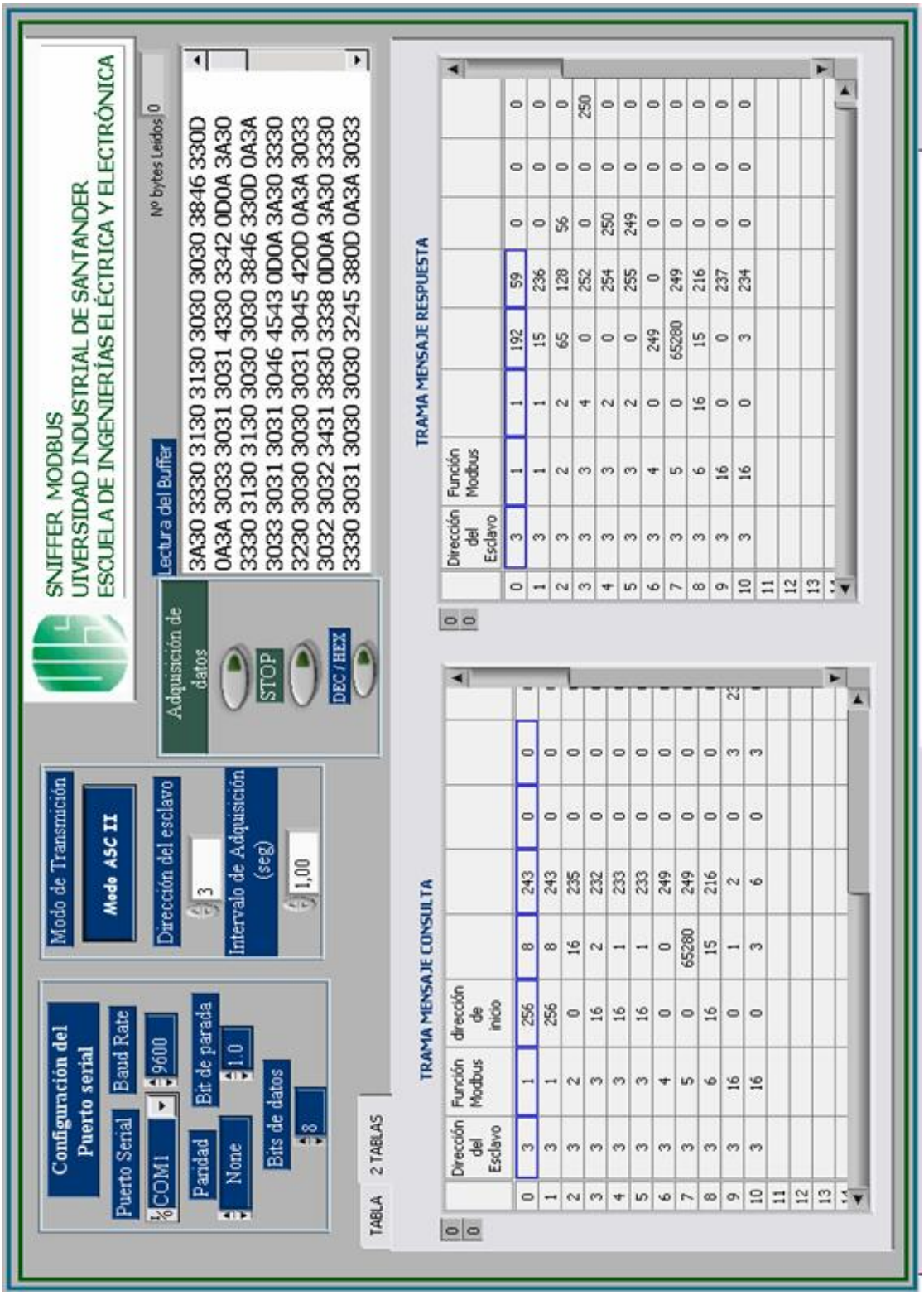
Puerto serie: (com1 ó com 2)
Baud Rate: (9600 bips/seg por defecto)
Paridad: *none* "ninguna", *odd* "impar", *even* "par".

Bit de parada: 1, 1.5 o 2 bits
Bits de datos: 7 o 8 bits
Modo de transmisión de la red: ASCII ó RTU
Dirección del Esclavo: Identificación del dispositivo al cual se quiere discriminar las tramas Consulta/Respuesta.
Tiempo de adquisición: tiempo que dura la adquisición de tramas.
Adquisición de datos: Inicia la captura de las tramas que fluyen por a red.
Stop: Detiene el programa.
Dec/Hex: Con este control se selecciona si se quieren ver los datos de la tabla en decimal (por defecto) o en hexadecimal.

Visualización:

Lectura del buffer: En este indicador *string* se observan todas las tramas capturadas en la red.
2 Tabla: En la primera tabla se observan en cada fila la trama de un mensajes consulta y en la segunda tabla se observa en cada fila la trama de un mensaje respuesta normal (funciones: 01, 02, 03, 04, 05, 06, 15, 16) para un esclavo especifico (las filas de las dos tablas son equivalentes).

Figura 95. Panel frontal SNIFFER MODBUS



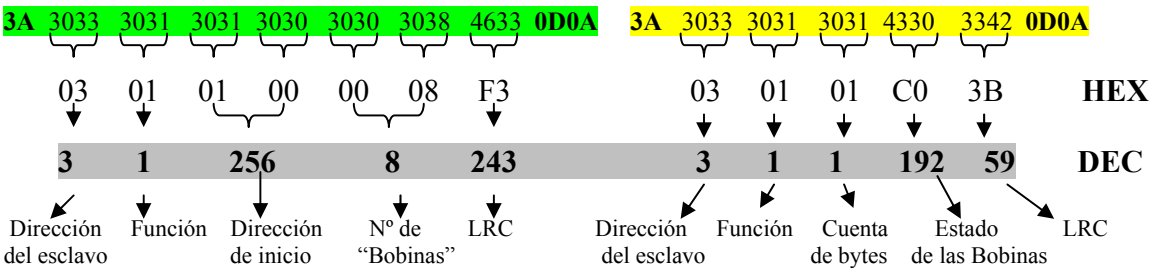
En modo ASCII el **"SNIFFER MODBUS"**, toma todas las tramas del arreglo **Lectura del Buffer [s]** y les quita los caracteres ASCII **":"** (3A h), **CR** (D h) y **LF** (A h) obteniendo el arreglo **[s2]**, luego convierte todos los caracteres ASCII del arreglo **[s2]** a números (este arreglo se llama **[s3]**). Seguidamente el programa busca en el arreglo **[s3]** el campo que corresponde a la dirección del esclavo, cuando lo encuentra verifica el **LRC** para diferentes longitudes de tramas Consulta, si alguno es correcto, toma los siguientes bytes del arreglo para confirmar si es una respuesta para esa trama. Si las dos tramas son correctas se visualizan en la primera fila de la tabla, este procedimiento se repite para todo el resto del arreglo **[s3]**.

Ejemplo:

En el visualizador "*string*" Lectura del Buffer se tiene el arreglo [s] en ASCII:

3A30 3330 3130 3130 3030 3030 3846 330D 0A 3A 3033 3031 3031 4330 3342 0D0A 3A30 3330 3130 3130 3030 3030 3846 330D 0A 3A 3033 3031 3031 3046 4543 0D0A 3A30 3330 3230 3030 3030 3031 3045 420D 0A 3A 3033 3032 3032 3431 3830 3338 0D0A 3A30 3330 3330 3031 3030 3030 3245 380D 0A 3A 3033 3033 3034 3030 4643 3030 3030 4641 0D0A 3A30 3330 3330 3031 3030 3030 3145 390D 0A 3A 3033 3033 3032 3030 4645 4641 0D0A 3A30 3330 3330 3031 3030 3030 3145 390D 0A 3A 3033 3033 3032 3030 4646 4639 0D0A……..

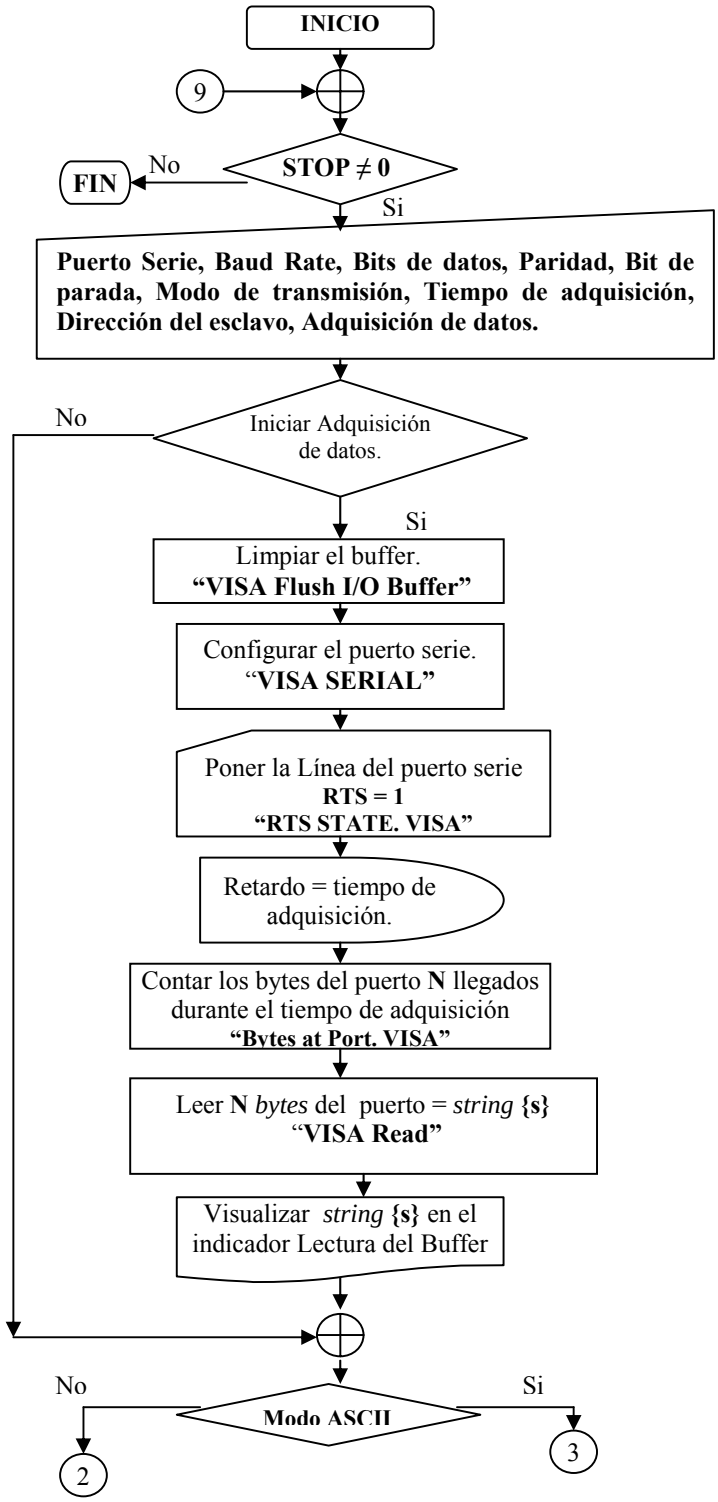
Las tramas Modbus sombreadas (consulta/repuesta) en el string Lectura del buffer se convierten de caracteres ASCII a números y cada dato se interpresta así:

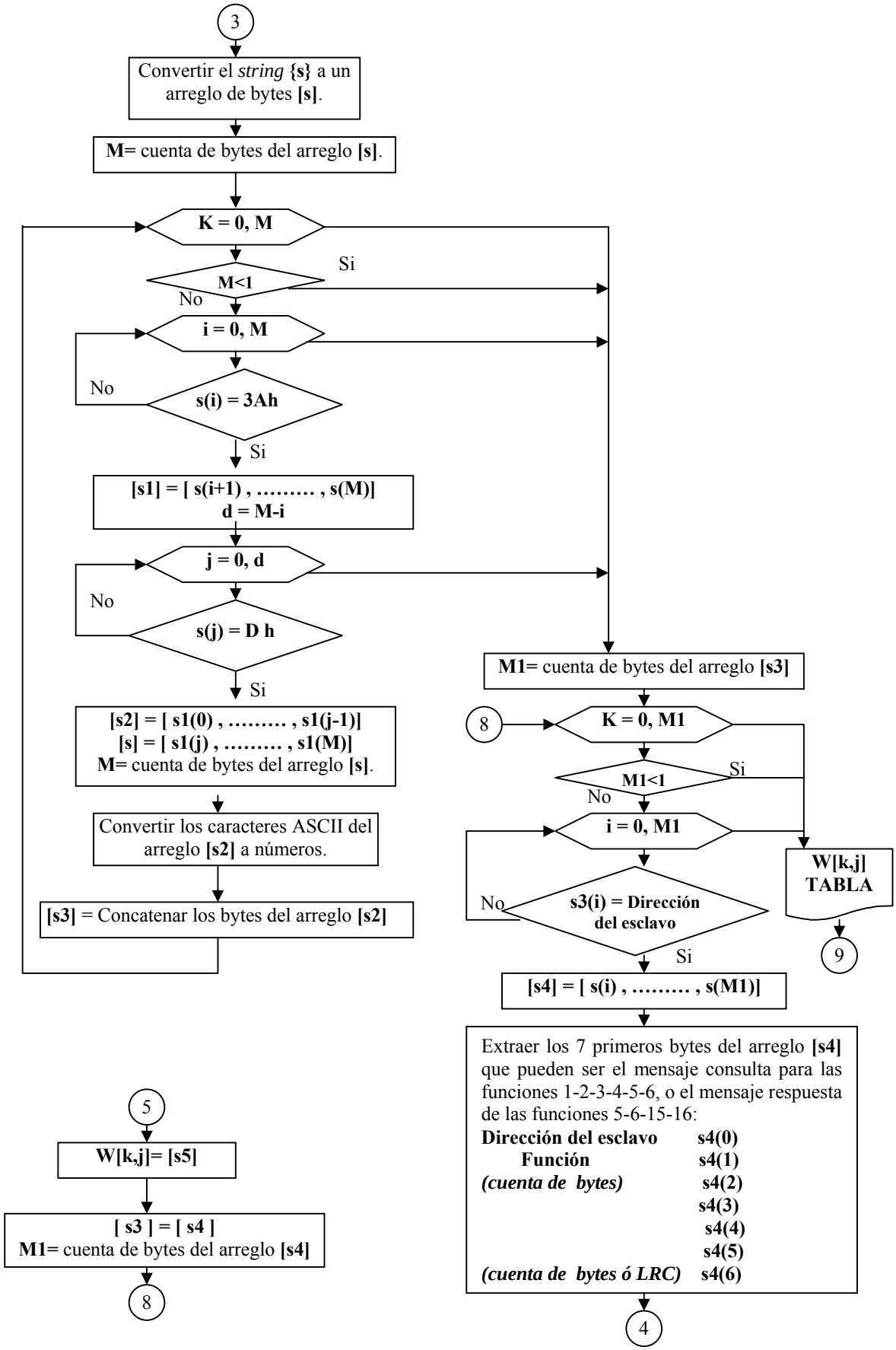


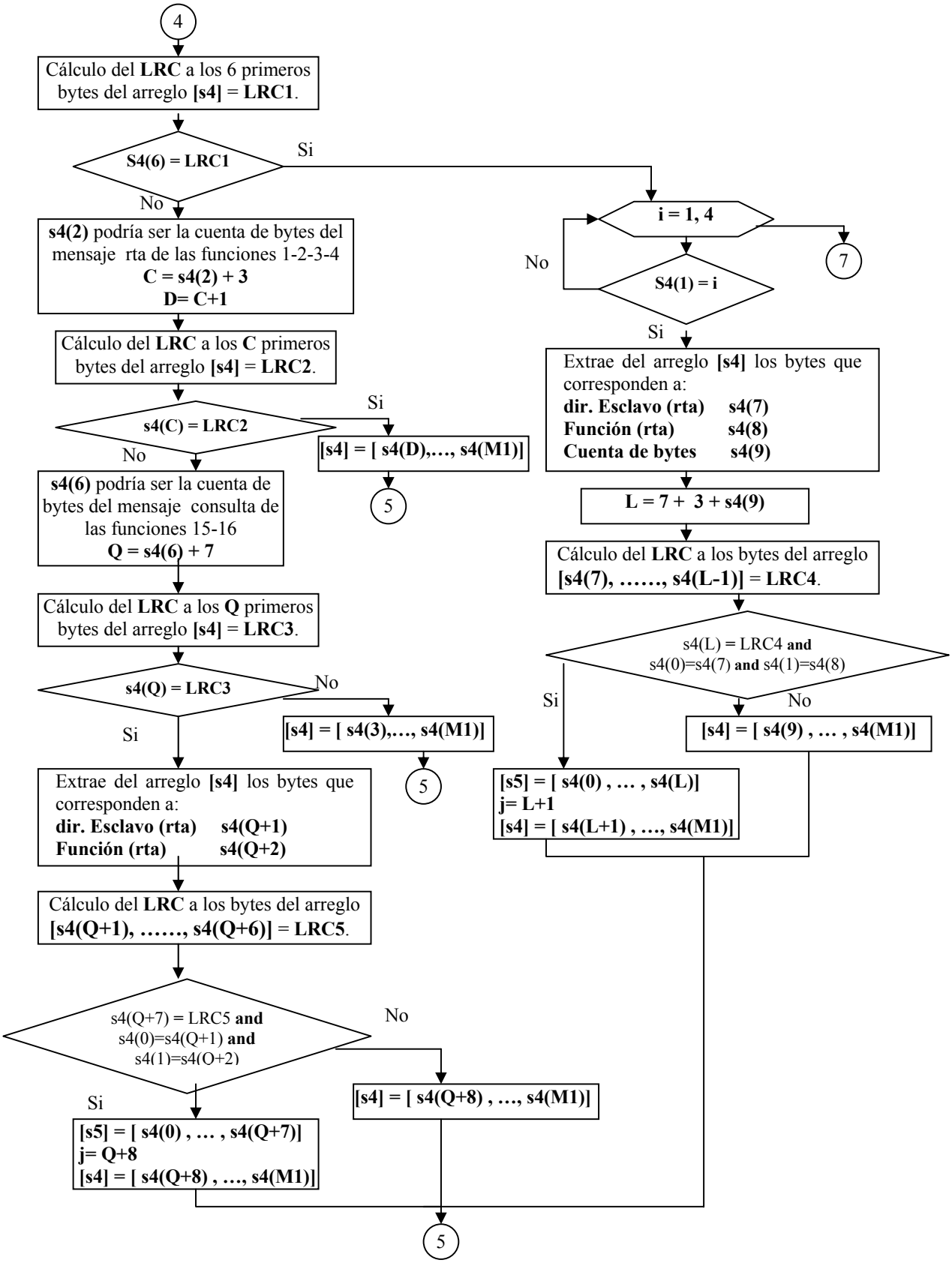
Observe que los números sombreados (decimal) son los mismos que aperecen en la primera fila de la tabla de la Figura 95, para interpretar correctamente estos datos el usuario debe tener una guia a la mano del Protocolo Modbus. Los ceros que se agregan al final de las filas son inválidos, aparecen porque la tabla se ajusta a la trama consulta/respuesta con mayor número de datos.

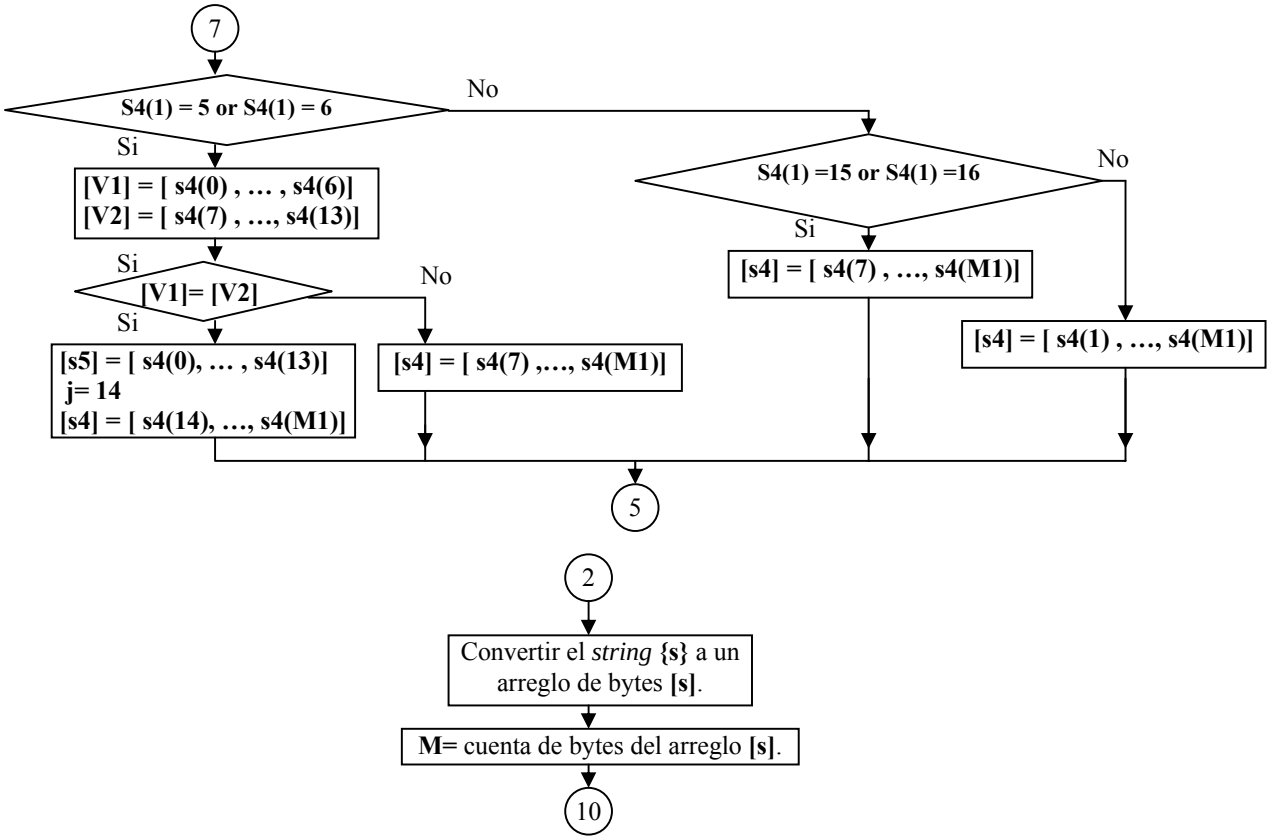
Para Modo RTU la interpretación de los datos es similar, la diferencia radica en que el cálculo de errores se realiza con el CRC. A continuación se presenta el diagrama de flujo de la programación realizada en LabVIEW para el SNIFFER MODBUS.

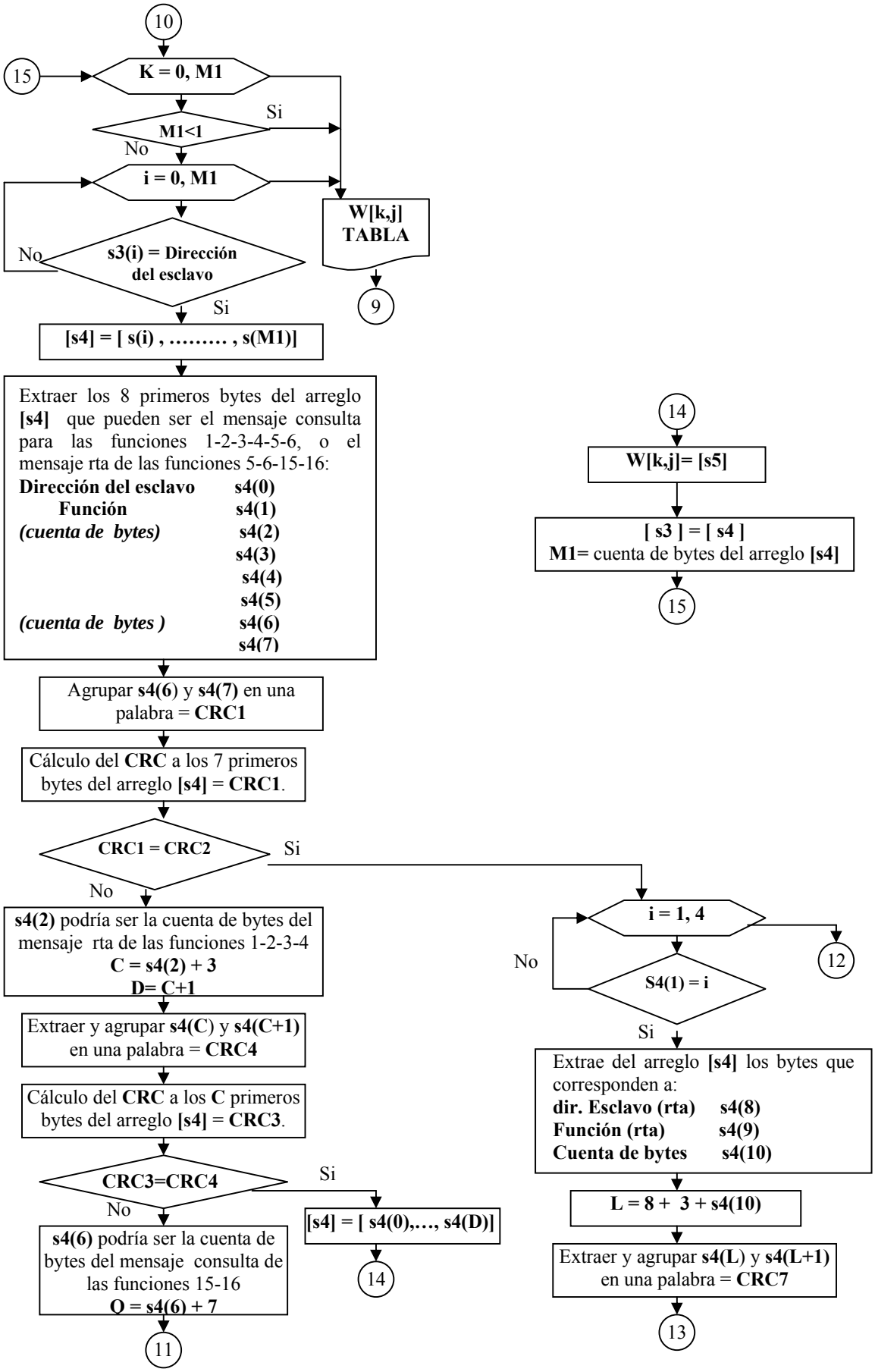
5.1 DIAGRAMA DE FLUJO SNIFFER MODBUS

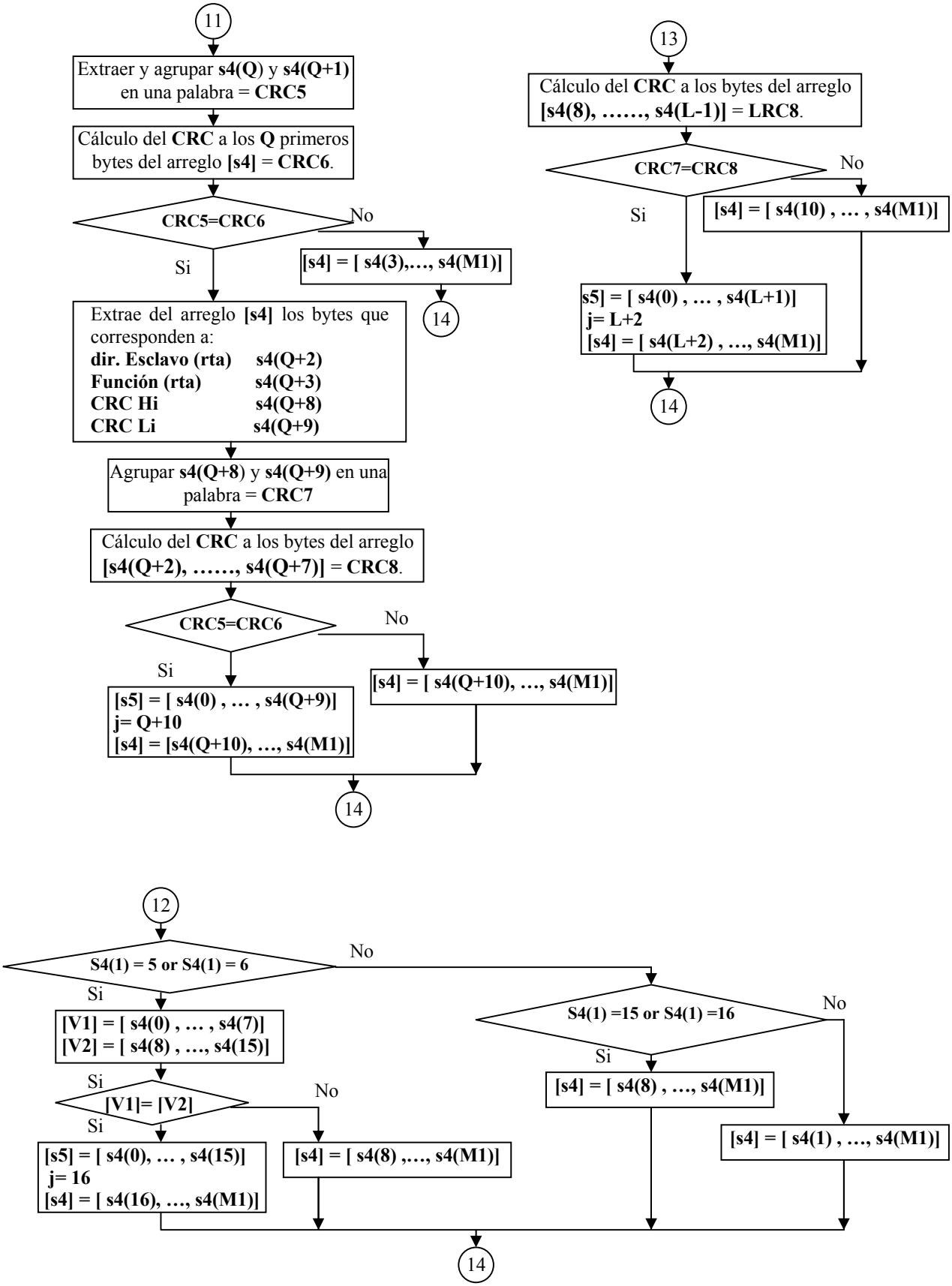



3
Convertir el *string* {s} a un arreglo de bytes [s].
M= cuenta de bytes del arreglo [s].
K = 0, M
M<1
Si
No
i = 0, M
No
s(i) = 3Ah
Si
[s1] = [s(i+1) , …… , s(M)]
d = M-i
j = 0, d
No
s(j) = D h
Si
[s2] = [s1(0) , …… , s1(j-1)]
[s] = [s1(j) , …… , s1(M)]
M= cuenta de bytes del arreglo [s].
Convertir los caracteres ASCII del arreglo [s2] a números.
[s3] = Concatenar los bytes del arreglo [s2]
M1= cuenta de bytes del arreglo [s3]
8
K = 0, M1
M1<1
Si
No
i = 0, M1
No
s3(i) = Dirección del esclavo
Si
W[k,j] TABLA
9
[s4] = [s(i) , …… , s(M1)]
Extraer los 7 primeros bytes del arreglo [s4] que pueden ser el mensaje consulta para las funciones 1-2-3-4-5-6, o el mensaje respuesta de las funciones 5-6-15-16:
Dirección del esclavo s4(0)
Función s4(1)
(cuenta de bytes) s4(2)
s4(3)
s4(4)
s4(5)
(cuenta de bytes ó LRC) s4(6)
4
5
W[k,j]= [s5]
[s3] = [s4]
M1= cuenta de bytes del arreglo [s4]
8


4
Cálculo del LRC a los 6 primeros bytes del arreglo [s4] = LRC1.
S4(6) = LRC1
Si
No
s4(2) podría ser la cuenta de bytes del mensaje rta de las funciones 1-2-3-4
C = s4(2) + 3
D= C+1
Cálculo del LRC a los C primeros bytes del arreglo [s4] = LRC2.
s4(C) = LRC2
Si
No
[s4] = [s4(D),…, s4(M1)]
5
s4(6) podría ser la cuenta de bytes del mensaje consulta de las funciones 15-16
Q = s4(6) + 7
Cálculo del LRC a los Q primeros bytes del arreglo [s4] = LRC3.
s4(Q) = LRC3
No
Si
[s4] = [s4(3),…, s4(M1)]
5
Extrae del arreglo [s4] los bytes que corresponden a:
dir. Esclavo (rta) s4(Q+1)
Función (rta) s4(Q+2)
Cálculo del LRC a los bytes del arreglo [s4(Q+1), ……, s4(Q+6)] = LRC5.
s4(Q+7) = LRC5 and s4(0)=s4(Q+1) and s4(1)=s4(Q+2)
No
Si
[s4] = [s4(Q+8) , …, s4(M1)]
[s5] = [s4(0) , … , s4(Q+7)]
j= Q+8
[s4] = [s4(Q+8) , …, s4(M1)]
5
i = 1, 4
No
S4(1) = i
7
Si
Extrae del arreglo [s4] los bytes que corresponden a:
dir. Esclavo (rta) s4(7)
Función (rta) s4(8)
Cuenta de bytes s4(9)
L = 7 + 3 + s4(9)
Cálculo del LRC a los bytes del arreglo [s4(7), ……, s4(L-1)] = LRC4.
s4(L) = LRC4 and s4(0)=s4(7) and s4(1)=s4(8)
Si
No
[s5] = [s4(0) , … , s4(L)]
j= L+1
[s4] = [s4(L+1) , …, s4(M1)]
[s4] = [s4(9) , … , s4(M1)]


7
S4(1) = 5 or S4(1) = 6
No
Si
[V1] = [s4(0) , … , s4(6)]
[V2] = [s4(7) , …, s4(13)]
Si
[V1]= [V2]
No
Si
[s5] = [s4(0), … , s4(13)]
j= 14
[s4] = [s4(14), …, s4(M1)]
[s4] = [s4(7) ,…, s4(M1)]
S4(1) =15 or S4(1) =16
No
Si
[s4] = [s4(7) , …, s4(M1)]
[s4] = [s4(1) , …, s4(M1)]
5
2
Convertir el string {s} a un arreglo de bytes [s].
M= cuenta de bytes del arreglo [s].
10


10
15
K = 0, M1
Si
M1<1
No
i = 0, M1
W[k,j]
TABLA
No
s3(i) = Dirección
del esclavo
Si
9
[s4] = [s(i) , …… , s(M1)]
Extraer los 8 primeros bytes del arreglo [s4] que pueden ser el mensaje consulta para las funciones 1-2-3-4-5-6, o el mensaje rta de las funciones 5-6-15-16:
Dirección del esclavo s4(0)
Función s4(1)
(cuenta de bytes) s4(2)
s4(3)
s4(4)
s4(5)
(cuenta de bytes) s4(6)
s4(7)
Agrupar s4(6) y s4(7) en una palabra = CRC1
Cálculo del CRC a los 7 primeros bytes del arreglo [s4] = CRC1.
CRC1 = CRC2
Si
No
s4(2) podría ser la cuenta de bytes del mensaje rta de las funciones 1-2-3-4
C = s4(2) + 3
D= C+1
Extraer y agrupar s4(C) y s4(C+1) en una palabra = CRC4
Cálculo del CRC a los C primeros bytes del arreglo [s4] = CRC3.
CRC3=CRC4
Si
No
[s4] = [s4(0),…, s4(D)]
14
s4(6) podría ser la cuenta de bytes del mensaje consulta de las funciones 15-16
Q = s4(6) + 7
11
14
W[k,j]= [s5]
[s3] = [s4]
M1= cuenta de bytes del arreglo [s4]
15
i = 1, 4
No
12
S4(1) = i
Si
Extrae del arreglo [s4] los bytes que corresponden a:
dir. Esclavo (rta) s4(8)
Función (rta) s4(9)
Cuenta de bytes s4(10)
L = 8 + 3 + s4(10)
Extraer y agrupar s4(L) y s4(L+1) en una palabra = CRC7
13


11
Extraer y agrupar s4(Q) y s4(Q+1) en una palabra = CRC5
Cálculo del CRC a los Q primeros bytes del arreglo [s4] = CRC6.
CRC5=CRC6
No
Si
[s4] = [s4(3),…, s4(M1)]
14
Extrae del arreglo [s4] los bytes que corresponden a:
dir. Esclavo (rta) s4(Q+2)
Función (rta) s4(Q+3)
CRC Hi s4(Q+8)
CRC Li s4(Q+9)
Agrupar s4(Q+8) y s4(Q+9) en una palabra = CRC7
Cálculo del CRC a los bytes del arreglo [s4(Q+2), ……, s4(Q+7)] = CRC8.
CRC5=CRC6
No
Si
[s5] = [s4(0) , … , s4(Q+9)]
j= Q+10
[s4] = [s4(Q+10), …, s4(M1)]
[s4] = [s4(Q+10), …, s4(M1)]
14
13
Cálculo del CRC a los bytes del arreglo [s4(8), ……, s4(L-1)] = LRC8.
CRC7=CRC8
No
Si
[s4] = [s4(10) , … , s4(M1)]
s5] = [s4(0) , … , s4(L+1)]
j= L+2
[s4] = [s4(L+2) , …, s4(M1)]
14
12
S4(1) = 5 or S4(1) = 6
No
Si
[V1] = [s4(0) , … , s4(7)]
[V2] = [s4(8) , …, s4(15)]
Si
[V1]= [V2]
No
Si
[s5] = [s4(0), … , s4(15)]
j= 16
[s4] = [s4(16), …, s4(M1)]
[s4] = [s4(8) ,…, s4(M1)]
S4(1) =15 or S4(1) =16
No
Si
[s4] = [s4(8) , …, s4(M1)]
[s4] = [s4(1) , …, s4(M1)]
14

6. APLICACIÓN

En este capítulo se presenta como aplicación el proceso de llenado, envasado y taponado en una fábrica; la aplicación se desarrolla con el fin de mostrar la intercomunicación de los PLC's y el PC utilizando el Protocolo Modbus y la técnica Maestro-esclavo.

El proceso fue tomado de dos referencias bibliográficas: Balcells[2] y Monografía de Redes Industriales Basadas en Profibus[3]. A continuación se presenta la descripción del proceso:

En el área de llenado se almacenan dos productos diferentes dentro de dos tanques (TANQUE 1 Y TANQUE 2), el producto llega a cada tanque por medio de dos tuberías denominadas Alimentación 1 y Alimentación 2 (para el TANQUE 1 y TANQUE 2 respectivamente). El paso de producto a los TANQUES 1 y 2 está restringido por las válvulas VE1 y VE2 respectivamente, los tanques se pueden llenar al tiempo siempre y cuando no estén siendo lavados. El producto se envía al área de envasado a través de una tubería por medio de la apertura de las válvulas VS1 y VS2, una condición del proceso es que cuando se abra la válvula VS1 la válvula VS2 debe permanecer cerrada y viceversa.
Para la limpieza se cuenta con tuberías que transportan agua, ésta ingresa a los TANQUES 1 y 2 en el momento en que las válvulas VEH1 y VEH2 son abiertas, dentro de cada tanque se encuentra un aspersor para dispersar el agua y enviarla a las paredes del tanque, luego de limpiar los tanques el agua es llevada a unos conductos de desagüe cuyo cierre y apertura está controlado por las válvulas VSR1 y VSR2. Los tanques se pueden lavar al tiempo.

El producto llega al área de envasado y se deposita en el Dosificador, el dosificador tiene que llenarse por encima del nivel inferior para poder llenar las botellas.

[2] BALCELLS, Joseph; ROMERAL, José. Autómatas Programables, Mexico D.F. Primera Edición.1998. p.37-40

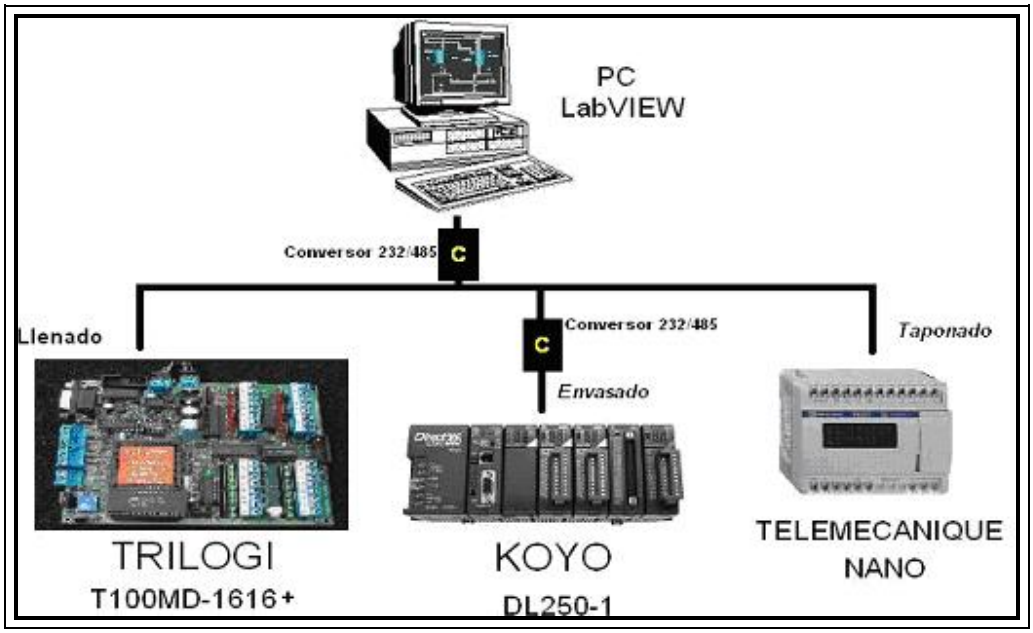
[3] PUERTO, Jhon; GOMEZ,David. Redes Industriales Basadas en Profibus. Volumen II. Bucaramanga. 1999. p302-334

Cuando existe una botella en el alimentador de botellas será detectada por el sensor SP1 y el cilindro 1 (CIL.1) la enviará a la BANDA 1, una vez en la BANDA 1 se activa el motor de la banda y la botella se desplaza hasta la posición de llenado (debajo de la válvula llenado de botellas VLLB) activando el sensor SP2, en este momento se deteniene el motor de la banda y se abre la válvula VLLB del dosificador por un tiempo hasta que la botella se llene totalmente.
En el momento en que la botella esté llena seguirá desplazándose por la BANDA 1 hasta llegar al final de la misma, donde es detectada por el sensor SP3 y es colocada en la BANDA 2 por la acción del cilindro 2 (CIL.2), la BANDA 2 conduce las botellas al área de taponado.
El dosificador para su limpieza está provisto con un aspersor que está conectado a las tuberías del agua por intermedio de la válvula VEHD, cuando ésta se abre el agua entra al dosificador y es dispersada por la acción del aspersor lavando el tanque, el agua residual se envía por un desagüe al abrir la válvula VSRD.

La sección de taponado tiene un dosificador de tapas, cuando una tapa llega al final del dosificador el sensor SP5 la detectará permitiendo que el cilindro 4 (CIL.4) la desplace hasta la posición de taponado (debajo del cilindro 3), una vez allí la fotocelda FOT1 indicará que la tapa está lista.
En el momento en que la botella que se desplaza por la BANDA 2 llegue a la posición de taponado, el sensor SP4 se activará y detendrá la BANDA 2. Al haber una botella en la posición de taponado y una tapa lista, el cilindro 3 (CIL.3) ejercerá una presión, para tapar la botella, que será determinada por el sensor SPR1.

La interfaz gráfica del proceso se diseñó en LabVIEW, en ésta, se convierten las variables del proceso en animaciones; para ello se configuró el PC como maestro. El proceso de llenado se programó en el PLC TRILOGI, el proceso de envasado en el PLC KOYO y el proceso de taponado en el PLC NANO TELEMECANIQUE, ver Figura 96.

Figura 96. Topología de la red física empleada

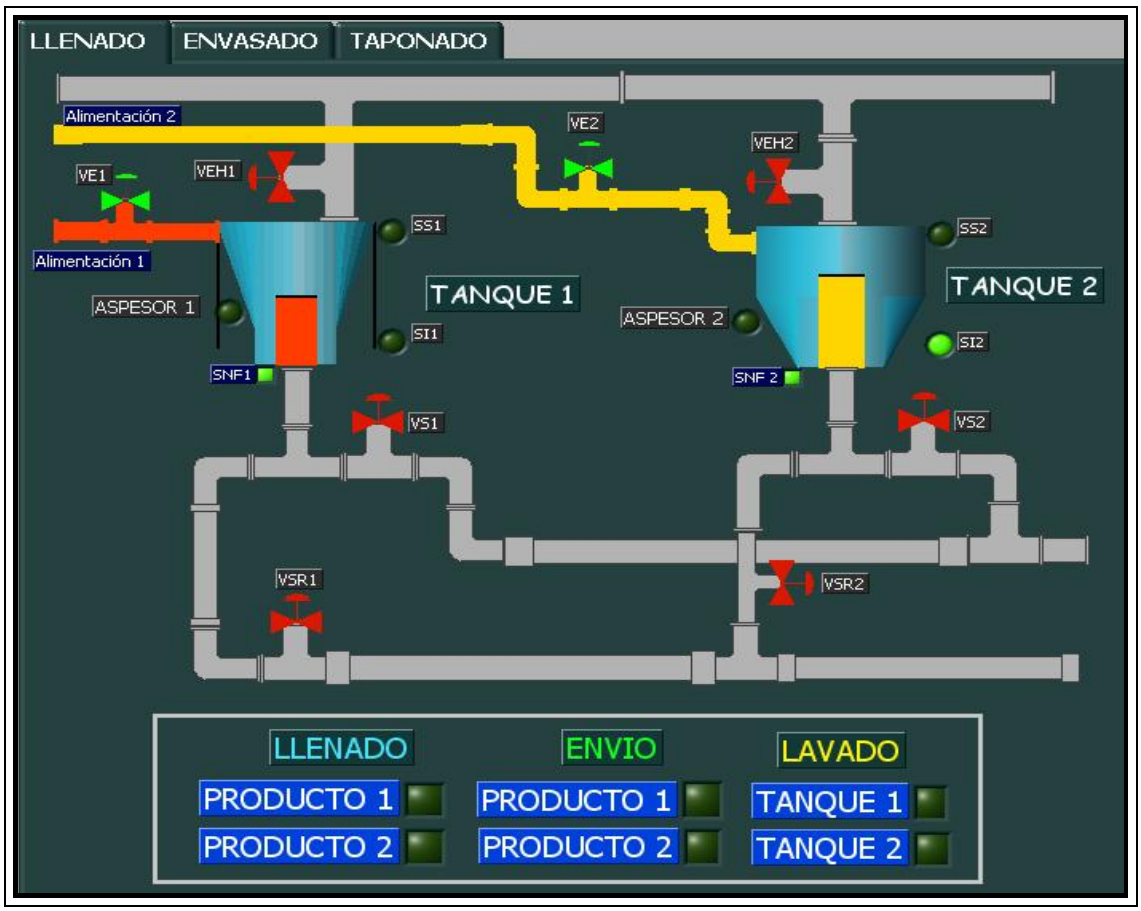


A continuación se presentan las variables de cada sección del proceso con su respectiva interfaz gráfica y la configuración de cada PLC para dicho propósito.

6.1 SISTEMA DE LLENADO DE TANQUES

En la Figura 97 se presenta la interfaz gráfica correspondiente al proceso de llenado de tanques.

Figura 97. Panel frontal en LabVIEW correspondiente al sistema de llenado de tanques



En LabVIEW cuando hay producto fluyendo por alguna tubería cambia de color, además las válvulas se ponen en verde cuando son abiertas.

En el sistema de llenado se tienen las siguientes variables:

Sensores: La etiqueta de los sensores en LabVIEW es de color azul, observe la Figura 97.

- **ALIMENTACIÓN 1 Y ALIMENTACIÓN 2**: Estos sensores indican la presencia de producto en las tuberías de entrada de cada uno de los tanques, son del tipo

normalmente abierto. Se representan en LabVIEW por un cambio de color en la tubería de entrada de producto de los tanques.

- **SNF1 y SNF2:** Sensores de nivel de fondo, son normalmente abiertos y cambian su estado cuando exista producto o agua en el fondo del tanque.

*Actuadores***:** La etiqueta de los actuadores en LabVIEW es de color gris.

- **VE1 y VE2**: Válvulas de entrada de producto al TANQUE 1 y TANQUE 2 respectivamente.
- **VS1 Y VS2**: Válvulas de salida del producto del TANQUE 1 y TANQUE 2 respectivamente.
- **VEH1 Y VEH2:** Válvulas de entrada de agua al TANQUE 1 y TANQUE 2 respectivamente.
- **ASPERSOR 1 y ASPERSOR 2:** Cada tanque tiene un aspersor que dispersa el agua hacia las paredes del tanque.
- **VSR1 Y VSR2:** Válvulas de salida de agua residual del TANQUE 1 y TANQUE 2 respectivamente.

Pulsadores:

- **ARRANQUE:** Pulsador arranque de almacenado.
- **EMERGENCIA:** Pulsador parada de emergencia.
- **TANQUE 1:** Pulsador llenado del tanque 1.
- **LAVADO 1:** Pulsador lavado del tanque 1.
- **TANQUE 2:** Pulsador llenado del tanque 2.
- **LAVADO 2:** Pulsador lavado del tanque 2.

Led's:

- **Led's SI1 Y SI2:** Señalización que indica que se ha alcanzado el nivel inferior del TANQUE 1 y TANQUE 2 respectivamente
- **Led's SS1 Y SS2:** Señalización que indica que se ha alcanzado el nivel superior del TANQUE 1 y TANQUE 2 respectivamente.

Llenado:

- **Led PRODUCTO 1 y Led PRODUCTO 2*:*** Señalización de la apertura de las válvulas VE1 y VE2 respectivamente en el panel frontal de LabVIEW.

Envío:

- **Led PRODUCTO 1 y Led PRODUCTO 2*:*** Señalización de la apertura de las válvulas VS1 y VS2 respectivamente en el panel frontal de LabVIEW.

Lavado:

- **Led TANQUE 1 y TANQUE 2**: Señalización del lavado de cada uno de los tanques, en el panel frontal de LabVIEW.

Emergencia:

- **Led EMERGENCIA ALMACENADO: S**eñalización que indica que ha ocurrido una parada de emergencia en el proceso de almacenado.

6.1.1 Características de los PLC's TRILOGI

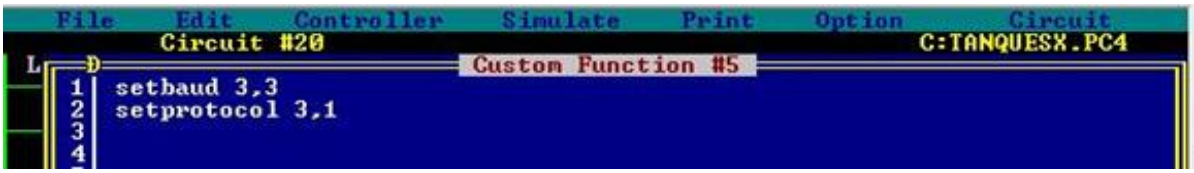
El proceso de llenado de tanques fue programado en el PLC TRILOGI, algunas características de éste son:

- 16 Entradas digitales (24 VDC)
- 16 Salidas digitales (1A y 24VDC)
- 4 Canales de 10 bits (entradas análogas)
- 2 Canales de 8 bits (salidas analógicas)
- **Puerto 1**: RS232 Comunicación PC (conector DB-9).
- **Puerto 3** :RS485 Comunicación PLC's (half duplex)
  - ASCII
  - **Modbus RTU**
  - **Modbus ASCII**
  - OMRON C20H

Configuración del PLC TRILOGI

El puerto 3 emplea el estándar RS485; para configurar los parámetros *Baud Rate*, paridad, bits de datos, bit de parada y protocolo de comunicación se programó una rutina *Custom Function,* ver Figura 98.

Figura 98. Rutina de configuración del puerto de comunicación 3



A continuación se presentan las características de la función ***Setbaud***:

Setbaud *Ch, Baud_No*	***Ch:*** Puerto de comunicaciones (1 ó 3).
	Baud_No: Toma un valor entre 0-255 (00h-FFh), con el cual se configuran los siguientes parámetros *Baud Rate*, bit de datos, bit de parada y paridad.

Setbaud 3, 3 Configura el puerto 3 a 9600 bits/seg, con un bit de parada, 8 bits de datos y sin verificar paridad, ver Figura 99.

Figura 99. Contiene los valores para configurar el parámetro *Baud_No*.

Baud \ Bits de datos, Bit de Parada, Paridad	8,1,n	8,1,e	8,1,o	7,1,n	7,1,e	7,1,o	8,2,n	8,2,e	8,2,o	7,2,n	7,2,e	7,2,o
110	0B	4B	6B	8B	CB	EB	1B	5B	7B	9B	DB	FB
150	0C	4C	6C	8C	CC	EC	1C	5C	7C	9C	DC	FC
300	0D	4D	6D	8D	CD	ED	1D	5D	7D	9D	DD	FD
600	0E	4E	6E	8E	CE	EE	1E	5E	7E	9E	DE	FE
1200	0F	4F	6F	8F	CF	EF	1F	5F	7F	9F	DF	FF
2400	01	41	61	81	C1	E1	11	51	71	91	D1	F1
4800	02	42	62	82	C2	E2	12	52	72	92	D2	F2
9600	03	43	63	83	C3	E3	13	53	73	93	D3	F3
19200	04	44	64	84	C4	E4	14	54	74	94	D4	F4
31250	05	45	65	85	C5	E5	15	55	75	95	D5	F5
38400	06	46	66	86	C6	E6	16	56	76	96	D6	F6
62500	07	47	67	87	C7	E7	17	57	77	97	D7	F7
100K	08	48	68	88	C8	E8	18	58	78	98	D8	F8
250K	09	49	69	89	C9	E9	19	59	79	99	D9	F9
500K	0A	4A	6A	8A	CA	EA	1A	5A	7A	9A	DA	FA

(Todos los números en Hexadecimal: &H00 to &HFF)

A continuación se presentan las características de la función ***Setprotocol*:**

Setprotocol ***Ch, Mode***	***Ch:*** Puerto de comunicaciones (1 ó 3)
	Mode: Protocolo de comunicación. 0- Auto sensado **1- Modo Modbus RTU** 2- Modo EMIT **3- Modo Modbus ASCII** 4- Protocolo Modo OMRON C20H 5- Modo comando Link host NATIVE

ASIGNACIÓN DE VARIABLES

***Entradas Digitales*:** En la Tabla 20 se encuentran los símbolos asignados a cada entrada discreta del PLC TRILOGI, además se encuentra la dirección Modbus que aparece en el mapa de memoria (memoria discreta) y la dirección equivalente en LabVIEW para acceder a la memoria donde se encuentra el estado de cada símbolo.

Tabla 20. Asignación de símbolos a las entradas discretas del PLC TRILOGI

Símbolo	Entradas "TRILOGI"	Dirección (memoria discreta)	
		MODBUS (dec)	LABVIEW (dec)
ARRANQUE	I 0	1	0
EMERGENCIA	I 1	2	1
TANQUE 1	I 2	3	2
ALIMENTACION 1	I 3	4	3
SNF1	I 4	5	4
LAVADO 1	I 5	6	5
TANQUE 2	I 6	7	6
ALIMENTACION 2	I 7	8	7
SNF2	I 8	9	8
LAVADO 2	I 9	10	9

Salidas Digitales: En la Tabla 21 se encuentran los símbolos asignados a cada salida discreta del PLC TRILOGI con la dirección Modbus que se encuentra en el Mapa de memoria de estos y la dirección equivalente para LabVIEW.

Tabla 21.Asignación de símbolos a las salidas discretas del PLC TRILOGI

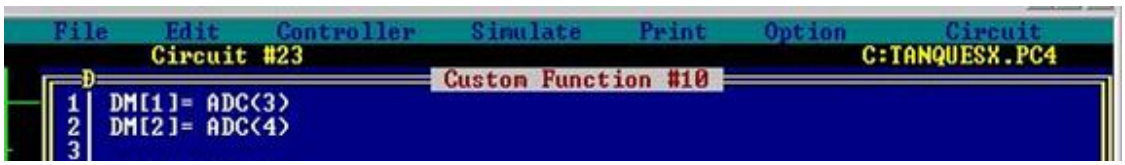
Símbolo	Salidas "TRILOGI"	Dirección (memoria discreta)	
		MODBUS(dec)	LabVIEW(dec)
VE1	Out 0	257	256
VS1	Out 1	258	257
VEH1	Out 2	259	258
VSR1	Out 3	260	259
ASPERSOR1	Out 4	261	260
VE2	Out 5	262	261
VS2	Out 6	263	262
VEH2	Out 7	264	263
VSR2	Out 8	265	264
ASPERSOR2	Out 9	266	265
LED EMERGENCIA	Out 10	267	266
SI1	Out 11	268	267
SS1	Out 12	269	268
SI2	Out 13	270	269
SS2	Out 14	271	270

Entradas Analógicas: Para simular el aumento de nivel en el TANQUE 1 y 2 se utilizaron 2 potenciómetros lineales, uno para cada tanque. En la Figura 100 se encuentra la rutina *Custom Function* en la que se asigna dos registros (DM[1] Y DM[2]) de la memoria de datos del PLC para guardar los valores de los canales analógicos 3 y 4.

A continuación se presentan la dirección Modbus y la dirección equivalente en LabVIEW de los registros:

	Modbus (dec)	LabVIEW (dec)
DM[1] …….	41001 ……..	1000
DM[2] …….	41002 ……..…	1001

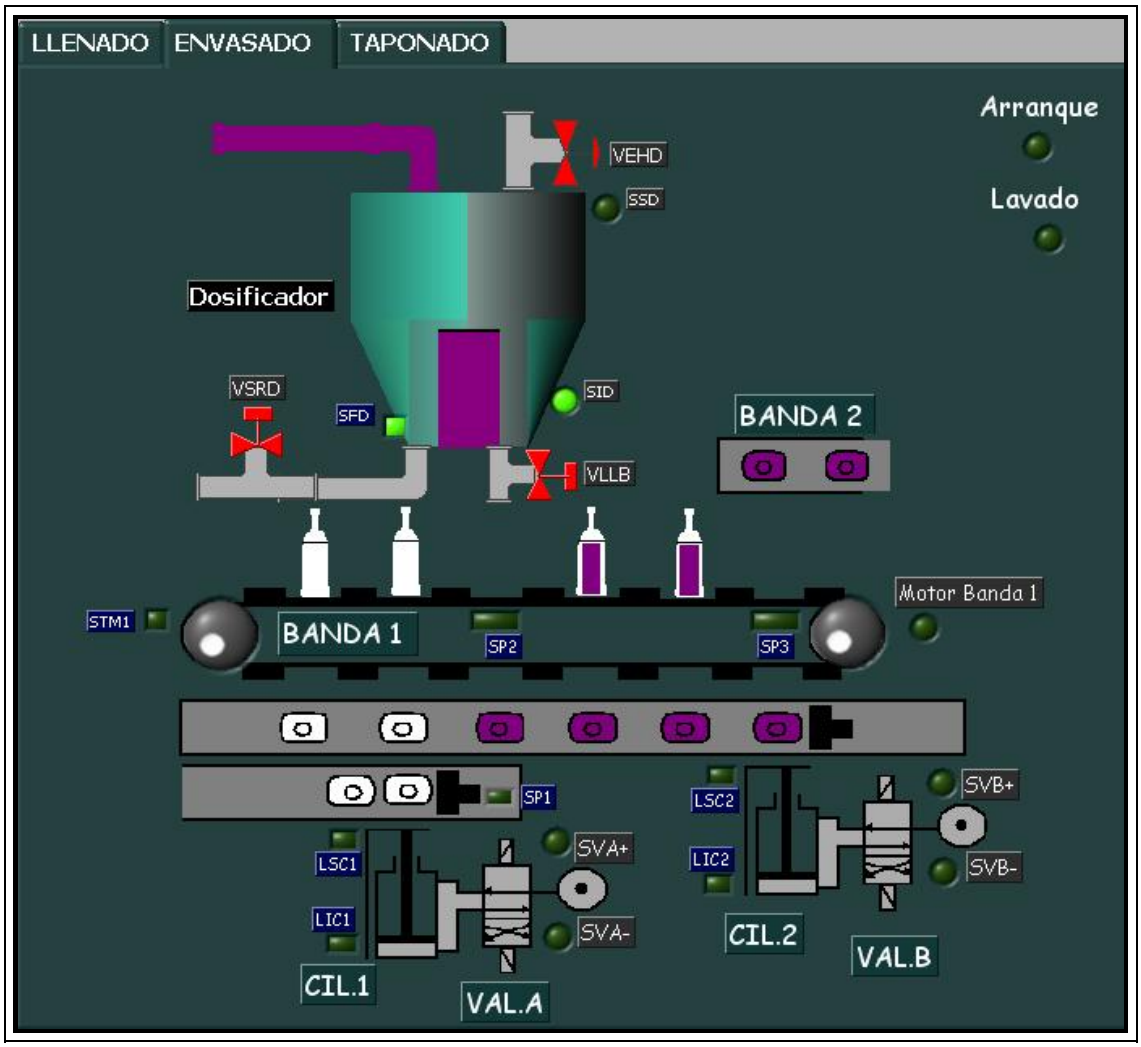
Figura 100. Rutina de asignación de registros a las entradas analógicas



6.2 SISTEMA DE ENVASADO DE PRODUCTO

En la Figura 101 se puede observar la interfaz gráfica correspondiente al proceso de envasado.

Figura 101. Panel frontal en LabVIEW correspondiente al sistema de envasado



En el sistema de envasado se tienen las siguientes variables:

Sensores:

- **SFD:** Sensor de nivel de fondo, es normalmente abierto y cambia su estado cuando existe producto o agua en el fondo del tanque.

- **STM1:** Sensor térmico del motor de la banda 1, es normalmente abierto y se cerrará cuando el motor exceda su límite de temperatura.
- **SP1:** Sensor de posición 1, es normalmente abierto y se cierra cuando exista una botella para ser enviada a la banda 1.
- **SP2:** Sensor de posición 2, es normalmente abierto y se cierra cuando la botella llega a la posición de llenado.
- **SP3:** Sensor de posición 3, es normalmente abierto y se cierra cuando la botella esté llena y lista para ser enviada a la banda 2.
- **LIC1:** Posición inferior del cilindro 1, es normalmente abierto y se cierra cuando el cilindro 1 llega a su límite inferior.
- **LSC1:** Posición superior del cilindro 1, es normalmente abierto y se cierra cuando el cilindro 1 llega a su límite superior.
- **LIC2:** Posición inferior del cilindro 2, es normalmente abierto y se cierra cuando el cilindro 2 llega a su límite inferior.
- **LSC2:** Posición superior del cilindro 2, es normalmente abierto y se cierra cuando el cilindro 2 llega a su límite superior.

*Actuadores***:**

- **VLLB:** Válvula de llenado de botellas.
- **VEHD:** Válvula de entrada de agua al dosificador.
- **ASPERSORD:** Dispersa el agua hacia las paredes del tanque.
- **VSRD:** Válvula de salida de agua residual.
- **MB1:** Motor de la banda 1.
- **SVA+:** Válvula solenoide A+, esta acción se realiza cuando el cilindro 1 envía una botella hacia la banda 1.
- **SVA-:** Válvula solenoide A-, esta acción se realiza cuando el cilindro 1 ha colocado una botella en la banda 1y retorna a su posición inicial.
- **SVB+:** Válvula solenoide B+, esta acción se realiza cuando el cilindro 2 envía una botella hacia la banda 2.

- **SVB-:** Válvula solenoide B-, esta acción se realiza cuando el cilindro 2 ha colocado una botella en la banda 2 y retorna a su posición inicial.

Pulsadores:

- **ARRANQUE:** Pulsador arranque del sistema embotellado.
- **EMERGENCIA:** Pulsador parada de emergencia.
- **LAVADO:** Pulsador lavado del dosificador.

Led's:

- **Led SID:** Señalización que indica que se ha alcanzado el nivel inferior del dosificador.
- **Led SSD:** Señalización que indica que se ha alcanzado el nivel superior del dosificador.
- **Led EMERGENCIA EMBOTELLADO:** señalización que indica que ha ocurrido una parada de emergencia en el proceso de embotellado.

6.2.1 Características de los PLC's koyo

- Módulo **D2–16ND3-216:**

  16 Entradas DC (24 VDC a 6mA)

- Módulo **D2–12TR**:

  12 Salidas Relé (1A y 24VDC)

- **Puerto 1**: RS232 Comunicación PC a 9600 baudios (RJ11 conector telefónico)
  - K-sequence
  - Esclavo *Direct*NET
  - **Esclavo MODBUS RTU**
  - *Direct*SOFT32 (software de programación)

- **Puerto 2** : **RS232C/RS422** hasta 38.4 k Baudios, comunicación PLC's (conector 15-pines)
  - K-sequence
  - *Direct*NET Maestro/Esclavo
  - **MODBUS RTU Maestro/Esclavo**
  - *Direct*SOFT32

Configuración del PLC KOYO

El puerto 2 emplea el estándar RS232 ó RS422, para poder incluir este PLC a la red se construyó un conversor RS232/RS485. En la Figura 102 se presenta la configuración del puerto de comunicación 2, para ingresar a esta ventana se hace a través del menú PLC, SETUP, Secondary Comm Port.

Figura 102. Configuración del puerto de comunicación 2 en un KOYO

ASIGNACIÓN DE VARIABLES

***ENTRADAS*:** En la Tabla 22 se encuentran los símbolos asignados a cada entrada discreta del PLC KOYO con su respectiva dirección Modbus.

Tabla 22.Asignación de símbolos a las entradas discretas del PLC KOYO

Símbolo	Entrada KOYO	Dirección Modbus (decimal)
ARRANQUE	X0	2048
EMERGENCIA	X1	2049
SFD	X2	2050
LAVADO	X3	2051
LSCI	X4	2052
LICI	X5	2053
LSC2	X6	2054
LIC2	X7	2055
SP1	X10	2056
SP2	X11	2057
SP3	X12	2058
STM1	X13	2059

SALIDAS: En la Tabla 23 se encuentran los símbolos asignados a cada salida discreta del PLC KOYO con su respectiva dirección Modbus.

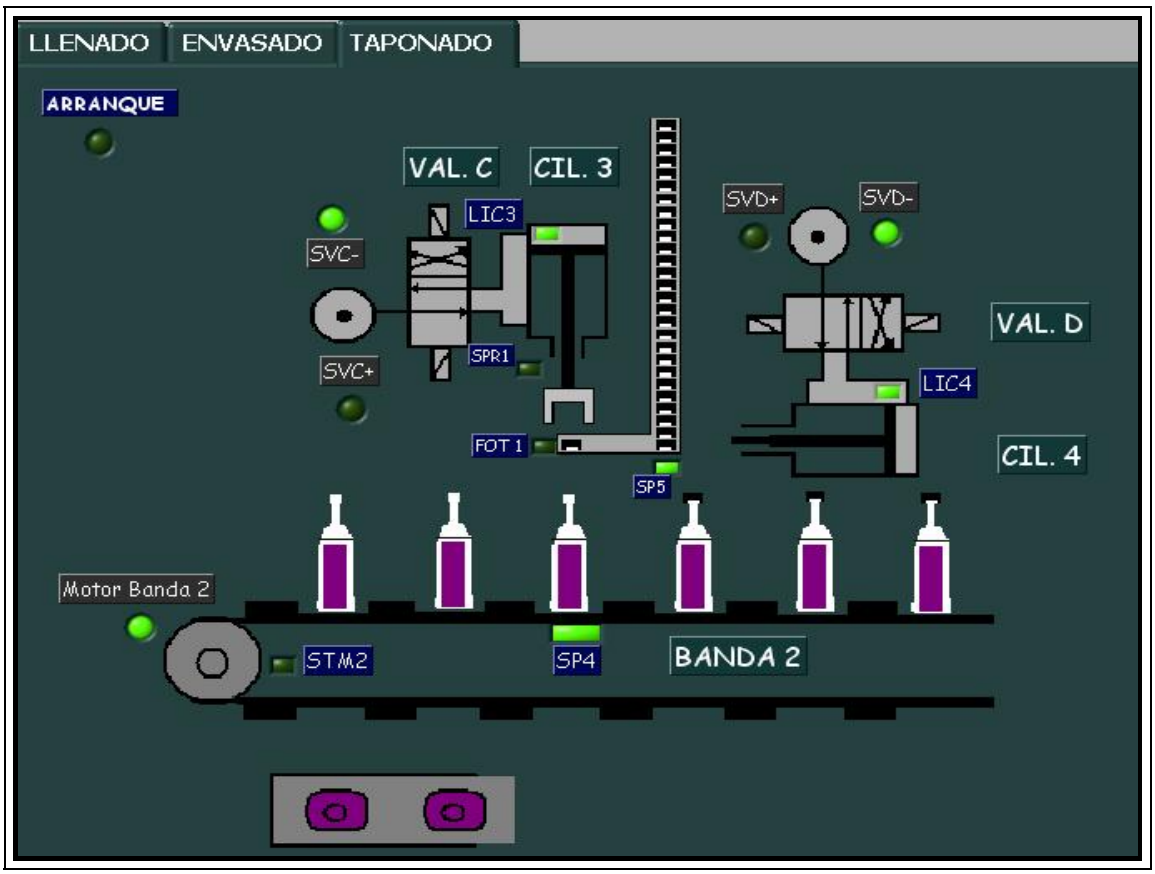
Tabla 23.Asignación de símbolos a las salidas discretas del PLC KOYO

Símbolo	Salida KOYO	Dirección MODBUS (decimal)
MB1	Y0	2048
SVA+	Y1	2049
SVA-	Y2	2050
SVB+	Y3	2051
SVB-	Y4	2052
VLLB	Y5	2053
VEHD	Y10	2056
VSRD	Y11	2057
ASPERSORD	Y12	2058
LUZ	Y13	2059
SID	Y14	2060
SSD	Y15	2061

6.3 SISTEMA DE TAPONADO DE BOTELLAS

En la Figura 103 se puede apreciar la interfaz gráfica correspondiente al proceso de Taponado.

Figura 103. Panel frontal en LabVIEW correspondiente al sistema taponado



En el sistema de taponado se tienen las siguientes variables:

Sensores:

- **STM2:** Sensor térmico del motor de la banda 2, es normalmente abierto y se cerrará cuando el motor exceda su límite de temperatura.
- **SP4:** Sensor de posición 4, es normalmente abierto y se cierra cuando la botella llega a la posición de taponado.

- **SP5:** Sensor de posición 5, es normalmente abierto y se cierra cuando una tapa está lista para ser enviada a la posición de taponado.
- **FOT1:** Fotocelda que indica el instante en que llega una tapa a la posición de taponado.
- **LIC3:** Posición inicial del cilindro 3, es normalmente abierto y se cierra cuando el cilindro 3 llega a su límite inferior.
- **LIC4:** Posición inicial del cilindro 4, es normalmente abierto y se cierra cuando el cilindro 4 llega a su límite inferior.
- **SPR1:** Sensor de presión, es normalmente abierto y se cierra cuando la presión de taponado llegue al máximo.

*Actuadores***:**

- **MB2:** Motor de la banda 2.
- **SVD+:** Válvula solenoide D+, esta acción se realiza cuando el cilindro 4 envía una tapa hacia la posición de taponado.
- **SVD-:** Válvula solenoide D-, esta acción se realiza cuando el cilindro 4 ha colocado una tapa en la posición de taponado y retorna a su posición inicial.
- **SVC+:** Válvula solenoide C+, esta acción se realiza cuando el cilindro 3 se desplaza para colocar una tapa sobre la botella.
- **SVC-:** Válvula solenoide C-, esta acción se realiza cuando el cilindro 3 ha tapado una botella y regresa a su posición inicial.

Pulsadores:

- **ARRANQUE:** Pulsador arranque del sistema taponado.
- **EMERGENCIA:** Pulsador parada de emergencia.

Led's:

- **Led EMERGENCIA TAPONADO:** señalización que indica que ha ocurrido una parada de emergencia en el proceso de taponado.

ASIGNACIÓN DE VARIABLES

***ENTRADAS*:** En la Tabla 24 se encuentran los símbolos y los bits internos de trabajo asignados a cada entrada discreta del PLC NANO TELEMECANIQUE con su respectiva dirección Modbus.

Tabla 24.Asignación de símbolos a las entradas discretas del PLC NANO TELEMECANIQUE

Símbolo	Entradas NANO	Bit Interno Asignada	Dirección Modbus (decimal)
RUN	I0.0		
ARRANQUE	I0.1	M30	30
EMERGENCIA	I0.2	M31	31
SP4	I0.3	M32	32
STM2	I0.4	M33	33
LIC4	I0.5	M34	34
SP5	I0.6	M35	35
FOT1	I0.7	M36	36
SPR1	I0.8	M37	37
LIC3	I0.9	M38	38

SALIDAS: En la Tabla 25 se encuentran los símbolos y los bits internos de trabajo asignados a cada salida discreta del PLC NANO TELEMECANIQUE con su respectiva dirección Modbus.

Tabla 25.Asignación de símbolos a las salidas discretas del PLC NANO TELEMECANIQUE

Símbolo	Salidas NANO	Marca Asignada	Dirección MODBUS
MB2	Q0.0	M41	41
SVD+	Q0.1	M42	42
SVD-	Q0.2	M43	43
SVC+	Q0.3	M44	44
SVC-	Q0.4	M45	45
LUZ EMERGENCIA	Q0.5	M46	46

6.3.1 Características de los PLC's nano TELEMECANIQUE

- 14 Entradas VDC
- 10 Salidas VDC (24V a 150mA)
- **Puerto de programación:** RS485 (conector Mini Din 8).

  Protocolo: UNI-TELWAY Maestro

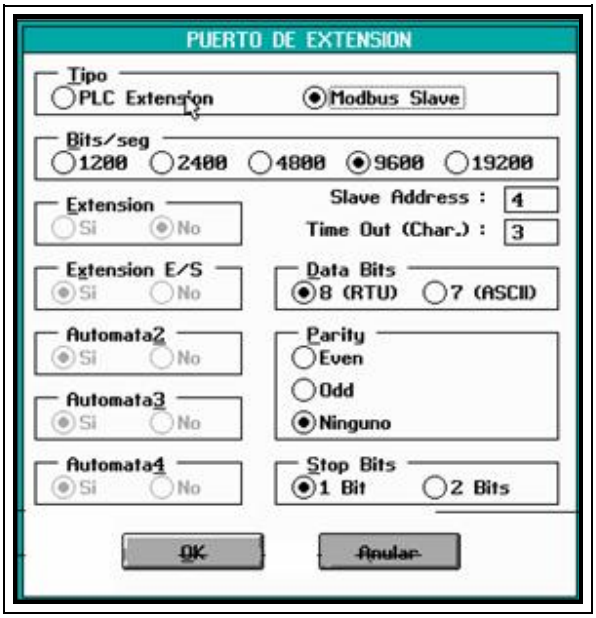
  UNI-TELWAY Esclavo

  ASCII
- **Puerto de extensión MODBUS/JBUS**: RS485 (longitud max: 1200 metros)
  - **Trama ASCII (7 bits)**
  - **Trama RTU (8 bits)**
  - 28 equipos esclavos máximo (direcciones lógicas 1-98)

Configuración del PLC NANO TELEMECANIQUE

En la Figura 104 se presenta la configuración del puerto de extensión de los PLC's NANO TELEMECANIQUE como esclavo, para ingresar a esta ventana se hace a través del menú Configuración y la opción puerto de extensión.

Figura 104. Configuración del puerto de extensión NANO TELEMECANIQUE.



6.4 MAESTRO DE LA RED "PC"

Luego de programar y configurar cada PLC, se procedió a construir la rutina de comunicación y de visualización en LabVIEW.

Los parámetros configurados en LabVIEW son:

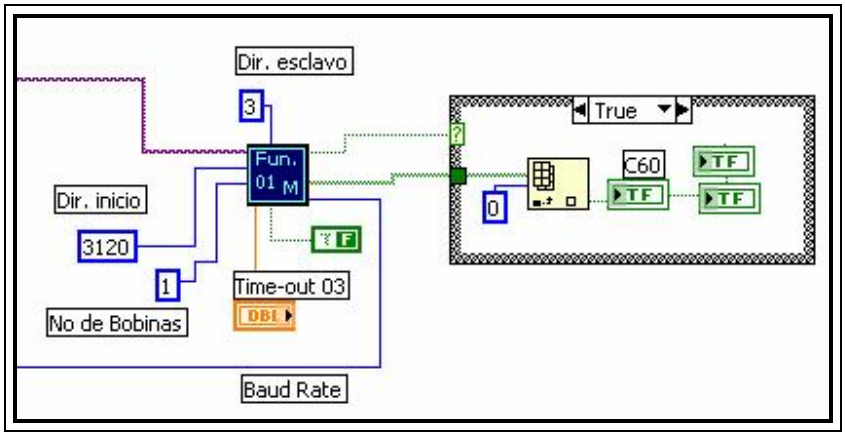
Baud Rate: 9600 bits/seg
Paridad: none
Bit de parada: 1
Bits de datos: 8 bits
Modo de transmisión: RTU (ya que las tres marcas de PLC's poseen este modo de transmisión).

Se construyó una librería Modbus (ver anexo B) para facilitar la construcción del programa Maestro.

Como la comunicación es serial en LabVIEW se contruyó una secuencia (con la Función *Secuence Structure*); a continuación se presenta la programación de cada secuencia con una breve explicación de la parte del proceso que realiza:

Secuencia 0:

Figura 105. Leer el relé C60 del PLC KOYO (Función 01)



En la secuencia 0 se pide leer el relé de control C60 (dirección 3120d) del PLC KOYO (Proceso de envasado) identificado con la dirección número 3, ver Figura 105.

Si el relé C60 esta en “1” significa que:

- El dosificador está vacío y limpio (inicialmente).
- Las válvulas de lavado del dosificador deben estar cerradas.

Si el relé C60 esta en “0” significa que:

- El dosificador se llenó y alcanzo el nivel superior, por lo tanto la válvula de salida de producto que este activada en el proceso de llenado se debe cerrar.
- Ocurre una parada de emergencia en el proceso de embotellado.

Secuencia 1:

En la secuencia 1 se forza el relé 2.0 (dirección 1040d) del PLC TRILOGI (Proceso de llenado de tanques, dirección del esclavo = 02), ver Figura 106.

Si el relé C60 esta en “1” se forza a “1” el relé 2.0 y ocurre que:

- Si hay producto suficiente en el tanque 1 (**SI1** en “1”), se activa la válvula de salida 1 (**VS1**).
- Si hay producto suficiente en el tanque 2 (**SI2** en “1”), se activa la válvula de salida 2 (**VS2**).

Si se activa una válvula de salida la otra no, ya que es una condición del proceso.

Si el relé C60 esta en “0” se forza a “0” el relé 2.0 y ocurre que:

- Si está activada la válvula de salida 1 (**VS1**), se cierra.
- Si está activada la válvula de salida 2 (**VS2**), se cierra.

Figura 106. Forzar el relé 2.0 del PLC TRILOGI (Función 05).

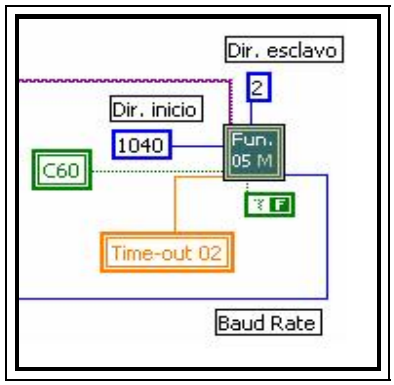
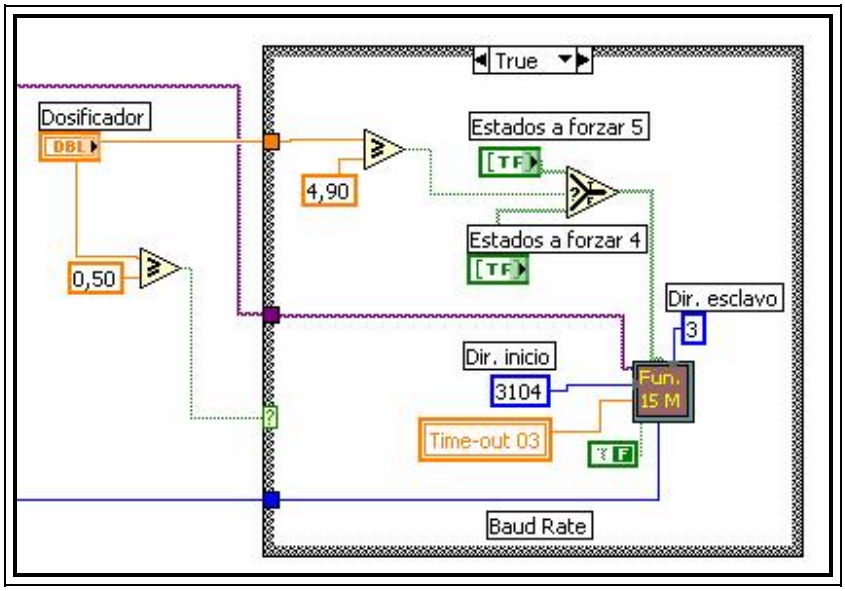


Figura 107. Forzar los relés C40 y C41 del PLC KOYO (Función 15)



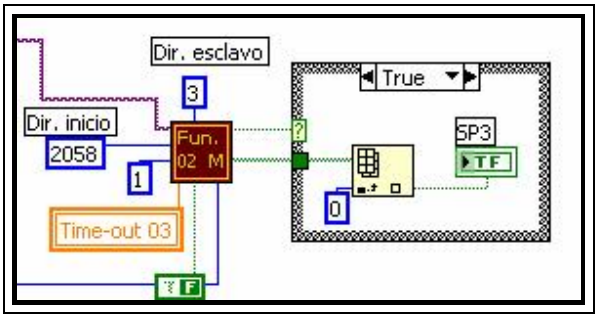
Secuencia 2:

En la secuencia 2 se simula el nivel del dosificador con el control:  , en esta secuencia se envía información del nivel del tanque al PLC KOYO, ver la Figura 107.

El rango del tanque varía de 0-5, si el nivel es mayor que 4,9 se forzan (a "1") los relés **C40** con dirección 3104d (forzando en la lógica del PLC la salida correspondiente al nivel inferior del dosificador **SID**) y **C41** con dirección 3105d (forzando en la lógica del PLC la salida correspondiente al nivel superior del dosificador **SSD**). Si el nivel es mayor de 0,5 y menor que 4,9 se forzan los relés C40 (a "1") y C41 (a "0").

Secuencia 3:

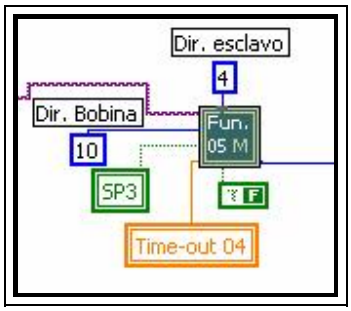
Figura 108. Leer entrada X12 del PLC KOYO (Función 02).



En la secuencia 3 se lee la entrada X12 (Sensor de posición 3 **SP3**) con dirección 2058d del PLC KOYO; esta entrada se activa cuando existe una botella para ser enviada a la banda 2, ver Figura 108.

Secuencia 4:

Figura 109. Forzar el bit interno M10 del PLC NANO, (Función 05)

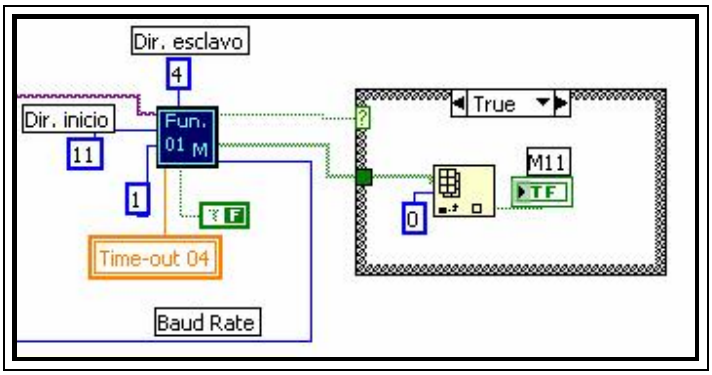


Si existe una botella para ser enviada a la banda 2 (**SP3** en "1"), se forza (a "1") el bit interno M10 del PLC NANO (Proceso de taponado, dirección de esclavo = 4) que en la lógica del PLC detiene el motor de la banda 2 (**MB2**), ver Figura 109.

Secuencia 5:

En la secuencia 5 se pide leer el bit interno M11 del PLC NANO, si está en "1" indica que el motor de la banda 2 está detenido y no hay parada de emergencia, ver Figura 110.

Figura 110. Leer el bit interno M11 del PLC NANO, (Función 01)



Secuencia 6:

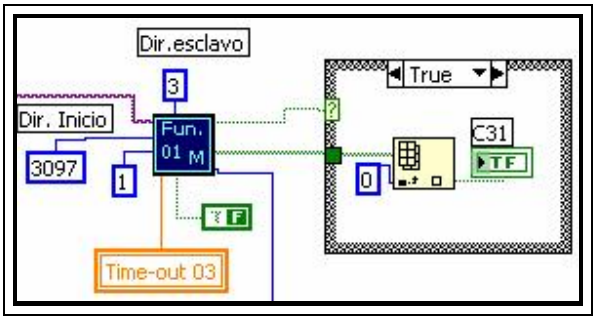
Si el bit interno M11 está en "1", se forza (a "1") el relé C62 (dirección 3122d) del PLC KOYO siendo esta una condición para activar la válvula solenoide B (encargada de empujar la botella de la banda 1 a la banda 2), ver Figura 111.

Figura 111. Forzar el relé C62 del PLC KOYO, (Función 15)

Secuencia 7:

En la secuencia 7 se pide leer el relé C31 (dirección 3097d) del PLC KOYO, si está activo indica que ya se envió la botella a la banda 2, ver Figura 112.

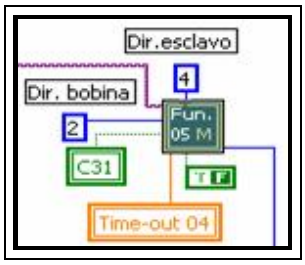
Figura 112. Leer el relé C31 del PLC KOYO (Función 01)



Secuencia 8:

Después de confirmar que la botella fue enviada a la banda 2, se forza (a "1") el bit interno M2 siendo una condición para arrancar el motor de la banda 2, ver Figura 113.

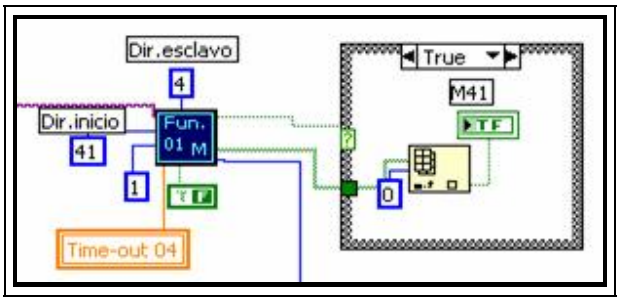
Figura 113. Forzar el bit interno M2 del PLC NANO (Función 5)



Secuencia 9:

Se lee el bit interno M41 del PLC NANO que corresponde a la salida motor banda 2 (**MB2**), ver Figura 114.

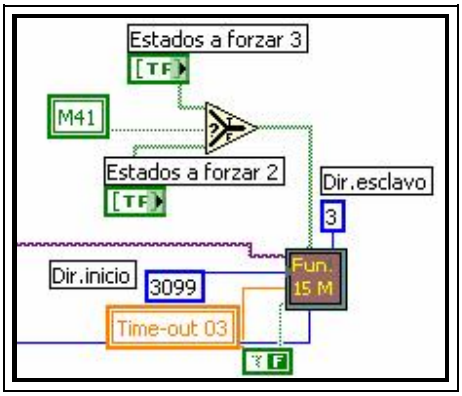
Figura 114. Leer el bit interno M41 del PLC NANO (Función 01)



Secuencia 10:

Si el bit interno M41 está en "1" (motor banda 2 en movimiento), se forza a "0" el relé C31 (dirección 3099d) que corresponde a la condición que confirma que ya se envió la botella a la banda 2, ver Figura 115.

Figura 115. Forzar el relé C31 del PLC KOYO (Función 15)



Secuencia 11:

En esta secuencia se tiene una estructura *Case Estructure* en la cual el usuario elige visualizar las variables de alguno de los tres PLC's.

Case 0:

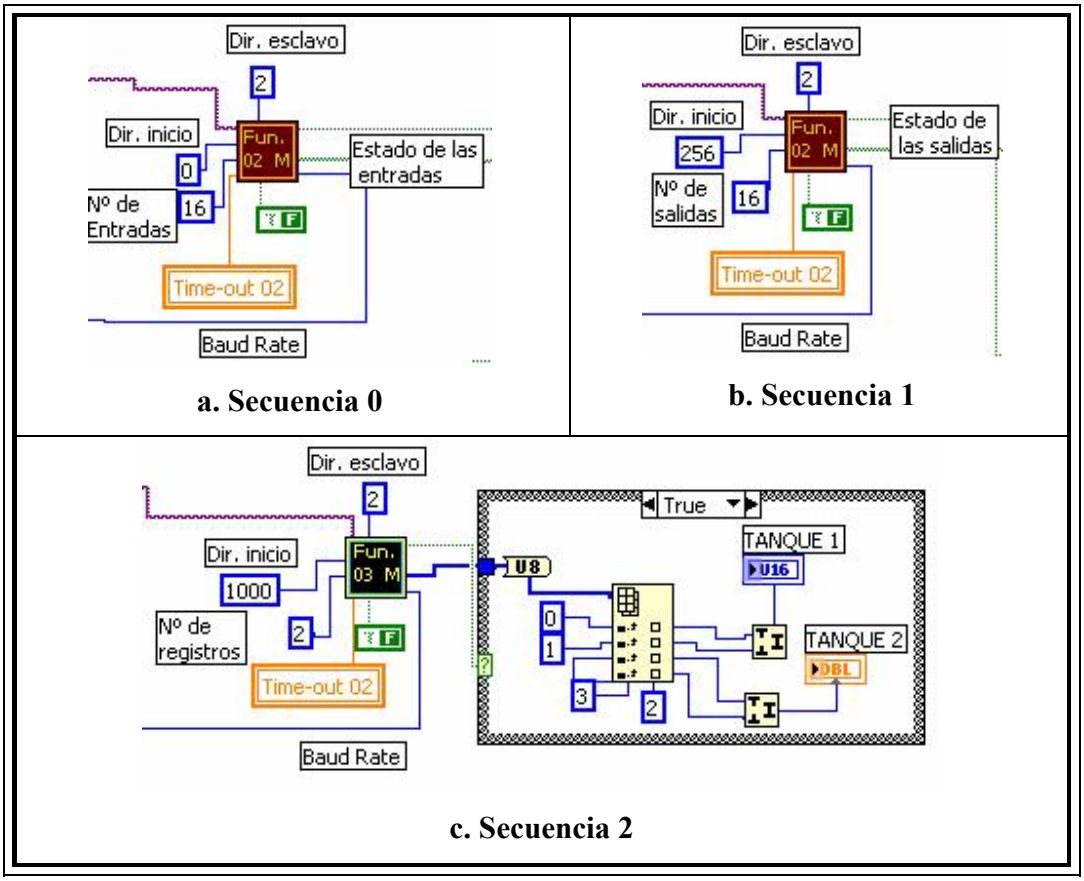
Este caso contiene una secuencia para leer las variables de entrada y de salida del PLC TRILOGI (Proceso de llenado de tanques), ver Figura 116.

Secuencia 0: Leer las entradas discretas del PLC TRILOGI, ver Figura116.a.

Secuencia 1: Leer las salidas discretas del PLC TRILOGI, ver Figura116.b.

Secuencia 2: Leer dos registros internos del PLC TRILOGI, ver Figura116.c.

Figura 116. Lectura de las entradas y salidas del PLC TRILOGI



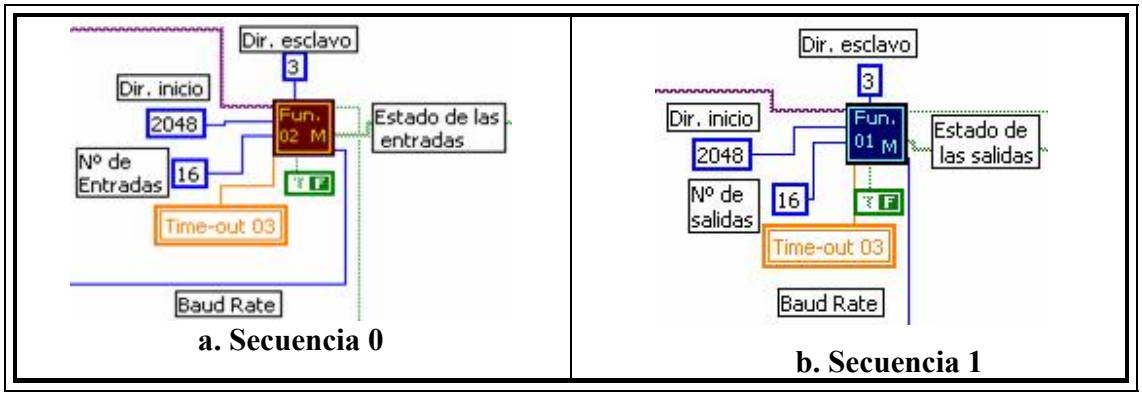
***Case 1*:**

Se encuentra una secuencia para leer las entradas y las salidas del PLC KOYO (Proceso de Envasado), ver Figura 117.

Secuencia 0: Leer las entradas discretas del PLC KOYO, ver Figura 117.a.

Secuencia 1: Leer las salidas discretas del PLC KOYO, ver Figura 117.b.

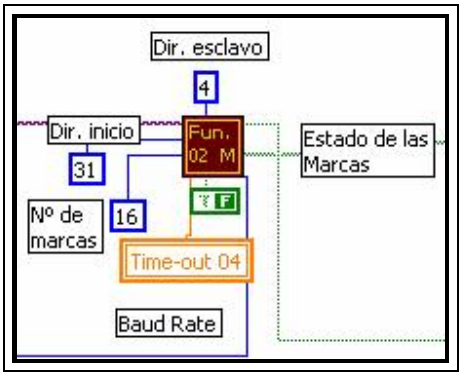
Figura 117. Lectura de las entradas y salidas del PLC KOYO



***Case 2*:**

Se leen los bits internos del PLC NANO que contienen el estado de las entradas y las salidas del Proceso de Taponado, ver Figura 118.

Figura 118. Lectura de las entradas y salidas del PLC NANO



7. CONCLUSIONES

- Se construyó un conversor RS-232/RS-485 para permitir la conexión del PC a la red MODBUS que emplea el estándar de comunicación serial RS 485. Dicho estándar utiliza una red con topología bus con comunicación *half-duplex* (transmisión y recepción por las mismas líneas) por tal razón se necesita una señal de habilitación para que el conversor adopte el sentido de transmisión o recepción. La habilitación se envía desde el programa en LabVIEW por la línea RTS del puerto RS-232 del PC, teniendo que: una señal de tensión negativa habilita al conversor en modo transmisión y una señal de tensión positiva lo habilita en modo recepción.

- En una red Modbus 485 se deben configurar con la misma información, en cada uno de los dispositivos conectados, los siguientes parámetros: Modo de transmisión (ASCII o RTU), Paridad, Velocidad de transmisión, bits de datos y bits de parada; para lograr que todos los dispositivos procesen correctamente las tramas Modbus que fluyen por la red. De lo contrario no se logró la comunicación.

- Se utilizó la función "VISA Configure Serial Port" del programa LabVIEW ya que esta configurar el puerto serial, agregando a cada carácter de la trama Modbus el bit de inicio, la paridad y los bits de parada; además empaqueta cada carácter de información (en 7 o 8 bits). Otro aspecto para elegirla es que permite controlar la línea RTS del puerto serie del PC.

- Se comprobo que una trama Modbus en modo ASCII tiene el doble de caracteres que en modo RTU (ya que cada byte del mensaje se transforma en dos caracteres ASCII) más un carácter de inicio que corresponde al símbolo ASCII ":", y dos caracteres finales que corresponden a los caracteres de control ASCII CR-LF; por tanto en modo ASCII se emplea mas del doble de tiempo transmitiendo las misma información que en modo RTU.

- Se configuró e implementó una red de comunicación industrial MODBUS (RS-485 y técnica Maestro/Esclavo) conformada por tres autómatas conectados como esclavos de diferentes fabricantes (TRILOGI, KOYO, TELEMECANIQUE de gama NANO) y el PC actuando como maestro.

- Se analizaron y comprobaron todas las peticiones MODBUS aceptadas por estos autómatas empleando un programa diseñado en LabVIEW (llamado "Maestro Modbus"), que construye las tramas del Mensaje Consulta del protocolo MODBUS para las funciones: 01 Leer Estado de las "Bobinas", 02 Leer Estado de las Entradas, 03 Leer Registros Internos, 04 Leer registros de Entrada, 05 Forzar una Bobina, 06 Fijar un Valor en un Registro, 07 Leer Estado de la Excepción, 08 Diagnóstico, 15 Forzar varias Bobinas, y 16 Fija varios Registros. Y seguidamente captura, procesa y visualiza las tramas respuesta enviadas por los autómatas esclavos.

- En los PLC's TELEMECANIQUE de la gama NANO no se pueden acceder directamente a los estados (ON/OFF) de los bits de entradas/salidas, bits de sistema, salida del temporizador, salida del contador,.. etc; ni los registros que contiene el valor de cuenta de los temporizadores, contadores,..etc. Consultando los manuales de estos PLC's se observa que solo permiten manipular los bits internos de trabajo %M y las palabras internas de trabajo %MW (propios de la programación); por lo tanto si se desea leer o forzar el estado de algún bit (ejemplo: entrada, salida) se le debe asignar un bit interno de trabajo %M en la programación del PLC; lo mismo sucede con los registros que se quieran leer o modificar, se les deben asignar una palabra de trabajo %MW en la programación del PLC. Para el direccionamiento en LabVIEW se tiene que el bit interno %M0 tienen la dirección 0 y el bit interno %M127 la dirección 127.

- Se elaboró un programa en LabVIEW (llamado SNIFFER MODBUS) el cual captura la información que viaja por la red Modbus 485 durante un intervalo de

tiempo definido por el usuario y seguidamente visualiza los mensajes consulta/respuesta (para las funciones Modbus: 01, 02, 03, 04, 05, 15, 16) de un esclavo específico. Esta herramienta sirve para examinar los mensajes consulta que obtuvieron una respuesta normal para cada uno de los esclavos conectados a la red. Es importante desde el punto de docencia porque permite al estudiante interpretar las tramas Modbus.

- Se simuló un proceso industrial (planta embotelladora), para el cual se configuró la red 485 con tres PLC's de diferente fabricante (cada PLC contiene una sección del proceso) y el PC (como Maestro). El proceso se intercomunicó con una rutina construida en LabVIEW utilizando las funciones MODBUS, la cual permitió al PC monitorear las variables de cada sección del proceso y controlar estados del proceso.

- Ninguno de los PLC's tenía implementada la función 07 que corresponde a leer estado de la Excepción y la función 08 que corresponde a diagnóstico; no obstante en un proyecto de grado que se esta desarrollando actualmente se está construyendo un esclavo Modbus en un microcontrolador que permitirá probar dichas funciones.

8. OBSERVACIONES

- Para alimentar el conversor (conectado al PC) se tuvo que utilizar una tensión de 9V que se encuentra dentro de los "simuladores" (dispositivos a los cuales se conectan los PLC's para formar la red, poseen interruptores que proporcionan 24V DC a las entradas y led's que visualizan los estados en que se encuentran las entradas y las salidas) y enviarla al pin 7 del conector DB-9 hembra (conector utilizado para la comunicación con el estándar RS-485 en los simuladores), por ello el conversor cuenta con un regulador de voltaje que entrega a los demás integrados una tensión de 5V.

- Debido a la necesidad de conectar el PLC KOYO (no posee el estándar RS-485) a la red se construyó una variante del conversor RS232/RS485, se diferencia con el conversor que se usa para conectar el PC a la red en dos cosas:
  1. La alimentación se toma del PLC y no del simulador debido a que el PLC proporciona una tensión de 5V, haciendo innecesario el regulador de voltaje.
  2. Se toma la señal RTS que envía el PLC como señal de habilitación para el conversor, debido a que esta señal no tiene la polaridad correcta para habilitar el conversor, se debe agregar un inversor al diseño.

- Otra variante del conversor RS232/RS485 se utiliza en la transferencia de programas del PC al PLC NANO y viceversa, las diferencias son las siguientes:

  1. La alimentación de 5V es entregada por el PLC no siendo necesario el regulador de voltaje.
  2. El PLC suministra la señal correcta de habilitación al conversor, evitando: el envío de una señal desde el PC y el acondicionamiento de la señal por hardware.

- Cuando el PLC NANO se configura en modo ASCII, este por defecto empaqueta cada carácter ASCII en 7 bits de datos; por lo tanto en el panel frontal de LabVIEW se debe ajustar los bits de datos a 7 para lograr la comunicación.

- Los PLC's TRILOGI permiten leer (Función 01 y 02) y forzar (función 05) el estado (ON/OFF) de las entradas, salidas (bobinas), contactos de los temporizadores, contactos de los contadores y los relés. En el mapa de memoria de los PLC's TRILOGI (Tabla 17) en la columna Bit MODBUS se encuentran las direcciones para los bits de los elementos nombrados; como el direccionamiento empieza en uno se debe restar uno a todas las direcciones. Ya que todas las direcciones en los mensajes Modbus son referenciadas a cero

- Los PLC's TRILOGI permiten leer (Función 03 y 04) y fijar (función 06 y 16) datos en los registros de entrada, de salida, de contactos de los temporizadores, de contactos de los contadores, relés, valor presente de los temporizadores, valor presente de los contadores, reloj, fecha y memoria de datos. En el mapa de memoria de los PLC's TRILOGI (Tabla 17) en la columna Palabras MODBUS se encuentran las direcciones para los registros nombrados; como el direccionamiento empieza en 40001 se debe restar 40001 a todas las direcciones. Ya que todas las direcciones en los mensajes Modbus son referenciadas a cero

- Los PLC's KOYO permiten leer (Función 01) y forzar (función 15) el estado de las salidas DC (Y), contactos de los temporizadores(T), contactos de los contadores(CT) , bits de las etapas (S) y los relés de control (C). En el Mapa de Memoria de los KOYO DL-250 (Tabla 18) se encuentran el rango de direcciones Modbus (un bit por dirección) para acceder a estos elementos. Por ejemplo para encontrar la dirección Modbus del relé de control C25, se debe:

  Convertir la dirección octal (C**25**) del elemento del PLC a decimal (**21**), seguidamente se suma el rango de inicio (3072 d) que aparece en la Tabla 18 correspondiente a los reles de control; obteniendo que la dirección Modbus es: 21 + 3072 = 3093 d.

- Los PLC's KOYO permiten leer (Función 02) el estado de las entradas DC (X) y de los relés especiales (SP). En el Mapa de Memoria de los KOYO DL-250 (Tabla 18) se encuentran el rango de direcciones (un bit por dirección) para acceder a estos elementos, la dirección Modbus se halla de forma similar al ejemplo del ítem anterior.

- Los PLC's KOYO permiten leer (Función 03 y 04) y fijar un dato (función 06 y 16) en un registro o en un grupo de registros que contienen el valor presente de los temporizadores, valor presente de los contadores y memoria de datos de usuario. En el Mapa de Memoria de los KOYO DL-250 (Tabla 18) se encuentran el rango de direcciones (un registro por dirección) para acceder a estos registros. Por ejemplo, para encontrar la dirección Modbus del registro V1400 que es un registro de la memoria de datos se debe:
  Convertir la dirección octal (V **1400**) del elemento del PLC a decimal (**768),** obteniendo que la dirección Modbus es 768 d.

- Cuando se leen los registros de los temporizadores, contadores y memoria de datos; los PLC's KOYO envían la información en código BCD siendo necesario que el usuario los interprete.

- Los PLC's KOYO no permitieron forzar una bobina (Función 05), pero si permiten forzar un grupo de bobinas (función 15).

BIBLIOGRAFÍA

- MODICON, INC., INDUSTRIAL AUTOMATION SYSTEMS. Protocolo Modbus, Guía de referencia [PDF]. Traducción de PI-MBUS-300 Rev. J. June 1996.

- LÁZARO ANTONIO MANUEL. LABVIEW. PROGRAMACIÓN GRÁFICA PARA EL CONTROL DE LA INSTRUMENTACIÓN. EDITORIAL PARANINFO 1997.

- TRIANGLE RESEARCH INTERNATIONAL PTE LTD. T100MD+ Super Programmable Controllers. User Manual's [PDF]. (United States of America), 1999.

- TRIANGLE RESEARCH INTERNATIONAL PTE LTD. TRILOGI. Programmer's Referente Part two. Programming Language Support forMODBUS and EMIT3.0 features of M+ series PLC [PDF]. (United States of America), 1999.

- AUTOMATIONDIRECT. DL205 User Manual Volume 1 of 2. D2–USER–M [PDF]. (United States of America) 1999.

- AUTOMATIONDIRECT. QUICK-START MANUAL. QS-DSOFT32-M [PDF]. (United States of America) 1999.

- SCHNEIDER ELECTRIC 1994-200. TLX DSCOM Setting up communication functios under PL7[PDF]. (United States of America) 1999.

- B&B Electronics Mfg. Co. Inc. RS-422 and RS-485 Application Note [PDF]. (United of States) Revised October 1997.

- RS 232 / EIA 232 Standard. ECE 351, Fall 2004 [PDF]

- DATASHEET SN75176B. DIFFERENTIAL BUS TRANSCEIVERS [PDF]. SLLS101D – JULY 1985 – REVISED APRIL 2003.

- DATASHEET MAX232, MAX2321 DUAL EIA-232 DRIVERS/RECEIVERS [PDF]. SLLS047L − FEBRUARY 1989 − REVISED MARCH 2004.

ANEXO A

El código ASCII

Como su propio nombre indica, **ASCII** (American Standard Code for Information Interchange), tiene origen americano, en el instituto **ANSI**. El ASCII es simplemente una convención para codificar un conjunto de 128 caracteres (letras, números y símbolos), numerados del 0 a 127. Este número se debe a que emplea sólo 7 bits ($2^7 = 128$), algo que parecía suficiente en la época en que se propuso el estándar. Mucha gente cree que el código ASCII tiene 256 caracteres (numerados del 0 al 255), que es la capacidad de los 8 bits de un Byte, pero lo cierto es el denominado "ASCII puro", también conocido como US-ASCII (técnicamente es el ANSI X3.4-1968), se compone de sólo 7 bits y 128 caracteres (rango 00h-7fh expresado en hexadecimal). El primero es el carácter nulo, y tiene el número 0; el último es el 127.

En los 128 caracteres del código ASCII original se encuentran las letras del alfabeto inglés (mayúsculas y minúsculas), los números, y algunos signos comunes (espacio en blanco, paréntesis, corchetes, suma y resta, dólar, porcentaje, etc.) además de otros como retorno de carro, tabulador, campana, etc. Estos últimos, que eran necesarios para controlar el funcionamiento de los antiguos teletipos, ocupan las 32 primeras posiciones y la última, y se denominan precisamente por eso **caracteres de control** o **no imprimibles**. Puesto que no representan ninguna letra ni signo de puntuación, se designan mediante nemónicos (NULL, NAK, CR, NL, etc).

En la Tabla 26, de caracteres ASCII, el valor Hexadecimal se encuentra entrando en la fila y columna. Por ejemplo, el carácter "A" está en fila 4 columna 1, luego su valor hexadecimal es 41 = 41h. El valor decimal se ha incluido debajo de cada carácter, para la "A" es 65d. Los caracteres no representables se han indicado por sus nemónicos. Al carácter "espacio" le corresponde el decimal 32 (20h).

Tabla 26.Caracteres ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENO 5	ACK 6	BEL 7	BS 8	TAB 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2	 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

ANEXO B

En la Tabla 27 se encuentran los íconos creados en LabVIEW correspondientes a las principales funciones del protocolo Modbus, a cada uno se le puede elegir uno de los dos modos de transmisión (ASCII o RTU).

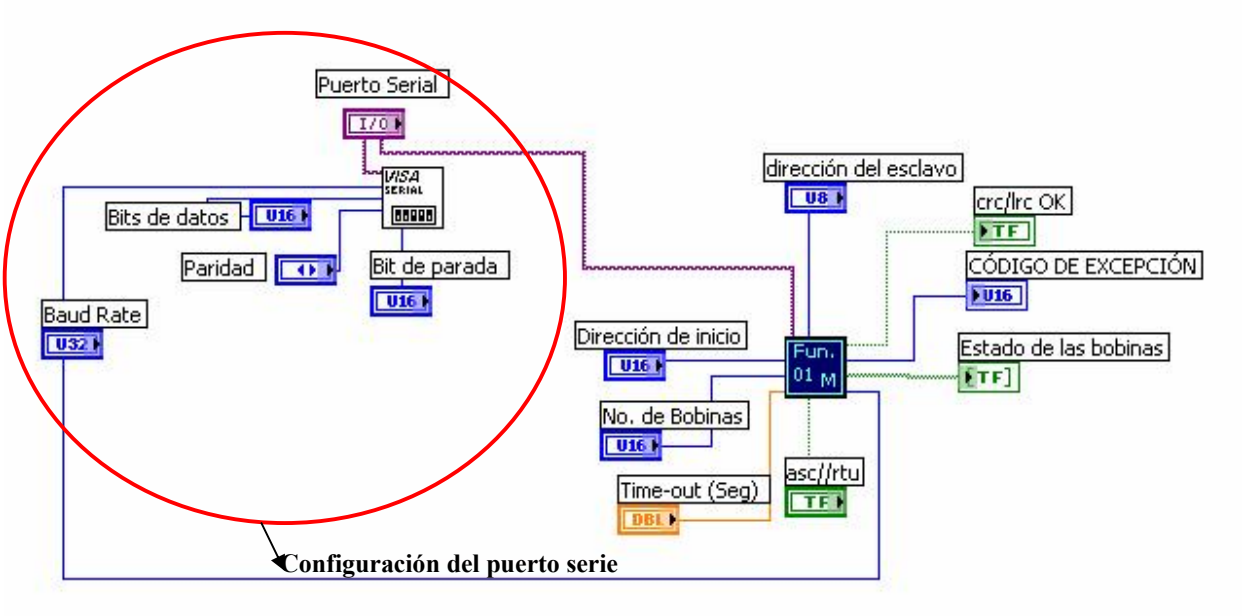
Tabla 27.Iconos de la librería Modbus creada en LabVIEW

	**Leer estado de las Bobinas**
	**Leer estado de las Entradas**
	**Leer Registros Internos**
	**Leer Registros de Entrada**
	**Forzar una Bobina**
	**Fijar un Valor en un Registro**
	**Forzar varias Bobinas**
	**Fijar Varios registros**

Controles y indicadores de la Función 01

En la Figura 119 se encuentra el cableado del sub.vi creado para la Función 01 del protocolo Modbus.

Figura 119. Cableado de la función 01



Controles

- **Dirección del esclavo:** identificación del dispositivo al que va dirigido el mensaje consulta.
- **Dirección de Inicio:** Corresponde a la posición de memoria del dispositivo donde se encuentra la información referente al estado de las bobinas.
- **No. de Bobinas:** Número bobinas (bits) consecutivas que se quieren consultar.
- **Time-Out:** Cantidad de tiempo que el puerto esperara después de haber enviado un mensaje del cual espera respuesta.
- **ascii//rtu:** El estado falso F corresponde a Modo RTU y el estado verdad T corresponde a Modo ASCII.

Indicadores

- **crc/lrc OK**: Indica que el mensaje respuesta no tiene errores.
- **Código de excepción:** entrega el código de la excepción si se produce una respuesta de excepción.
- **Estado de las bobinas:** este indicador contiene un arreglo de bits que contiene el estado de las bobinas del esclavo que fueron solicitadas en el mensaje consulta.

Para el resto de funciones el cableado es similar, varía en la información que se envía en el mensaje consulta y en el resultado obtenido en el mensaje respuesta.