CAPÍTULO 1

INTRODUCCIÓN GENERAL

1.1 EVOLUCIÓN E IMPORTANCIA DE LAS REDES INDUSTRIALES[1]

La evolución de la industria ha sido realmente cada vez más acelerada debido a los muchos cambios que se imponen con la modernidad y los avances, tanto tecnológicos como administrativos, que se producen en todos los campos que tienen alguna relación con ella y la afectan directa o indirectamente.

Fenómenos mundiales como la competitividad, la búsqueda a toda costa del aumento de la calidad, la productividad, el acceso a la información y la satisfacción del cliente han promovido un cambio en la manera de situar a la industria y todos los procesos de ella en el ámbito de una sociedad globalizada como la que se vive hoy.

Lo que en un principio correspondió a una política de “inspección de trabajadores” en las fábricas pronto fue evolucionando hacia la reducción de costos de producción y la satisfacción de los clientes tanto internos (trabajadores) como externos (usuarios finales).

Pronto se dieron cuenta que para satisfacer todo ello no solo bastaba fijarse en el área de producción de la industria, reduciendo los posibles errores de fabricación para minimizar pérdidas, evitando que seres humanos realicen tareas repetitivas o aumentando las unidades producidas en función del tiempo, sino que era necesario integrar cada uno de los procesos que se relacionaban con la industria (sean productivos o administrativos).

Si bien con el aparecimiento de los computadores personales y el Internet se ha facilitado enormemente la gestión de información en las empresas, tradicionalmente ese manejo de información se lo realizaba a nivel meramente administrativo, mediante redes informáticas, excluyendo la visión que se pudiera tener del departamento de producción.

El intercambio de información entre ambos ámbitos de la misma industria se lo hacía exclusivamente mediante informes, con lo cual el proceso visto globalmente correspondía siempre retrasado a un lapso pasado, con muchas actividades intermedias, dificultando así la toma de decisiones oportunas para la gerencia.

Las necesidades de la industria moderna, que se entiende como un todo integrado y no solamente como un conjunto de islas que funcionan transparentemente unas de otras, han obligado a que este concepto se haya difundido rápidamente y se amplíen otros, -incluso el mismo concepto de red, que comparte no solamente información estadística descriptiva sino completa a la hora de tomar decisiones-, en tiempo real, para toda la organización con cada uno de sus departamentos. Sin lugar a dudas, la creciente automatización de los más diversos procesos industriales se debe en gran medida al aumento de la tecnología en dispositivos denominados inteligentes. Aumento que se ha verificado tanto en sus prestaciones como en la manera en la que ellos son capaces de interactuar con otros dispositivos formando redes, haciendo posible no solamente el control centralizado y distribuido de la fabricación sino la información completa de la industria, incluso en los niveles de producción y en tiempo real por parte de la alta gerencia que ahora es capaz de obtener datos de primera mano en su propia red administrativa con una alta confiabilidad.

Del desarrollo descrito aparecen conceptos como el de Fabricación Integrada por Computador (CIM, por sus siglas en inglés), el de Diseño y Manufactura Asistidos por Computador (CAD-CAM), el de Ingeniería Asistida por Computador (CAE) y la producción Justo a Tiempo (Just In Time), entre otros, que han querido optimizar los procesos a partir de la tecnología con la que se puede contar para realizarlo.

A nivel electrónico, es necesario utilizar un “lenguaje común”, o por lo menos unos “traductores” que permitan la comunicación entre dispositivos tan diversos como el sensor de una máquina, o un controlador de cualquier tipo, válvulas del proceso, estaciones de control de partes o el computador de cualquier gerente que necesita saber cómo se mueve la línea de producción al momento, y tiene la autorización para manejar esta información que se convierte en un valor muy sensible de la empresa, para así tomar correctivos administrativos a tiempo, diseñar una estrategia de comercialización de productos, etc.

Son conocidas las redes en el ámbito de clientes en cualquier organización: la red de computadores de la universidad, la del trabajo o el propio Internet que permite compartir información para cumplir una determinada tarea; lo que no se suele ver con tanta frecuencia, a menos que se trabaje en la industria, son las redes industriales las cuales tienen una vital importancia por todo lo que se ha mencionado anteriormente. Y es en este campo en el que la electrónica es de suma importancia, junto con el desarrollo de tecnología en la que se funden campos diversos como la automatización y control, y con el de las comunicaciones que aquí encuentran un punto de interacción común.

1.2 REDES DE DATOS INDUSTRIALES[2]

Una red de comunicación se puede definir básicamente como el soporte de un conjunto de terminales interconectados entre sí con el objetivo de intercambiar información, compartir recursos de hardware y software y mejorar la eficacia de aplicaciones mediante la distribución. Una red industrial maneja el mismo concepto pues es también una red de comunicación.

La particularidad en una red de comunicación, consiste en un conjunto de terminales que se enlazan para el intercambio de información; éste se constituye por automatismos de entorno industrial, es decir, sensores, actuadores, bloques de entrada y salida, unidades registradoras, transmisores y controladores, computadores industriales, PLCs o robots, de acuerdo a la escala de automatización que se muestra en la figura 1.1.

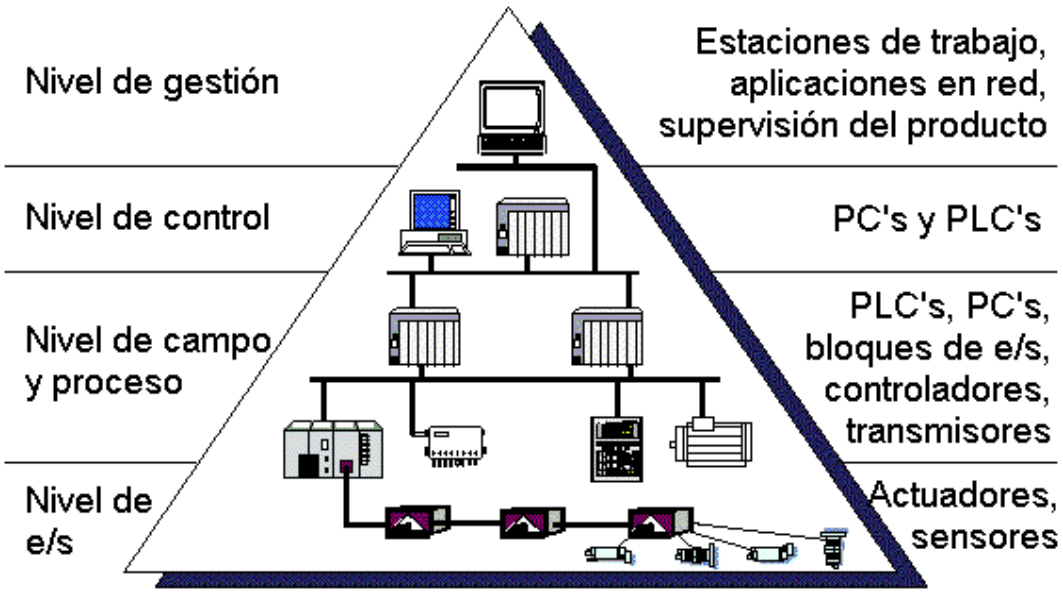

Figura 1.1. Pirámide de la automatización

El ámbito industrial básicamente tiene 4 niveles de automatización debido a la gran cantidad de equipos que coexisten en redes de este tipo.

El primero, el más sencillo y por ello en la base de la pirámide anterior, es el **nivel de entrada–salida (E/S)**, formado por sensores y actuadores que se encuentran directamente en el proceso y por tanto son los encargados de manejarlo directamente.

El segundo, que corresponde a un nivel un poco más alto, es el denominado **nivel de campo y proceso**, que es en el cual ya se integran pequeños automatismos formados por bloques de transmisión y control mediante E/S y PLCs generalmente. En este nivel ya se utilizan buses de campo que se definirán más adelante.

El tercero es el **nivel de control** que se encarga de la dirección completa de la red en sus distintas fases: diseño, control de calidad, programación, etc. En este nivel se utilizan equipos de alta escala como PLCs o computadores industriales.

Y finalmente el nivel más alto corresponde al **nivel de gestión** que básicamente integra el área de producción con el área de administración y gerencia (inventarios, ventas, contabilidad, etc.). Este nivel suele incluir equipos como computadores personales conectados en redes locales, las conocidas LAN y WAN.

Del sinnúmero de ventajas que resultan cuando se enlazan sistemas de una planta automatizada con otros sistemas como: mecanismos automáticos programables, sistemas de control digitales o de control numérico, incluso con sistemas de captación, presentación de datos, de proceso y gestión, se mencionan principalmente cinco:

1. La posibilidad de intercambio confiable y rápido de información entre automatismos que controlan fases sucesivas de un mismo proceso productivo.

2. La factibilidad de comunicación humano-máquina en base de terminales inteligentes que permiten no solo observar sino programar el proceso en términos de lenguaje muy parecido al humano.

3. La adquisición de datos desde sensores y su procesamiento para un posterior control de calidad, gestión, estadística u otros procesos que requiere la industria.

4. La notable facilidad de la empresa para adaptarse rápidamente a la evolución de la industria y a la diversificación de sus productos.

5. La posibilidad de uso de lenguajes de alto nivel que permiten tratar bajo un mismo entorno todas y cada una de las celdas automatizadas desde la fase de diseño hasta la fase de explotación y gestión.

La figura 1.2 presenta el estándar OSI[3] (sistema de interconexión abierta, por sus siglas en inglés) de la Organización Internacional de Estándares (ISO). Éste consta de siete niveles distribuidos en tres grandes bloques (bloque de

transmisión, bloque de transporte y bloque de niveles superiores) que grafican las relaciones de comunicación para cualquier sistema.

Figura 1.2. Modelo de comunicaciones ISO – OSI

El primer gran bloque, de transmisión, lo constituyen los niveles físico, de enlace y de red. El segundo es el nivel de transporte y el tercer bloque está constituido por los niveles de sesión, de presentación y de aplicación.

El **nivel físico** permite el flujo de bits por un medio físico y corresponde básicamente a la interfase de señales físicas mensurables como voltaje o corriente.

El **nivel de enlace**, en cambio, permite la estructuración de esos bits bajo un formato básico denominado trama (o *frame*, en inglés); la transmisión de tramas, la verificación de errores y retransmisión en caso de ser necesario. Este nivel posee dos capas, el MAC (*media acces control* o control de acceso al medio) que depende de cada técnica utilizada para accesar al medio y el LLC (*logic link control* o, en español, control de enlace lógico) que es común para todas las redes.

El **nivel de red** estructura el mensaje de la comunicación en paquetes que previamente fueron encapsulados en tramas en el nivel anterior de enlace. Esta estructuración se realiza tanto en el transmisor como en el receptor.

El **nivel de transporte** permite enviar mensajes a múltiples destinos posibilitando la comunicación bien sea punto a punto o multiplexada sin errores.

El **nivel de sesión** fija “sesiones” para cada uno de los usuarios de una máquina, permitiendo la gestión de información en la red (posibilidad de que únicamente ciertos usuarios puedan conectarse, modo, manera, accesos, etc.).

El **nivel de presentación** define la estructura de los datos (su tipo: número, caracter, etc.) así como los códigos empleados para transmitir (ASCII o RTU).

Y finalmente, el **nivel de aplicación,** que provee todos los servicios de red como correo electrónico, protocolo de transferencia de archivos, acceso remoto a máquinas, páginas web, entre otros.

Cada uno de los elementos que se encuentran en un nivel de la pirámide de automatización cuando se conectan a otros necesariamente lo hacen a través de alguno(s) de los niveles del modelo OSI.

Desde un simple actuador o sensor que posiblemente (y dependiendo del tipo) podría comunicarse con otro a nivel físico o de enlace, hasta un computador que incluirá todos los niveles del modelo OSI. Existen, básicamente, dos tipos de redes industriales que corresponden a los estándares más difundidos a nivel mundial.

Las redes industriales son las ubicadas jerárquicamente más alto debido a sus prestaciones. Estos dos estándares de redes industriales son:

- MAP (Protocolo de fabricación automática, por sus siglas en inglés), que fue impulsado por General Motors como un producto diseñado especialmente para el entorno industrial. Fue luego normalizada por el

IEEE (Institute of Electrical and Electronics Engineers). Permite la integración a redes LAN y WAN y aunque no actúa a nivel de bus de campo la integración se la puede realizar mediante terminales que sirven de puente.

- ETHERNET, registrada por Xerox Corporation (quien la diseñó), Digital Inc. e Intel, es una red flexible que permite diversas topologías (de árbol o de bus) con comunicación semi-dúplex entre 10 Mbps y 100 Mbps. Su diseño se basa en el modelo OSI, en los niveles especialmente inferiores (físico, de enlace y de red). Su particular aplicación en redes de oficina (administrativas u ofimáticas) le ha permitido evolucionar rápidamente.

1.3 BUSES DE CAMPO[2]

Como se mencionó anteriormente, en el nivel de campo y proceso de la pirámide de la automatización se utilizan ya los buses de campo que son los más próximos a los elementos que forman parte de un proceso industrial.

Los buses de campo se conectan, de acuerdo al modelo OSI, en las capas física, de enlace y transporte, a través de protocolos simples y con las funciones mínimas para la comunicación.

Su importancia radica en que poseen la capacidad de juntar muchos tipos de dispositivos de entrada/salida (E/S) en tiempo real a través de un bus serial de datos digitales, que se conectan físicamente gracias a la capa MAC (Control de acceso al medio) que trabaja a nivel de tarjeta de red, en protocolos como RS-232, RS–422 o RS-485.

Esto permite un mejor sistema de comunicación debido a que:

- Disminuye la cantidad de errores.
- Reduce el cableado y el equipo complementario.
- Facilita la variación en cuanto al número de elementos de la red.

La fiabilidad del sistema de bus de campo se debe principalmente al tiempo real que estos pueden manejar ya que los mensajes son pequeños (tramas pequeñas) y se realizan tanto peticiones como respuestas periódicamente.

Las limitaciones, así mismo, que tienen los buses de campo corresponden a los mecanismos para que los mensajes sean pequeños y por tanto fáciles de gestionar en un tiempo corto, manejando adecuadamente los errores para evitar una sobrecarga de mensajes en el bus. Pero quizá la mayor limitación de los buses de campo es la carencia de estandarización a nivel mundial pues cada fabricante utiliza su propio sistema.

La única normalización con la que se cuenta para los buses de campo es la norma IEC (comité TC65C-WG6), que si bien es importante en el nivel de capa física, es muy abierta en los demás niveles, ayudando en realidad poco a la estandarización.

Dentro de los sistemas más utilizados como buses de campo se encuentran:

- Modbus Modicon, marca registrada de la firma Gould Inc.
- Bitbus, marca registrada de la firma Intel.
- Profibus, impulsada por fabricantes alemanes.
- FIP (Factory Instrumentation Bus), impulsada por fabricantes franceses.
- MIL-STD-1553B, auspiciada por algunos fabricantes norteamericanos.
- Foundation Fieldbus
- AS-I
- Device-Net
- Industrial Ethernet, que integra más niveles del modelo OSI directamente para entorno industrial.

1.4 IMPORTANCIA DEL BUS DE CAMPO MODBUS MODICON[4]

Uno de los buses de campo más importantes es el Modbus Modicon. Técnicamente, Modbus es uno de los protocolos más tempranamente utilizados para controladores lógicos programables, los bien famosos PLCs.

Un protocolo es solamente un lenguaje común para comunicar dos entidades separadas para complementar una tarea.

En el caso de Modbus, el protocolo permite al maestro leer o escribir la memoria de datos de un esclavo remoto. Así que Modbus es realmente solo un simple protocolo de acceso remoto a memoria. Distinto a la mayoría de protocolos, un programador competente puede agregar calidad a un driver Modbus pues es un protocolo abierto.

Modbus no tiene restricciones por encima de los 16 bits de palabra en el acceso a su contenido. Puede empaquetar bits, enteros o porciones de cadenas, números de punto flotante, contadores o bloques de programa.

Lo anterior permite realizar cualquier tarea de control, desde un simple interruptor para una puerta hasta una compleja estructura, que maneje tareas más complicadas. La mayoría de otros protocolos tienen una docena de diferentes comandos para diferentes tipos de memoria, contadores, temporizadores, reloj, etc. Usando esos otros protocolos se puede volver más complicada la programación.

La importancia del protocolo Modbus Modicon radica en que puede comunicarse con gran cantidad de dispositivos sobre una gran variedad de redes que incluyen redes industriales Modbus Modicon y Modbus Plus, y las redes estándar tales como MAP y Ethernet, que se mencionaron anteriormente.

De acuerdo a las redes descritas en el apartado anterior, las mismas son accesadas a través de puertos incorporados en los controladores o mediante adaptadores de red, módulos de opción, y gateways que están disponibles para

Modicon. Para los fabricantes de equipos, los programas de Modicon ModConnect están disponibles para una estrecha integración de redes como Modbus Plus en diseños de producto patentados.

Varias firmas como Fuji Electric, Allen-Bradley o Schneider Electric han creado dispositivos que pueden ser fácilmente conectados a redes que negocian bajo el protocolo Modbus y que comunican con otros protocolos de mayores prestaciones que utilizan más capas del modelo ISO–OSI como TCP/IP, Profibus, Fieldbus, Industrial Ethernet, etc. Como ya se ha mencionado anteriormente, el lenguaje común usado por todos los controladores Modicon es el protocolo Modbus. Este protocolo define una estructura de mensaje que los controladores reconocen y usan, sin importar el tipo de red sobre la cual se están comunicando.

La estructura del mensaje Modbus describe el proceso que un controlador usa para solicitar el acceso a otro dispositivo, como responderá a peticiones de otros dispositivos, y como los errores serán detectados y reportados. Establece un formato común para la presentación y el contenido de los campos del mensaje. El protocolo Modbus proporciona el estándar interno que los controladores Modicon utilizan para el análisis de mensajes. Durante la comunicación sobre una red Modbus el protocolo determina cómo cada controlador sabrá su dirección de dispositivo, cómo reconocerá un mensaje direccionado hacia el, cómo determinará la clase de acción a ser tomada, y la manera de extraer un dato o información contenida en el mensaje. Si se requiere una contestación, el controlador construirá el mensaje de contestación y lo enviará usando el mismo formato en el protocolo Modbus.

En otras redes, los mensajes que contiene el protocolo Modbus encajan en la estructura de la trama o paquete que se utiliza en la red. Por ejemplo, los controladores de la red Modicon para Modbus Plus o MAP, con aplicaciones asociadas a librerías de software y drivers, proporcionan la conversión entre el protocolo de mensaje Modbus y los protocolos específicos de trama de esas redes que usan para comunicarse entre sus dispositivos de nodo. Esta conversión también se extiende a direcciones de nodo, encaminamiento de trayectorias, y métodos de chequeo de errores específicos a cada clase de red. Por ejemplo, las

direcciones de dispositivo Modbus contenidas en dicho protocolo serán convertidas en direcciones de nodo, previa a la transmisión de mensajes. Los campos de chequeo de errores también serán aplicados a los paquetes de mensaje, consecuente con cada protocolo de red. La figura 1.3 muestra la manera como los dispositivos pueden interconectarse en una jerarquía de redes que emplea técnicas de comunicación diferentes.

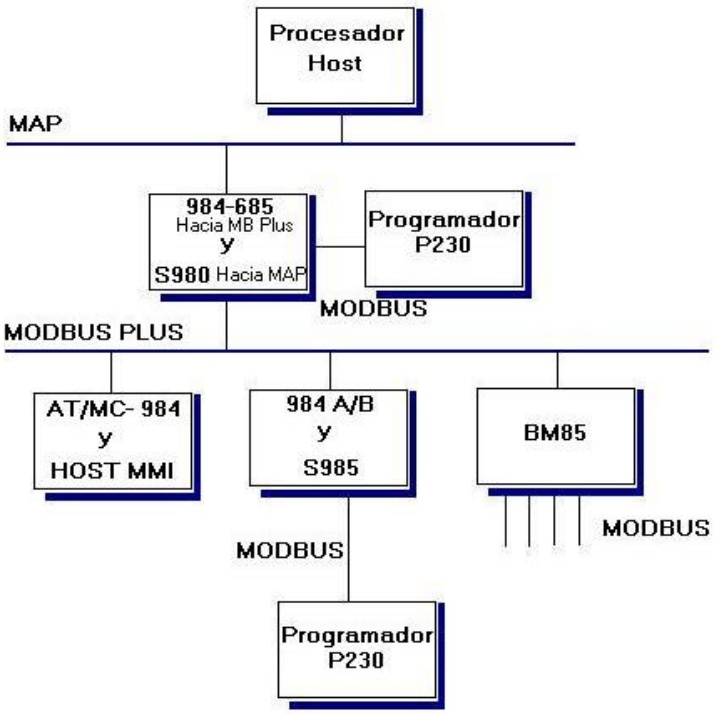

Figura 1.3. Descripción de una aplicación del protocolo Modbus

Referencias:

1. Universidad Rafael Belloso Chacín, Decanato de Investigación y Postgrado, Maestría en Telemática. Maracaibo, febrero del 2000.

2. Bacells, Joseph y Romeral, José Luis, Autómatas programables, Serie Mundo Electrónico, Marcombo editores.

3. Electrónica y computadores, N° 9, Primera edición, Publicaciones CEKIT, Pereira Colombia 1994, Págs. 21 – 25.

4. *http://www.modbus.org, Modbus Protocol Reference Guide.pdf y Modbus over Serial Line-Specification and Implementation Guide V1.0.pdf.*

CAPÍTULO 2

DESCRIPCIÓN DEL PROTOCOLO MODBUS

2.1 DESCRIPCIÓN GENERAL DEL FUNCIONAMIENTO DEL PROTOCOLO[1]

2.1.1 La conexión serial RS-232C

Los puertos Modbus estándar sobre controladores Modicon utilizan una interfase serial compatible RS-232C que define completamente aspectos como los pines de salida del conector, cables, niveles de señal, velocidad de transmisión y comprobación de paridad.

El enlace RS–232C recibe su nombre de la norma americana de EIA (Electrical Industries Association), equivalente al estándar europeo V.24 del CCIT (Comité Consultivo Internacional Telegráfico y Telefónico).

En concreto, el enlace definido por la norma utiliza 25 líneas (para datos y control) y conectores DB-25. La denominación V.24 viene del hecho de que los niveles de tensión utilizados son de +12V y –12V (para 0 y 1 lógicos, respectivamente). En realidad, existe una banda de tolerancia para estas tensiones aproximadamente de entre ±5V a ±15V, los positivos para el 0 lógico y los negativos para el 1 lógico.

No siempre se suele utilizar todas las líneas para esta interfase, lo que hace que muchos utilicen un conector DB-9 en lugar del conector DB-25. Según la norma, la distancia máxima entre elementos conectados según este estándar no debe sobrepasar los 15m para evitar ruido electromagnético, aunque en la práctica suele funcionar correctamente para distancias de hasta unos 100m.

2.1.2 El conector DB-9

El conector DB-9 es el que realiza de interfase para la conexión serial como se mencionó anteriormente, su distribución de pines se detalla en la tabla 2.1.

Tabla 2.1. Distribución de pines del conector DB-9

DB-9	Función	Abreviación
1	Data Carrier Detect	DCD
2	Receive Data	RD or RX or RXD
3	Transmitted Data	TD or TX or TXD
4	Data Terminal Ready	DTR
5	Signal Ground	GND
6	Data Set Ready	DSR
7	Request To Send	RTS
8	Clear To Send	CTS
9	Ring Indicator	RI

Su respectivo conector se presenta en la figura 2.1.

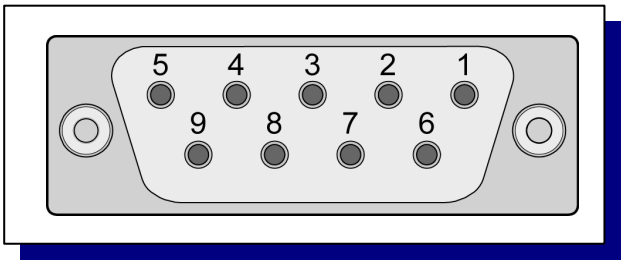

Figura 2.1. Diagrama de pines del conector DB-9

2.1.3 El circuito integrado MAX232

Este integrado es usado para comunicar un sistema digital con un PC o sistema basado en el bus serie RS-232, pues dispone internamente de 4 conversores de niveles TTL al bus estándar RS-232 y viceversa. El C.I. MAX232 lleva internamente 2 conversores de nivel de TTL a RS-232 y otros 2 de RS-232 a TTL con lo que en total se puede manejar 4 señales del puerto; por lo general las más usadas son: TX, RX, RTS, CTS, estas dos últimas son las usadas por el protocolo para handshaking pero no es imprescindible su uso.

Para que el C.I. MAX232 funcione correctamente se deberá poner unos condensadores externos de un 1 µF, tal como se muestra en la figura 2.2.

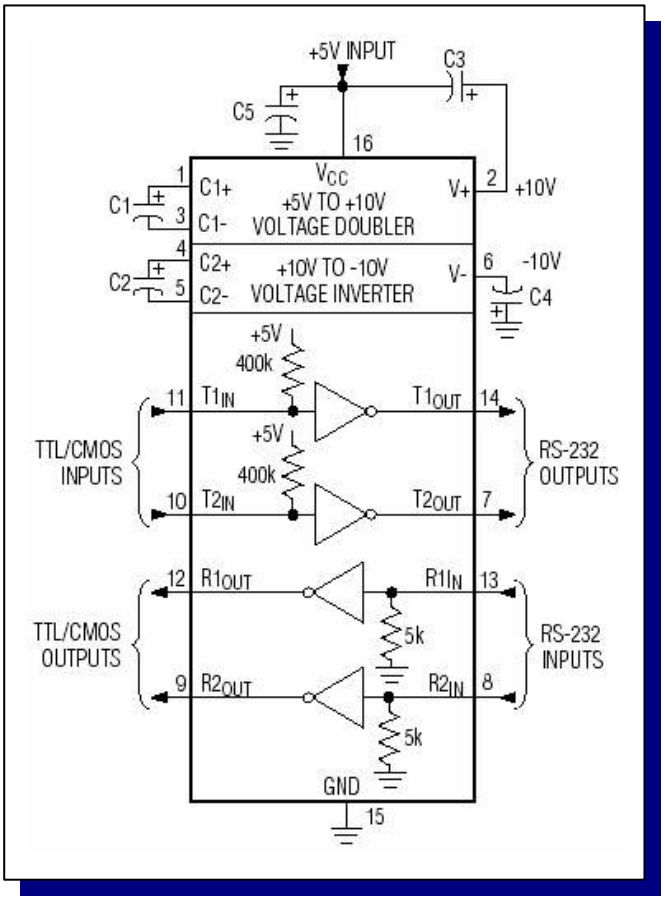

Figura 2.2. Diagrama del C.I. MAX232

2.2 GUÍA DE ESPECIFICACIÓN E IMPLEMENTACIÓN MODBUS SERIAL[1]

El protocolo serial Modbus es un protocolo Maestro-esclavo que toma lugar en el nivel 2 del modelo OSI. La interfase TIA/EIA-484 (RS-485) de dos-hilos es la más comúnmente utilizada. Como una opción adicional, la interfase RS-485 de cuatro-hilos también puede ser implementada.

La interfase serial TIA/EIA-232 (RS-232) se utiliza cuando solamente se requiere una comunicación punto a punto de corto alcance. En la figura 2.3 se da una explicación general de la pila de comunicación serial Modbus comparado con las 7 capas del modelo OSI.

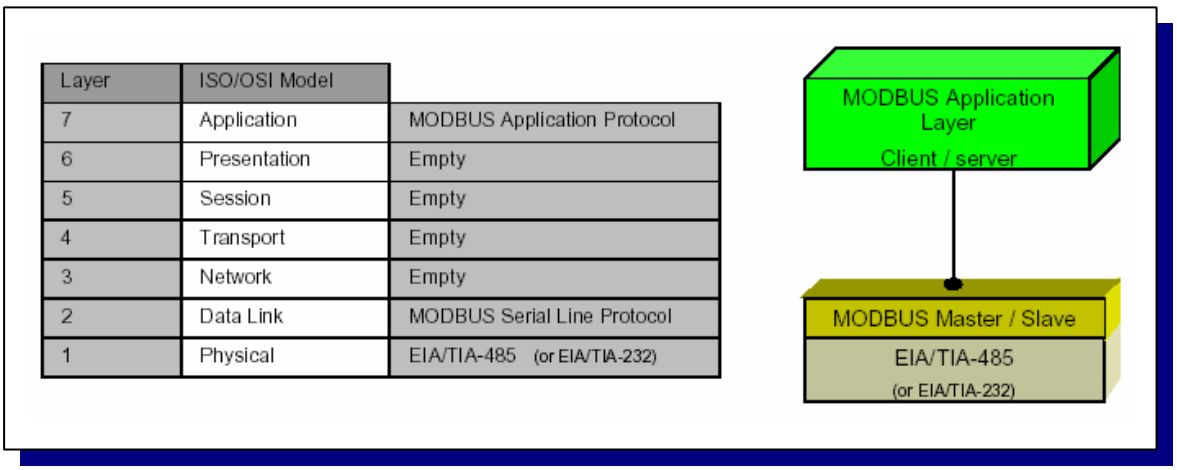

Figura 2.3. Protocolo Modbus y Modelo ISO/OSI

La capa de aplicación Modbus del protocolo de mensaje, posicionada en el nivel 7 del modelo OSI, provee una comunicación cliente/servidor entre dispositivos conectados en buses o redes. El rol del cliente sobre una línea serial Modbus está proporcionado por el Maestro del bus serial y los nodos esclavos que actúan como servidores.

2.2.1 Capa Física Modbus

Una nueva solución Modbus sobre una línea serial puede implementarse en una interfase eléctrica de acuerdo con el estándar EIA/TIA-485 (también conocido como estándar RS-485)[2]. Este estándar permite sistemas multipunto, punto a punto, en configuración de dos-cables. En suma, muchos dispositivos podrían también implementar la interfase RS-485 de cuatro-hilos o la interfase RS-232.

Sobre un sistema Modbus estándar, todos los dispositivos son conectados en paralelo sobre un cable constituido por tres conductores. Dos de estos (configuración de dos-hilos) forman un par trenzado balanceado, sobre el cual los datos son transmitidos bidireccionalmente a una tasa de transmisión de datos típica de 9600 bits por segundo.

2.2.1.1 Definición Modbus de dos-hilos

En un bus de dos hilos (figura 2.4) solamente un driver tiene derecho de transmitir. En realidad, un tercer conductor debe ser interconectado a todos los dispositivos del bus; el común.

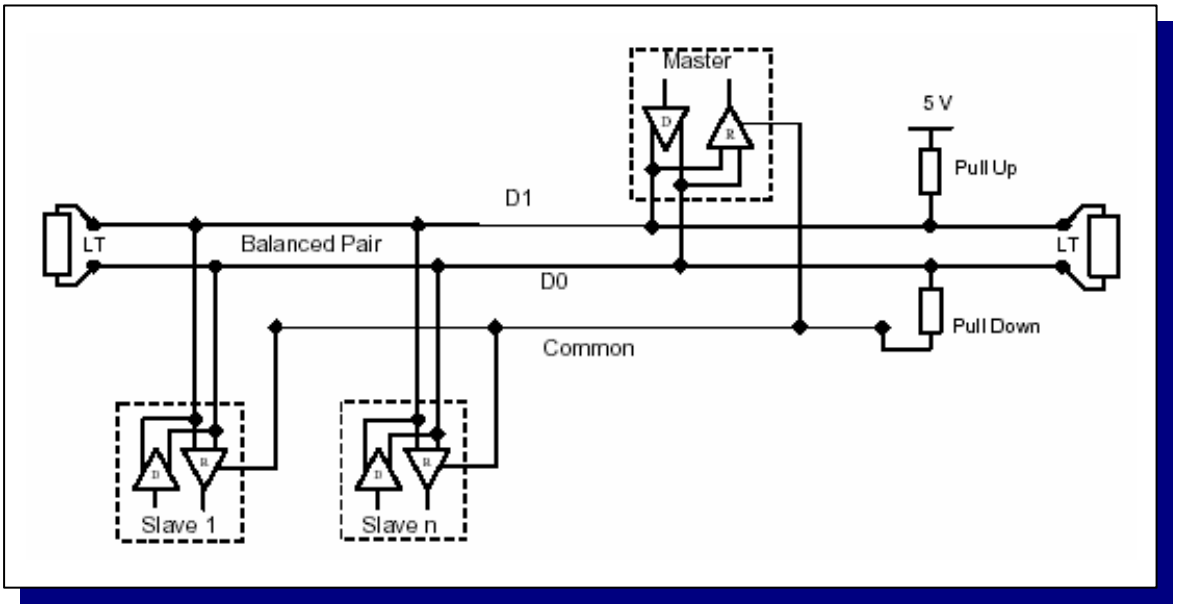

Figura 2.4. Topología general de dos-hilos

2.2.1.1.1 Requerimientos de los Sistemas Multipunto

Para algunos de los sistemas multipunto EIA/TIA-485, los siguientes requerimientos son aplicables.

2.2.1.1.2 Máximo número de dispositivos sin repetidora

Una figura de 32 dispositivos es siempre autorizada sobre un sistema Modbus RS-485 sin repetidora. Dependiendo de:

- todas las posibles direcciones,
- la figura de las Unidades de Carga RS-485 usadas por los dispositivos,
- y la línea de polarización necesaria.

2.2.1.1.3 Topología

Una configuración Modbus RS-485 sin repetidora tiene un cable a través del cual los dispositivos son conectados directamente o por cables de derivación cortos. Este cable es llamado bus, sus dos extremos deben tener Terminaciones de Línea. También es posible el uso de repetidoras entre Modbus RS-485.

2.2.1.1.4 Longitud

La longitud del cable del bus debe ser limitada y depende de varios factores: la tasa de transmisión de datos, el calibre del cable, su impedancia característica, el número de esclavos conectados y la configuración de la red (2 o 4 hilos). Por ejemplo, para una tasa de transmisión (baude rate) de 9600bps y un calibre de cable AWG26, la longitud máxima del bus será de 1000m.

Las derivaciones conectadas hacia los esclavos deben ser cortas y nunca superar los 20m. Si un tap multipuerto es usado para conectar n esclavos, cada conexión debe respetar una longitud máxima de 40m dividido para n. Dentro de esta configuración de dos-hilos el tercer conductor (común) debe estar conectado a tierra protectora en un solo punto del bus, que por lo general es el dispositivo maestro. La conexión de la tierra de la fuente de poder al común es opcional.

2.2.1.1.5 Terminación de Línea

Una reflexión en la línea de transmisión es el resultado de una discontinuidad en su impedancia que una onda viajera ve cuando esta se propaga. Para minimizar las reflexiones desde los extremos del cable RS-485 este requiere colocar una Terminación de Línea (LT) cerca de cada uno de los dos extremos del bus. Es importante que la línea esté terminada en ambos extremos ya que la propagación es bidireccional, por tal razón no es permitido colocar mas de dos LT sobre un par balanceado pasivo D0-D1 y menos aún colocar un LT sobre un cable de derivación, en pocas palabras, cada terminación de línea debe ser conectada entre los dos conductores de la línea balanceada: D0 y D1.

Si no existe polarización en los conductores de la línea balanceada, las terminaciones de línea estarán constituídas por resistencias de 150 ohmios (0.5W). Por el contrario, si existe polarización, las terminaciones de línea estarán constituídas de un capacitor de 1nF (10V mínimo) en serie con una resistencia de 120 ohmios (0.25W). Todo lo descrito se especifíca en una interconexión RS-485, en una interconexión RS-232 ninguna terminación debe ser cableada.

2.2.1.1.6 Polarización de Línea

Cuando no hay actividad sobre un par balanceado RS-485, las líneas son susceptibles a ruido externo o interferencia. Para asegurar que este receptor se mantenga en un estado constante cuando no hay señal de datos presentes, muchos dispositivos necesitan un bias en la red.

Si uno o muchos dispositivos necesitan polarización, un par de resistencias deben ser conectadas sobre un par balanceado RS-485:

- Una resistencia de Pull-up a 5V sobre el circuito D1,
- Una resistencia de Pull-down para el circuito común sobre el circuito D0.

Los valores de estas resistencias deben ser entre 450 ohmios y 650 ohmios. Esta última puede permitir un alto número de dispositivos sobre un bus de línea serial.

El máximo número de dispositivos autorizados en una línea serial Modbus es reducido por 4 desde una línea serial Modbus sin polarización.

2.2.2 Capa de Enlace de Datos Modbus

2.2.2.1 Principio del Protocolo Maestro / Esclavo Modbus

El nodo maestro distribuye una petición Modbus a los nodos esclavos en dos formas:

2.2.2.1.1 Modo Unicast

El Maestro direcciona un esclavo individual. Luego de recibir y procesar la petición, el esclavo devuelve un mensaje al maestro.

En este modo, una transacción Modbus consiste de dos mensajes: una petición del maestro y una respuesta del esclavo. Cada esclavo debe tener una única dirección (desde 1 a 247) las que pueden ser direccionadas independientemente por otros nodos. La representación de este modo se muestra en la figura 2.5.

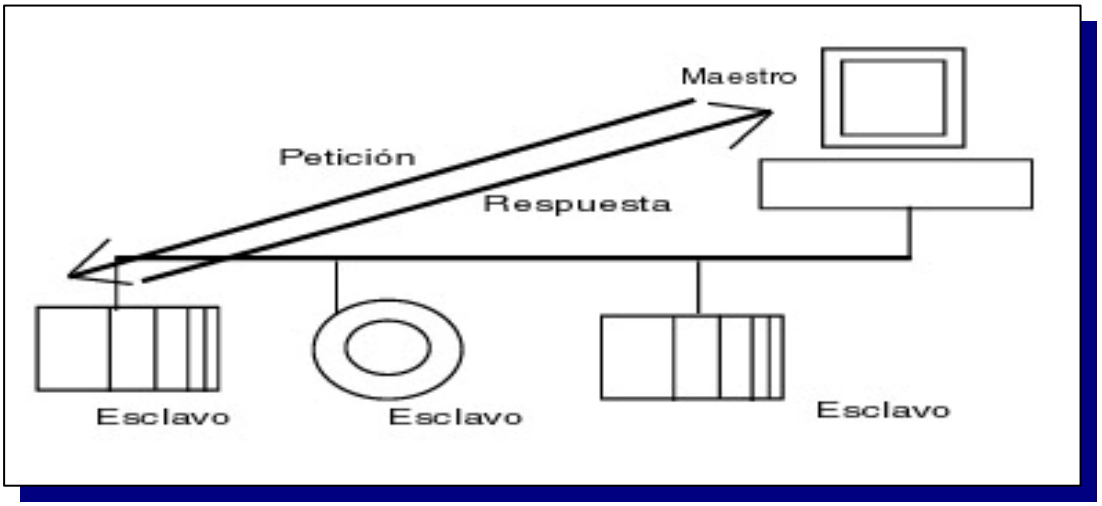

Figura 2.5. Modo Unicast

2.2.2.1.2 Modo Broadcast

El maestro puede enviar una petición a todos los esclavos. No hay respuestas a las peticiones de broadcast hechas por el maestro.

Las peticiones de broadcast son necesariamente peticiones de escritura. Todos los dispositivos deben aceptar la función de broadcast para la función de escritura.

La dirección cero está reservada al intercambio de broadcast.

La representación de este modo se muestra en la figura 2.6.

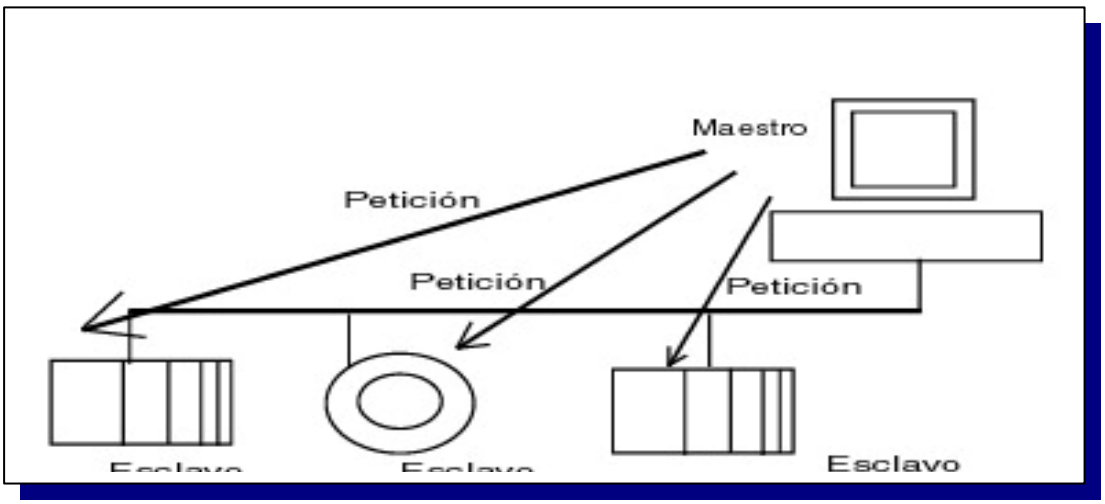

Figura 2.6. Modo Broadcast

2.2.2.2 Reglas de Direccionamiento Modbus

El espacio de direccionamiento Modbus comprende 256 diferentes direcciones, dividiéndose tal como lo muestra la tabla 2.2.

Tabla 2.2. Espacio de direccionamiento Modbus

Dirección de broadcast	Direcciones de los esclavos individuales	Reservado
0	de 1 a 247	de 248 hasta 255

La dirección 0 está reservada como la dirección de broadcast. Todos los nodos esclavos deben reconocer dicha dirección. El nodo Maestro Modbus no tiene una dirección específica, solamente los nodos esclavos tienen una dirección. Esta dirección debe ser única en el bus serial Modbus.

2.3 LA COMUNICACIÓN MAESTRO-ESCLAVO[1]

Los controladores Modbus pueden ser conectados bien directamente en red o bien vía módems, y se comunican usando la técnica maestro-esclavo, en la cual solamente un dispositivo (denominado maestro) puede iniciar las transacciones (peticiones). Los otros dispositivos (denominados esclavos) únicamente

responden con los datos solicitados por el maestro, o ejecutando la acción solicitada en la petición.

El maestro puede direccionar esclavos individuales o puede iniciar un mensaje de difusión (broadcast) a todos los esclavos. Los esclavos, a su vez, devuelven un mensaje (la respuesta) a las preguntas que les son direccionadas individualmente. Los esclavos no necesitan devolver respuestas si el maestro genera y envía preguntas de difusión (broadcast).

El protocolo Modbus establece el formato para la pregunta del maestro, poniendo en ella la dirección del dispositivo (o la orden de difusión), un código de función que define la acción solicitada, cualquier dato a enviarse y un campo de revisión de errores, como se verá posteriormente. El mensaje de respuesta del esclavo, evidentemente, también se construye utilizando el protocolo Modbus. Éste contiene la dirección del dispositivo esclavo que responde, el código de función, los campos que confirman la acción tomada, cualquier dato a ser devuelto y un campo de revisión de errores.

Si ocurrió un error en la recepción del mensaje o si el esclavo no puede realizar la acción solicitada, éste construirá un mensaje de error y lo enviará como respuesta.

Además de las capacidades estándar de Modbus, algunos modelos de controladores Modicon pueden comunicarse sobre el protocolo Modbus Plus, usando puertos incorporados o adaptadores de red, y sobre MAP usando adaptadores de red. En estas redes, los controladores se comunican con la técnica punto a punto, en la cual cualquier controlador puede iniciar transacciones con los otros. Así un controlador puede funcionar como esclavo o como maestro en transacciones separadas.

Las múltiples trayectorias internas se proporcionan con frecuencia para permitir el proceso concurrente de las transacciones maestro-esclavo. En el nivel de mensaje, el protocolo Modbus todavía aplica el principio maestro-esclavo aunque el método de la comunicación a través de la red es punto a punto.

Si un controlador origina un mensaje, lo hace como un dispositivo maestro y espera una respuesta de un dispositivo esclavo. De forma similar, cuando un controlador recibe un mensaje construye una respuesta del esclavo y la devuelve al controlador que la originó. El formato de comunicación Maestro-Esclavo Modbus se presenta en la figura 2.7.

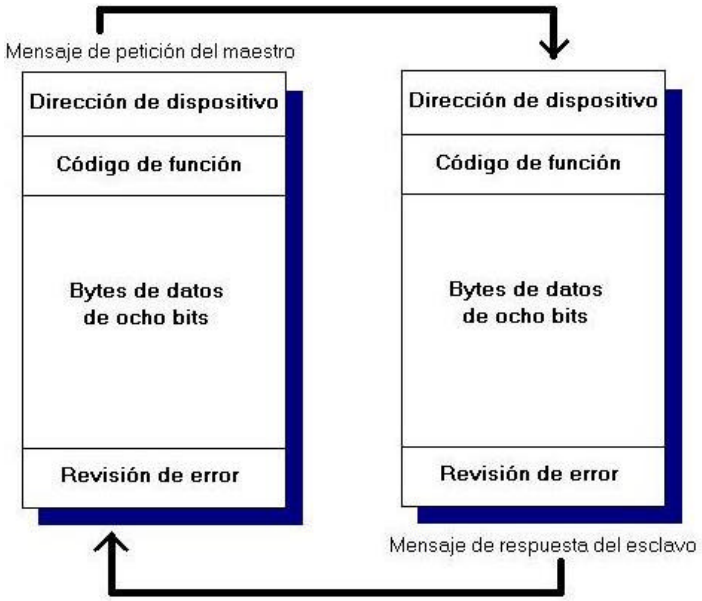

Figura 2.7. Formato de comunicación Maestro – Esclavo Modbus

El código de la función en la pregunta le dice al dispositivo esclavo direccionado la clase de acción a realizarse. Los bytes de datos contienen cualquier información adicional que el esclavo necesite para realizar la función.

Por ejemplo, el código de función 03H preguntará al esclavo sobre la lectura de registros y éste a su vez responderá con su contenido. El campo de datos debe contener la información que diga al esclavo en qué registro comenzar y cuántos registros va a leer. El campo de chequeo de error proporciona un método para que el esclavo valide la integridad del contenido del mensaje.

Los bytes de datos contienen los datos recogidos por el esclavo, tal como valores del registro o estado. Si ocurre un error, el código de la función se modifica para indicar que la respuesta es una respuesta de error, y los bytes de datos en ese instante vienen a contener un código que describe dicho error.

El campo de chequeo de error permite que el maestro confirme que el contenido del mensaje es válido.

2.3.1 Los dos Modos de Transmisión Serial

Los controladores pueden estar dispuestos para comunicarse en redes estándar Modbus usando cualquiera de los dos modos de transmisión: el ASCII o el RTU. Los usuarios pueden seleccionar el modo deseado, junto con los parámetros de la comunicación del puerto serie (velocidad, paridad, etc.), durante la configuración de cada controlador.

El modo y los parámetros seriales deben ser iguales para todos los dispositivos en una red Modbus. La selección del modo ASCII o RTU, pertenece solamente a las redes estándar Modbus.

Esta selección define el contenido de bits de los campos del mensaje transmitido serialmente en esas redes, y determina también la manera en que la información será empacada y decodificada en los campos del mensaje.

En otras redes como MAP y Modbus Plus, los mensajes Modbus se colocan en tramas que no se relacionan con la transmisión serial. Por ejemplo, una petición de lectura de registros se puede manejar entre dos controladores sobre Modbus Plus, sin considerar la disposición actual del puerto serie Modbus de cualquier controlador.

2.3.1.1 Modo ASCII

Cuando los controladores están dispuestos para comunicarse sobre una red Modbus usando modo ASCII (Código Estándar Americano para el Intercambio de Información, de sus siglas en inglés), cada byte de ocho bits en un mensaje se envía como dos caracteres ASCII. La ventaja principal de este modo es que permite intervalos de tiempo de hasta un segundo entre caracteres sin causar un error. El formato para cada byte en modo ASCII se muestra en la tabla 2.3.

Tabla 2.3. Formato de byte en comunicación modo ASCII

Sistema de codificación
hexadecimal: caracteres ASCII entre 0... 9, A... F
Un carácter hexadecimal contenido en cada carácter ASCII del mensaje
Bits por Byte
1 bit de inicio. 7 bits de datos, el bit menos significativo se envía primero. 1 bit para paridad (par o impar); si no existe paridad, no hay bit de paridad. 1 bit de parada; 2 bits si no hay paridad.
Campo de Chequeo de Error Chequeo de redundancia longitudinal (LRC)

2.3.1.2 Modo RTU

Cuando los controladores están dispuestos a comunicarse en una red Modbus usando modo RTU (Unidad Terminal Remota), cada byte de ocho bits en un mensaje contiene dos caracteres hexadecimales de cuatro bits.

La ventaja principal de este modo es que su mayor densidad de caracteres permite un procesamiento de datos mejor que el ASCII para la misma velocidad.

Cada mensaje se debe transmitir en un flujo continuo, siendo el formato para cada byte en modo RTU el que se muestra en la tabla 2.4.

2.3.2 Entramado del Mensaje Modbus

En cada uno de los dos modos de transmisión serial (ASCII o RTU), el mensaje Modbus es colocado por el dispositivo que transmite la trama en la que se conoce un punto inicial y otro final. Esto permite recibir de los dispositivos el inicio del mensaje, leer la porción de la dirección y determinar el dispositivo que está direccionado (o todos los dispositivos, si el mensaje es de difusión), y saber cuando se termina el mensaje. Gracias a ello, los mensajes parciales pueden ser detectados fácilmente y los errores se pueden fijar como un resultado.

Tabla 2.4. Formato de byte en comunicación modo RTU

Sistema de codificación
Binario de ocho bits, hexadecimal: 0... 9, A... F
Dos caracteres hexadecimales contenidos en cada campo de ocho bits del mensaje.
Bits por Byte
1 bit de inicio. 8 bits de datos, el bit menos significativo se envía primero. 1 bit para paridad (par o impar); si no existe paridad, no hay bit de paridad. 1 bit de parada; 2 bits si no hay paridad.
Campo de Chequeo de Error
Chequeo de redundancia cíclica (CRC)

En redes como MAP o Modbus Plus, el protocolo de red maneja el entramado de mensajes con los delimitadores de inicio y fin que son específicos para la red.

Esos protocolos también manejan la entrega al dispositivo de destino, haciendo el campo de dirección Modbus, un mensaje innecesario para la transmisión actual (la dirección de Modbus es encaminada y convertida a una dirección del nodo de red por el controlador que lo origina o su adaptador de red).

2.3.2.1 Entramado ASCII

En modo ASCII, los mensajes comienzan con el caracter dos puntos (:) (en ASCII 3AH), y finalizan con un salto de línea y retorno de carro (CRLF, en ASCII 0DH y 0AH).

Los caracteres permisibles transmitidos para el resto de los campos son los hexadecimales 0... 9, A... F.

Los dispositivos de red supervisan el bus de red continuamente para encontrar el caracter dos puntos (:).

Cuando se recibe uno, cada dispositivo esclavo o maestro decodifica el siguiente campo (el campo de dirección) para descubrir si es el dispositivo direccionado.

Pueden transcurrir intervalos de hasta un segundo entre caracteres dentro del mensaje. Si ocurre un mayor intervalo, el dispositivo de recepción asume que ha ocurrido un error.

En la tabla 2.5 se muestra un mensaje típico de trama en modo ASCII.

Tabla 2.5. Formato de trama en modo ASCII

Modo ASCII					
Inicio	**Dirección**	**Función**	**Datos**	**Revisión LRC**	**Fin**
1 caracter	2 caracteres	2 caracteres	n caracteres	2 caracteres	2 caracteres

2.3.2.2 Entramado RTU

En el modo RTU los mensajes comienzan con un intervalo silencioso de al menos 3,5 tiempos de caracter. Esto es fácilmente implementado como un múltiplo de los tiempos del caracter en la velocidad que se está utilizando sobre la red (demostrado como T1-T2-T3-T4 en la figura de la página siguiente).

El primer campo entonces transmitido es la dirección del dispositivo.

Los caracteres permisibles transmitidos para todos los campos son los hexadecimales 0... 9, A... F. Los dispositivos de red supervisan el bus de la red continuamente, incluyendo también los intervalos “silenciosos”.

Cuando se recibe el primer campo (campo de dirección), cada dispositivo lo decodifica para descubrir si es el dispositivo direccionado. Después del último caracter transmitido, un intervalo similar de por lo menos de 3,5 tiempos del caracter marca el fin del mensaje. Un nuevo mensaje puede comenzar después

de este intervalo. La trama del mensaje completo debe ser transmitida como un flujo continuo.

Si un intervalo silencioso de más de 1,5 tiempos de caracter ocurre antes de la terminación de la trama, el dispositivo de recepción limpia el mensaje incompleto y asume que el siguiente byte será el campo de dirección de un nuevo mensaje.

De manera similar, si un nuevo mensaje comienza mucho antes que 3,5 tiempos de caracter que sigue a un mensaje previo, el dispositivo de recepción lo considerará una continuación del mensaje anterior. Esto fijará un error, pues el valor en el campo final del CRC no será válido para los mensajes combinados.

En la tabla 2.6 se muestra un mensaje típico de trama en modo RTU.

Tabla 2.6. Formato de trama en modo RTU

Modo RTU					
Inicio	**Dirección**	**Función**	**Datos**	**Revisión LRC**	**Fin**
T1-T2-T3-T4	8 bits	8 bits	n*8 bits	16 bits	T1-T2-T3-T4

2.3.3 Campo de Dirección

El campo de dirección de una trama de mensaje contiene dos caracteres (para el modo ASCII) u ocho bits (para el RTU). Las direcciones válidas de un dispositivo esclavo están en el rango decimal de 0 a 247. Los dispositivos esclavos individuales tienen asignadas direcciones en el rango de 1 a 247. Un maestro direcciona a un esclavo colocando la dirección de éste en el campo de dirección del mensaje. Cuando el esclavo envía la respuesta, éste coloca su propia dirección en el campo de dirección de la respuesta para indicar al maestro que esclavo está respondiendo.

La dirección 0 es utilizada para la dirección de mensajes de difusión (broadcast), que es reconocida por los dispositivos esclavos. Cuando el protocolo

Modbus es utilizado en niveles altos de red, el envío de mensajes de difusión (broadcast) podría no ser permitido o podría reemplazarse por otros métodos. Por ejemplo, Modbus Plus utiliza una base de datos global compartida que puede ser actualizada con cada rotación token.

2.3.4 Campo de Código de Función

El campo del código de función de una trama de mensaje contiene dos caracteres (para el modo ASCII) u ocho bits (para el modo RTU).

Los códigos válidos están en el rango decimal de 1 a 255. De estos, algunos son aplicables a todos los controladores Modicon mientras otros solo son aplicables a ciertos modelos, y algunos mas reservados a un uso futuro.

Cuando un mensaje es enviado desde un dispositivo maestro a otro esclavo, el campo de código de función dice al esclavo la clase de acción a realizar. Por ejemplo, leer los estados ON/OFF de un grupo de bobinas o entradas discretas; leer los datos contenidos en un grupo de registros; leer el estado de diagnóstico de un esclavo; escribir bobinas o registros señalados; o la verificación del programa dentro del esclavo.

Cuando el esclavo responde al maestro utiliza el campo del código de la función para indicar una respuesta normal (libre de error) o para indicar que ocurrió una cierta clase de error (llamada respuesta de excepción). Para una respuesta normal, el esclavo devuelve repitiendo simplemente el código de función original. Para una respuesta de excepción, el esclavo devuelve un código que es equivalente al código de función original con el bit más significativo fijado a un 1 lógico.

2.3.5 Campo de Datos

El campo de datos se construye usando sistemas de dos dígitos hexadecimales, en el rango de 00 a FF. Se realizan a partir de un par de

caracteres ASCII, o de un caracter RTU, según el modo de transmisión serial de la red.

El campo de datos de los mensajes enviados desde un maestro hacia un esclavo contiene la información adicional que el esclavo debe utilizar para tomar la acción definida por el código de función. Esto puede incluir puntos como direcciones de entradas discretas y de registros, la cantidad de puntos a ser manejados, y la cuenta de los bytes de datos actuales en el campo.

Por ejemplo, si el maestro solicita al esclavo leer un grupo registros holding (código de función 03H), el campo de datos especifica el registro de inicio y cuántos registros deben ser leídos.

Si el maestro escribe a un grupo de registros en el esclavo (código de función 10H), el campo de datos especifica el registro de inicio, cuántos registros va a escribir, la cuenta de bytes de datos a seguir en el campo de datos, y los datos que se escribirán en los registros.

Si no ocurre un error, el campo de datos de la respuesta de un esclavo hacia un maestro contiene los datos solicitados. Si ocurre un error, el campo contiene un código de excepción que la aplicación del maestro puede usar para determinar la próxima acción a tomarse. Por ejemplo, en una petición de un dispositivo maestro a un esclavo para responder con esta comunicación con un registro de acontecimiento, el esclavo no requiere ninguna información adicional. El código de la función solamente especifica la acción.

2.3.6 Campo de Revisión de Error

Se usa, para las redes estándar de Modbus, dos clases de métodos de revisión de error. El contenido del campo de revisión de error depende del método que se está utilizando. Cuando se utiliza el modo ASCII para realizar entramado de caracteres, el campo de revisión de error contiene dos caracteres ASCII. Los caracteres de revisión de error son el resultado del cálculo de un Chequeo de

Redundancia Longitudinal (LRC) que es realizado en el contenido del mensaje, sin tomar en cuenta el caracter de inicio dos puntos (:) y el de finalización CRLF.

Los caracteres LRC se añaden al mensaje como el último campo que precede a los caracteres CRLF. Cuando se utiliza, en cambio, el modo RTU para el entramado de caracteres, el campo de revisión de error contiene un valor de 16 bits implementado como dos bytes de 8 bits. El valor de la revisión de error es el resultado del cálculo de un Chequeo de Redundancia Cíclica (CRC) que es realizado en el contenido del mensaje.

El campo CRC se añade como el último campo en el mensaje. Cuando se hace esto, el byte de orden inferior del campo es añadido primero, seguido por el byte de orden superior. El byte de orden superior del CRC es el último a ser enviado en el mensaje.

Cuando los mensajes son transmitidos en redes seriales estándar Modbus, cada caracter o byte se envía en este orden (de izquierda a derecha):

Bit menos significativo (**LSB**) ... Bit más significativo (**MSB**)

Las tablas 2.7 y 2.8 muestran el entramado de caracteres ASCII y RTU, respectivamente.

Tabla 2.7. Formato de trama Modbus ASCII con y sin paridad

ASCII con revisión de paridad									
Inicio	1	2	3	4	5	6	7	Paridad	Parada

ASCII sin revisión de paridad									
Inicio	1	2	3	4	5	6	7	Parada	Parada

Las redes seriales estándar Modbus utilizan dos clases de revisión de error. La revisión de paridad (par o impar) puede ser opcionalmente aplicada a cada caracter. La revisión de trama (LRC o CRC) es aplicada al mensaje entero. La

revisión de caracter y la revisión de trama de mensaje son generadas en el dispositivo maestro y aplicados al contenido del mensaje antes de la transmisión.

Tabla 2.8. Formato de trama Modbus RTU con y sin paridad

RTU con revisión de paridad										
Inicio	1	2	3	4	5	6	7	8	Paridad	Parada

RTU sin revisión de paridad										
Inicio	1	2	3	4	5	6	7	8	Parada	Parada

El dispositivo esclavo revisa cada caracter y la trama del mensaje entero durante la recepción. El maestro es configurado por el usuario para esperar un intervalo predeterminado de tiempo antes de abortar la transacción. Este intervalo es fijado para ser lo suficientemente largo para cualquier respuesta normal de un esclavo.

Si el esclavo detecta un error de transmisión, el mensaje no será tomado en cuenta. El esclavo no construirá una respuesta al maestro. De este modo el tiempo de descanso expirará y permitirá al programa del maestro manejar el error. Si un mensaje es direccionado a un dispositivo esclavo no existente causará un tiempo de descanso.

Otras redes tales como MAP o Modbus Plus usan la revisión de trama en un nivel superior del contenido de un mensaje Modbus. En estas redes, los campos de revisión LRC o CRC de un mensaje Modbus no se aplica.

En caso de un error de transmisión, los protocolos de comunicación específicos a esas redes notifican al dispositivo original que ha ocurrido un error, y permite que se revise o aborte según como se haya preseleccionado. Si se entrega un mensaje y el dispositivo esclavo no puede responder, puede ocurrir un error de tiempo de descanso que se puede detectar por el programa del maestro.

Los usuarios pueden configurar los controladores para una revisión de paridad par o impar o para ninguna comprobación de paridad. Esto determinará la forma como el bit de paridad será fijado en cada carácter.

Si ninguna de las dos paridades se especifican, la cantidad de bits 1 será contada en la porción de datos de cada carácter (siete bits de datos en modo ASCII, u ocho para RTU). El bit de paridad será entonces fijado a un 0 o a un 1 para dar como resultado un número par o impar de bits 1.

Cuando se transmite el mensaje, el bit de paridad se calcula y se aplica a la trama de cada caracter. El dispositivo de recepción cuenta la cantidad de bits 1 y fija un error si no son iguales según lo configurado para ese dispositivo (todos los dispositivos sobre una red Modbus se deben configurar para utilizar el mismo método de revisión de paridad).

La comprobación de paridad puede detectar un error solamente si un número impar de bits es seleccionado en una trama de caracter durante la transmisión. Por ejemplo, si se emplea la comprobación de paridad impar, y dos bits 1 se pierden de un caracter que contiene tres bits 1, el resultado sigue siendo una cuenta impar de bits 1.

Si no se especifica ninguna comprobación de paridad, no se transmite ningún bit de paridad y ningún chequeo de paridad puede ser realizado. Un bit de parada adicional se transmite para completar la trama de caracter. En modo ASCII, los mensajes incluyen un campo de revisión de error que se basa en el método de Chequeo de Redundancia Longitudinal (LRC).

El campo LRC comprueba el contenido del mensaje, excluyendo el caracter de inicio dos puntos (:) y el par de finalización CRLF. Éste se aplica sin importar que método de chequeo de paridad se esté utilizando para caracteres individuales del mensaje.

El campo LRC es un byte conteniendo un valor binario de ocho bits. El valor de LRC es calculado por el dispositivo que transmite, el cual lo añade al mensaje. El

dispositivo de recepción calcula un LRC durante la recepción del mensaje y compara este valor calculado con el valor actual que recibió en el campo LRC. Si los dos valores no son iguales el resultado es un error.

El LRC es calculado agregando juntos bytes sucesivos de ocho bits del mensaje complementando entonces el resultado. Esto se realiza en el contenido del campo del mensaje ASCII excluyendo los caracteres de inicio de mensaje dos puntos (:) y el par de finalización CRLF.

En lógica de escalera, la función CKSM calcula un LRC del contenido del mensaje.

En modo RTU, los mensajes incluyen un campo de revisión de error que se basa en el método de Chequeo de Redundancia Cíclica (CRC).

El campo del CRC comprueba el contenido del mensaje entero. Se aplica sin importar el método de chequeo de paridad usado para los caracteres individuales del mensaje. El campo CRC tiene dos bytes conteniendo un valor binario 16 bits. El valor CRC es calculado por el dispositivo que transmite, que lo añade al mensaje.

El dispositivo de recepción recalcula un CRC durante la recepción del mensaje y compara el valor calculado con el valor actual que recibió en el campo CRC. Si dos de los valores no son iguales, el resultado es un error. El CRC comienza cargando primero un registro de 16 bits lleno de unos. Entonces el proceso comienza aplicando bytes sucesivos de ocho bits del mensaje al contenido actual del registro.

Solamente los ocho bits de datos en cada caracter son utilizados para generar el CRC. Los bits de inicio, parada y el de paridad, no se aplican al CRC.

Durante la generación del CRC, a cada caracter de ocho bits se aplica un OR exclusivo con el contenido del registro. Entonces el resultado cambia de dirección del bit menos significativo (LSB), con un cero llenado en la posición del bit más

significativo (MSB). Se extrae y se examina el LSB. Si el LSB fue un “1”, al registro se le aplica un OR exclusivo con el registro presente, fijando el valor. Si el LSB fue un “0”, el OR exclusivo no toma lugar. Este proceso se repite hasta que se han realizado ocho cambios.

Después del último cambio (octavo), al siguiente byte de ocho bits se le aplica un OR exclusivo con el valor del registro en actual, y el proceso se repite ocho veces más como ya se describió.

El contenido final del registro, después de que todos los bytes del mensaje han pasado por este proceso, es el valor del CRC. Cuando el CRC se añade al mensaje, el byte de orden inferior es añadido primero, seguido por el byte de orden superior.

En lógica de escalera, la función CKSM calcula un CRC para el contenido del mensaje.

2.4 CÓDIGOS DE FUNCIÓN DEL PROTOCOLO MODBUS[1]

En la tabla 2.9 se listan las funciones principales que pueden ser implementadas en el protocolo Modbus.

2.4.1 Función 01H: Lectura de bobinas

Esta función permite leer el estado de las salidas discretas (bobinas), comenzando desde la primera bobina cuya dirección es 0. Las bobinas 1-16 son direccionadas como 0-15.

Cuando una de ellas se encuentre encendida, la respuesta será un 1, caso contrario la respuesta será 0. La dirección de inicio que se encuentra en la pregunta corresponde a la dirección desde la cual se leerá el estado de las salidas discretas, y a continuación se especifica el número de salidas que serán leídas.

Tabla 2.9. Códigos de Función del Protocolo Modbus

Códigos de Función del Protocolo Modbus	
Función	**Descripción**
01H	Lectura de bobinas
02H	Lectura del estado de entrada
03H	Lectura de registros holding
04H	Lectura de registros de entrada
05H	Forzar la escritura de una bobina individual
06H	Programar un registro individual
07H	Lecturas del estado de excepción
08H	Funciones de diagnóstico
subfunción 00	Retorno de pedido de recuperación de información
subfunción 01	Reinicio de la opción comm
subfunción 02	Retorno del registro de diagnóstico
subfunción 03	Cambio del delimitador ASCII de entrada
subfunción 04	Modo de solo escucha forzada
subfunción 10	Borrado de contadores y registro de diagnóstico
subfunción 11	Retorno de cuenta de mensajes de bus
subfunción 12	Retorno de cuenta de error de bus de comm
subfunción 13	Retorno de cuenta de error de bus de excepción
subfunción 14	Retorno de cuenta de mensaje de esclavo
subfunción 15	Retorno de cuenta de no-respuesta de esclavo
subfunción 16	Retorno de cuenta NAK de esclavo
subfunción 17	Retorno de cuenta de esclavo ocupado
subfunción 18	Retorno de cuenta de caracter overrrun de bus
subfunción 19	Retorno de cuenta de error overrrun
subfunción 20	Borrado de bandera y contador de overrrun
subfunción 21	Toma / borrado de estadísticas Modbus plus
0BH	Llamada al contador del evento Comm
0CH	Llamada a la anotación del evento Comm
0FH	Forzar la escritura de múltiples bobinas
10H	Programar múltiples registros
11H	Reporte de ID del esclavo
14H	Lectura de la referencia general
15H	Escritura de la referencia general
16H	Registro enmascarado 4xxxx de escritura
17H	Registros 4xxxx de lectura / escritura
18H	Lectura de la cola FIFO

En la respuesta, el byte count representa la cantidad de grupos de 8 bobinas cuyos estados se leerán; es decir, si requerimos la lectura de más de 8 bobinas (y de menos de 17), el byte count será 2, si es de más de 16 el byte count es de 3. Las respuestas de los estados se muestran luego del byte count y corresponden a tantos bytes como muestre la cantidad del byte count. Los bytes de las bobinas que no se lean serán rellenados con ceros.

La tabla 2.10 presenta un ejemplo del formato de pregunta y respuesta para la lectura de bobinas.

Tabla 2.10. Formato de petición y respuesta para la lectura de bobinas

Lectura de bobinas			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	01	Función	01
Dir. Inicio (H)	00	Byte Count	02
Dir. Inicio (L)	00	Salida (B8 - B1)	01
Núm. Salidas (H)	00	Salida (B10 - B9)	03
Núm. Salidas (L)	0A	Rev. Error (LRC-CRC)	--
Rev. Error (LRC-CRC)	--		

2.4.2 Función 02H: Lectura del estado de entrada

De forma análoga a la función anterior, ésta permite leer el estado de las entradas discretas comenzando desde la primera cuya dirección es 0.

Cuando una de las entradas se encuentre energizada, la respuesta será un 1, caso contrario la respuesta será 0. Las entradas 1-16 son direccionados como 0-15.

La dirección de inicio que se encuentra en la pregunta corresponde a la dirección desde la cual se leerá el estado de las entradas discretas, y a continuación se especifica el número de entradas a ser leídas.

En la respuesta, el byte count representa la cantidad de grupos de 8 entradas cuyos estados se leerán. Las respuestas de los estados se muestran luego del byte count y corresponden a tantos bytes como muestre la cantidad del byte count. Los bytes de las entradas que no se lean serán rellenados con ceros.

La tabla 2.11 presenta un ejemplo del formato de pregunta y respuesta para la lectura de entradas discretas.

Tabla 2.11. Formato de petición y respuesta para la lectura de entradas digitales

Lectura de entradas digitales			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	02	Función	02
Dir. Inicio (H)	00	Byte Count	01
Dir. Inicio (L)	00	Datos (10002 - 10001)	03
Núm. Entradas (H)	00	Rev. Error (LRC-CRC)	--
Núm. Entradas (L)	02		
Rev. Error (LRC-CRC)	--		

2.4.3 Función 03H: Lectura de registros holding

Esta función lee el contenido de los registros holding en el esclavo. Al igual que las anteriores funciones, los registros 1-16 son direccionados como 0-15.

En la petición se incluye la dirección de inicio y el número de registros a ser leídos. El byte count de la respuesta se calcula con el número de registros según sea múltiplo de 8; y los datos de respuesta se colocan después del byte count (tantos como el byte count indique). En la respuesta se rellena con ceros los datos que no sean leídos.

La tabla 2.12 presenta un ejemplo del formato de pregunta y respuesta para la lectura de registros holding.

Tabla 2.12. Formato de petición y respuesta para la lectura de registros holding

Lectura de registros holding			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	03	Función	03
Dir. Inicio (H)	00	Byte Count	02
Dir. Inicio (L)	00	Datos (H) (Reg. 40001)	AA
Núm. Registros (H)	00	Datos (L) (Reg. 40001)	00
Núm. Registros (L)	01	Rev. Error (LRC-CRC)	--
Rev. Error (LRC-CRC)	--		

2.4.4 Función 04H: Lectura de registros de entrada

Esta función se utiliza para leer el estado de los registros de entrada del esclavo especificado. Los registros 1-16 son direccionados como 0-15.

En la petición se coloca la dirección de inicio y el número de registros de entrada a leerse. En la respuesta, se calcula el byte count y luego los datos pertenecientes a los registros de entrada que se desean leer, tal como puede verse en el ejemplo del formato de petición y respuesta de la tabla 2.13.

Tabla 2.13. Formato de petición y respuesta para la lectura de registros de entrada

Lectura de registros de entrada			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	04	Función	04
Dir. Inicio (H)	00	Byte Count	02
Dir. Inicio (L)	05	Datos (H) (Reg. 30006)	12
Núm. Registros (H)	00	Datos (L) (Reg. 30006)	0B
Núm. Registros (L)	01	Rev. Error (LRC-CRC)	--
Rev. Error (LRC-RC)	--		

2.4.5 Función 05H: Forzar la escritura de una bobina individual

Esta función permite la escritura de una única salida (bobina). La bobina 1 es direccionada como 0. La respuesta de esta función es un eco de la petición, es decir, devuelve la misma trama que le fue entregada, tal como puede verse en el ejemplo del formato de petición y respuesta de la tabla 2.14.

2.4.6 Función 06H: Escribir un registro individual

Esta función sirve para escribir individualmente un registro interno. El registro 1 se direcciona como 0. La respuesta de esta función es un eco de la petición, es decir, devuelve la misma trama que fue enviada al esclavo, tal como se puede apreciar en el ejemplo del formato de petición y respuesta de la tabla 2.15.

Tabla 2.14. Formato de petición y respuesta para escribir una sola bobina

Escribir una bobina individual			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	05	Función	05
Dir. Bobina (H)	00	Dir. Bobina (H)	00
Dir. Bobina (L)	00	Dir. Bobina (L)	00
Dato forzado (H)	FF	Dato forzado (H)	FF
Dato forzado (L)	00	Dato forzado (L)	00
Rev. Error (LRC-CRC)	--	Rev. Error (LRC-CRC)	--

Tabla 2.15. Formato de petición y respuesta para la escritura de un registro individual

Escribir un registro individual			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	06	Función	06
Dir. Registro (H)	00	Dir. Registro (H)	00
Dir. Registro (L)	00	Dir. Registro (L)	00
Dato (H)	00	Dato (H)	00
Dato (L)	03	Dato (L)	03
Rev. Error (LRC-CRC)	--	Rev. Error (LRC-CRC)	--

2.4.7 Función 07H: Lecturas del estado de excepción

Esta función permite leer el contenido de las ocho bobinas del estado de excepción en un controlador esclavo.

Tabla 2.16. Distribución de las bobinas del estado de excepción de acuerdo a los controladores

Modelo del controlador	Bobinas	Tarea
M84, 184/384, 584, 984	1-8	Definido por el usuario
484	257	Estado de la batería
	258-264	Definido por el usuario
884	761	Estado de la batería
	762	Estado protegido de la memoria
	763	Estado RIO healt
	764-768	Definido por el usuario

En cada controlador, las bobinas tienen una tarea predefinida. Otras bobinas pueden ser programadas por el usuario para mantener la información acerca del estado del controlador, por ejemplo “máquina ON/OFF”, “cabezas desplegadas”, “seguridades satisfechas”, “existen condiciones de error”, u otras banderas definidas por el usuario.

Esta función provee una forma simple de acceder a esta información, debido a que las referencias a las bobinas de excepción son conocidas. Las tareas predefinidas de las bobinas de excepción se presentan en la tabla 2.16.

La petición tan solo envía la función y la respuesta lo hace con el dato de las bobinas, así como se muestra en el ejemplo de la tabla 2.17.

Tabla 2.17. Formato de petición y respuesta para la lectura del estado de excepción

Lectura del estado de excepción			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	07	Función	07
Rev. Error (LRC-CRC)	--	Dato de bobina	6D
		Rev. Error (LRC-CRC)	--

2.4.8 Función 08H: Funciones de diagnóstico

La función de diagnóstico proporciona una serie de pruebas para revisar el sistema de comunicación entre maestro y esclavo o para chequear varias condiciones internas de error dentro del esclavo.

La función utiliza un código de subfunción en un campo de dos bytes para definir el tipo de prueba que va a ser realizada. El esclavo hace un eco tanto del código de función como del de subfunción en una respuesta normal.

Muchas de las recuperaciones de diagnóstico utilizan un campo de datos de dos bytes para enviar datos de diagnóstico o información de control al esclavo.

Algunos diagnósticos causan que los datos sean devueltos desde el esclavo en el campo de datos de una respuesta normal. En general, emitir una función de diagnóstico a un dispositivo esclavo no afecta la ejecución del programa usuario en el esclavo.

Un dispositivo esclavo puede, sin embargo, ser forzado al modo “Solo escucha” en el cual monitoreará los mensajes sobre el sistema de comunicaciones pero no los responderá. Esto puede afectar el resultado de su programa de aplicación si depende sobretodo de cualquier futuro intercambio de datos con el dispositivo esclavo. Generalmente, el modo obliga a retirar un dispositivo esclavo que no funcione bien en el sistema de comunicación. Si bien no se tratará en detalle las subfunciones, conviene decir que es un tema importante cuando la aplicación que se va a realizar es más compleja y requiere verificar datos y errores de distintos tipos.

En la tabla 2.9. se enlistan los códigos de subfunción que soporta Modbus con su descripción, para tener una idea más amplia de todo lo que es posible diagnosticar con este grupo de funciones.

2.4.9 Función 0BH: Llamada al contador del evento Comm

Retorna una palabra de estado y un evento de cuenta desde el contador de evento de comunicaciones de un esclavo.

Usando esta llamada, se cuenta antes y después una serie de mensajes, y un maestro puede determinar si los mensajes fueron manejados normalmente por el esclavo.

El contador de evento del controlador es incrementado una vez por cada mensaje finalizado exitosamente. No es incrementado por respuestas de excepción, comandos de sondeo o comandos de llamada al contador de evento.

El contador de evento puede restaurarse por medio de la función de diagnóstico (código 08H), con una subfunción de Arranque de Opción de

Comunicaciones (código 0001H) o Clear Counters y Registro de Diagnóstico (código 000AH).

La respuesta normal contiene una palabra de estado de dos bytes, y una cuenta de evento de dos bytes. La palabra de estado será FFFFH, si un comando de programa previamente emitido está aún siendo procesado por el esclavo (es decir, existe una condición de ocupado). De otra manera, la palabra de estado se llenará con ceros.

La tabla 2.18 presenta un ejemplo del formato de pregunta y respuesta para la llamada al contador del evento comm.

Tabla 2.18. Formato de petición y respuesta para la llamada al contador del evento comm

Llamada al contador del evento comm			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	0B	Función	0B
Rev. Error (LRC-CRC)	--	Estado (H)	FF
		Estado (L)	FF
		Contador de evento (H)	01
		Contador de evento (L)	0B
		Rev. Error (LRC-CRC)	--

2.4.10 Función 0CH: Llamada a la anotación del evento Comm

Retorna una palabra de estado, cuenta de evento, cuenta de mensaje y un campo de bytes del evento desde el esclavo.

La palabra de estado y la cuenta del evento son idénticas a las retornadas por la función 0BH. El contador de mensaje contiene la cantidad de mensajes procesados por el esclavo desde la última vez que se reinició. El campo de bytes del evento contiene un conjunto de entre 0 y 64 bytes, con cada byte que corresponde al estado de una operación Modbus de envío o recepción para el esclavo. Los eventos son ingresados por el esclavo dentro del campo en orden

cronológico. El byte 0 es el evento más reciente. Cada nuevo byte reemplaza al byte más antiguo desde el campo.

La tabla 2.19 presenta un ejemplo del formato de pregunta y respuesta para la anotación del evento comm.

Tabla 2.19. Formato de petición y respuesta para la llamada a la anotación del evento comm

Llamada a la anotación del evento comm			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	0C	Función	0C
Rev. Error (LRC-CRC)	--	Byte Count	0B
		Estado (H)	00
		Estado (L)	00
		Contador de evento (H)	01
		Contador de evento (L)	0B
		Contador de mensaje ()	01
		Contador de mensaje (L)	21
		Evento 0	20
		Evento 1	00
		Rev. Error (LRC-CRC)	--

2.4.11 Función 0FH: Forzar la escritura de múltiples bobinas

Obliga a cada bobina (referencia 0X) en una secuencia a tomar un valor de ON u OFF. Cuando existe un mensaje de difusión (broadcast), la función obliga la misma referencia de las bobinas en todos los esclavos conectados.

En la petición, un 1 en una posición de bit del campo, solicita a la correspondiente bobina a adoptar el estado ON, así también como un 0 a adoptar un estado OFF. La respuesta normal devuelve la dirección del esclavo, código de función, dirección de inicio y la cantidad de bobinas escritas.

La tabla 2.20 presenta un ejemplo del formato de pregunta y respuesta para la escritura de múltiples bobinas.

Tabla 2.20. Formato de petición y respuesta para la escritura de múltiples bobinas

Escribir múltiples bobinas			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	0F	Función	0F
Dir. Bobinas (H)	00	Dir. Bobinas (H)	00
Dir. Bobinas (L)	00	Dir. Bobinas (L)	00
Núm. Bobinas (H)	00	Núm. Bobinas (H)	00
Núm. Bobinas (L)	09	Núm. Bobinas (L)	09
Byte count	02	Rev. Error (LRC-CRC)	--
Datos forzados (H)	01		
Datos forzados (L)	E5		
Rev. Error (LRC-CRC)	--		

2.4.12 Función 10H: Programar múltiples registros

Sirve para escribir los valores dentro de una secuencia de registros holding (referencias 4X). Cuando existe difusión (broadcast), la función escribe las mismas referencias de registros en todos los esclavos conectados.

El mensaje de pregunta especifica las referencias del registro a ser programado. Siguiendo la misma lógica que todas las funciones anteriores, los registros son direccionados comenzando desde cero, es decir, el registro 1 es direccionado como 0.

La respuesta normal devuelve la dirección del esclavo, el código de función, la dirección de inicio y la cantidad de registros escritos, tal como se aprecia en el ejemplo de petición y respuesta de la tabla 2.21.

2.4.13 Función 11H: Reporte de ID del esclavo

Devuelve una descripción del tipo de controlador presente en la dirección del esclavo, el estado presente del indicador Run del esclavo y otra información específica del dispositivo esclavo, como se observa en el ejemplo de petición y respuesta de tabla 2.22.

Tabla 2.21. Formato de petición y respuesta para la escritura de múltiples registros

Escribir múltiples registros			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	10	Función	10
Dir. Registros (H)	00	Dir. Registros (H)	00
Dir. Registros (L)	00	Dir. Registros (L)	00
Núm. Registros (H)	00	Núm. Registros (H)	00
Núm. Registros (L)	01	Núm. Registros (L)	01
Byte count	02	Rev. Error (LRC-CRC)	--
Datos (H)	01		
Datos (L)	E5		
Rev. Error (LRC-CRC)	--		

Tabla 2.22. Formato de petición y respuesta para el reporte de ID del esclavo

Reporte de ID del esclavo			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	11	Función	11
Rev. Error (LRC-CRC)	--	Byte Count	dep.
		Dirección del esclavo	dep.
		Estado de indicador RUN	00
		Datos adicionales	dep.
		Rev. Error (LRC-CRC)	--

2.4.14 Función 14H: Lectura de la referencia general

Devuelve el contenido de los registros en referencias de archivo de Memoria Extendida (6XXXXX). El mensaje de difusión (broadcast) no es permitido. La función puede leer múltiples grupos de referencias. Los grupos pueden separarse (no contiguamente), pero las referencias dentro de cada grupo deben ser secuenciales. La pregunta contiene la dirección estándar del esclavo Modbus, el código de función, la cuenta de byte y los campos de chequeo de error. El resto de la pregunta especifica el grupo o grupos de referencias a leerse. Cada grupo es definido en un campo separado, “sub-peticiones”, el cual contiene 7 bytes.

La respuesta normal es una serie de sub-respuestas, una por cada sub-petición. El campo de cuenta de byte es la cuenta total combinada de bytes en todas las sub-respuestas. Cada sub-respuesta contiene un campo que muestra su propia cuenta de byte. La tabla 2.23 presenta un ejemplo del formato de pregunta y respuesta para la lectura de la referencia general.

Tabla 2.23. Formato de petición y respuesta para la lectura de la referencia general

Lectura de la referencia general			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	14	Función	14
Byte Count	0E	Byte Count	0C
Sub pet1. Tipo ref.	06	Sub res1. Byte count	05
Sub pet1. Núm. Archivo (H)	00	Sub res1. Tipo ref.	06
Sub pet1. Núm. Archivo (L)	04	Sub res1. Dato registrado (H)	0D
Sub pet1. Dir. Inicio (H)	00	Sub res1. Dato registrado (L)	FE
Sub pet1. Dir. Inicio (L)	01	Sub res1. Dato registrado (H)	00
Sub pet1. Contador de registro (H)	00	Sub res1. Dato registrado (L)	20
Sub pet1. Contador de registro (L)	02	Sub res2. Byte count	05
Sub pet2. Tipo ref.	06	Sub res2. Tipo ref.	06
Sub pet2. Núm. Archivo (H)	00	Sub res2. Dato registrado (H)	33
Sub pet2. Núm. Archivo (L)	03	Sub res2. Dato registrado (L)	CD
Sub pet2. Dir. Inicio (H)	00	Sub res2. Dato registrado (H)	00
Sub pet2. Dir. Inicio (L)	09	Sub res2. Dato registrado (L)	40
Sub pet2. Contador de registro (H)	00	Rev. Error (LRC-CRC)	--
Sub pet2. Contador de registro (L)	02		
Rev. Error (LRC-CRC)	--		

2.4.15 Función 15H: Escritura de la referencia general

Escribe el contenido de los registros en referencias de archivo de Memoria Extendida (6XXXXX). Esta función no soporta el mensaje de difusión (broadcast).

La función puede leer múltiples grupos de referencias. Los grupos pueden ser separados (no contiguamente), pero las referencias dentro de cada grupo deben ser secuenciales.

La pregunta contiene la dirección estándar del esclavo Modbus, el código de función, la cuenta de byte y los campos de chequeo de error. El resto de la pregunta especifica el grupo o grupos de referencias a escribirse y los datos a ser escritos dentro de ellos. Cada grupo es definido en un campo separado, “sub-peticiones”, el cual contiene 7 bytes más de datos.

La respuesta normal es un eco de la pregunta. La tabla 2.24 presenta un ejemplo del formato de pregunta y respuesta para la escritura de la referencia general.

Tabla 2.24. Formato de petición y respuesta para la escritura de la referencia general

Escritura de la referencia general			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	15	Función	15
Byte Count	0D	Byte Count	0D
Sub pet1. Tipo ref.	06	Sub pet1. Tipo ref.	06
Sub pet1. Núm. Archivo (H)	00	Sub pet1. Núm. Archivo (H)	00
Sub pet1. Núm. Archivo (L)	04	Sub pet1. Núm. Archivo (L)	04
Sub pet1. Dir. Inicio (H)	00	Sub pet1. Dir. Inicio (H)	00
Sub pet1. Dir. Inicio (L)	07	Sub pet1. Dir. Inicio (L)	07
Sub pet1. Registro de cuenta (H)	00	Sub pet1. Registro de cuenta (H)	00
Sub pet1. Registro de cuenta (L)	03	Sub pet1. Registro de cuenta (L)	03
Sub pet1. Registro de datos (H)	06	Sub pet1. Registro de datos (H)	06
Sub pet1. Registro de datos (L)	AF	Sub pet1. Registro de datos (L)	AF
Sub pet1. Registro de datos (H)	04	Sub pet1. Registro de datos (H)	04
Sub pet1. Registro de datos (L)	BE	Sub pet1. Registro de datos (L)	BE
Sub pet1. Registro de datos (H)	10	Sub pet1. Registro de datos (H)	10
Sub pet1. Registro de datos (L)	0D	Sub pet1. Registro de datos (L)	0D
Rev. Error (LRC-CRC)	--	Rev. Error (LRC-CRC)	--

2.4.16 Función 16H: Registro enmascarado 4xxxx de escritura

Modifica el contenido de un registro 4XXXX específico usando una combinación de una máscara AND, una máscara OR y el contenido actual del registro. La función puede ser usada para poner a uno o limpiar bits individuales en el registro. Esta función tampoco soporta el mensaje de difusión (broadcast).

La pregunta especifica la referencia 4XXXX a escribirse, los datos a usarse como máscara AND y los datos a usarse como máscara OR. El algoritmo de la función es:

Resultado = (Contenido Presente AND Máscara_and) OR (Máscara_or AND Máscara _and negada).

La respuesta normal es un eco de la pregunta. La respuesta es devuelta después de que el registro ha sido escrito.

La tabla 2.25 presenta un ejemplo del formato de pregunta y respuesta para el registro enmascarado 4xxxx de escritura.

Tabla 2.25. Formato de petición y respuesta para el registro enmascarado 4xxxx de escritura

Registro enmascarado 4xxxx de escritura			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	16	Función	16
Dirección de referencia (H)	00	Dirección de referencia (H)	00
Dirección de referencia (L)	04	Dirección de referencia (L)	04
Máscara AND (H)	00	Máscara AND (H)	00
Máscara AND (L)	F2	Máscara AND (L)	F2
Máscara OR (H)	00	Máscara OR (H)	00
Máscara OR (L)	25	Máscara OR (L)	25
Rev. Error (LRC-CRC)	--	Rev. Error (LRC-CRC)	--

2.4.17 Función 17H: Registros 4xxxx de lectura / escritura

Realiza una combinación de una operación de lectura y escritura en una única transacción Modbus. Esta función no soporta mensaje de difusión (broadcast) y solamente está implementada en el controlador 984-785, como se muestra en el ejemplo del formato de pregunta y respuesta de la tabla 2.26.

Tabla 2.26. Formato de petición y respuesta para los registros 4xxxx de lectura / escritura

Registros 4xxxx de lectura / escritura			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	14	Función	14
Byte Count	0E	Byte Count	0C
Sub pet1. Tipo ref.	06	Sub res1. Byte count	05
Sub pet1. Núm. Archivo (H)	00	Sub res1. Tipo ref.	06
Sub pet1. Núm. Archivo (L)	04	Sub res1. Dato registrado (H)	0D
Sub pet1. Dir. Inicio (H)	00	Sub res1. Dato registrado (L)	FE
Sub pet1. Dir. Inicio (L)	01	Sub res1. Dato registrado (H)	00
Sub pet1. Contador de registro (H)	00	Sub res1. Dato registrado (L)	20
Sub pet1. Contador de registro (L)	02	Sub res2. Byte count	05
Sub pet2. Tipo ref.	06	Sub res2. Tipo ref.	06
Sub pet2. Núm. Archivo (H)	00	Sub res2. Dato registrado (H)	33
Sub pet2. Núm. Archivo (L)	03	Sub res2. Dato registrado (L)	CD
Sub pet2. Dir. Inicio (H)	00	Sub res2. Dato registrado (H)	00
Sub pet2. Dir. Inicio (L)	09	Sub res2. Dato registrado (L)	40
Sub pet2. Contador de registro (H)	00	Rev. Error (LRC-CRC)	--
Sub pet2. Contador de registro (L)	02		
Rev. Error (LRC-CRC)	--		

2.4.18 Función 18H: Lectura de la cola FIFO

Lee el contenido de una cola FIFO (Primero en Entrar Primero en Salir o First Input First Output, en inglés) de los registros 4XXXX. La función devuelve una cuenta de los registros en la cola, seguido por una fila de datos.

Pueden leerse más de 32 registros: la cuenta, más allá de 31 registros de datos enfilados. El registro de cuenta de cola es devuelto primero, seguido por el registro de datos en fila.

La función lee el contenido en cola, pero no lo puede limpiar. El mensaje de difusión (broadcast) no es soportado por esta función. La pregunta especifica el inicio de la referencia 4XXXX a leerse desde la cola FIFO.

En una respuesta normal, la cuenta de byte muestra la cantidad de bytes a seguir, los bytes de cuenta de cola y los bytes de registro de datos (pero no

incluyen el campo de chequeo de error). La cuenta de cola es la cantidad de registros de datos en la cola (no incluye el registro de cuenta).

Si la cuenta de cola excede a 31, una respuesta de excepción es devuelta con un código de error de 03 (Valor de Datos Ilegal).

Esta función es soportada solamente por el controlador 984-785. La tabla 2.27 presenta un ejemplo del formato de pregunta y respuesta para la lectura de la cola FIFO.

Tabla 2.27. Formato de petición y respuesta para la lectura de la cola FIFO

Lectura de la cola FIFO			
Petición		**Respuesta**	
Nombre del campo	**Hex**	**Nombre del campo**	**Hex**
Dir. Esclavo	01	Dir. Esclavo	01
Función	18	Función	18
Dirección del puntero FIFO (H)	04	Byte count (H)	00
Dirección del puntero FIFO (L)	DE	Byte count (L)	0B
Rev. Error (LRC-CRC)	--	FIFO count (H)	00
		FIFO count (L)	03
		FIFO data reg. 1 (H)	01
		FIFO data reg. 1 (L)	BB
		FIFO data reg. 2 (H)	12
		FIFO data reg. 2 (L)	84
		FIFO data reg. 3 (H)	13
		FIFO data reg. 3 (L)	22
		Rev. Error (LRC-CRC)	--

Referencias:

1. http://www.modbus.org, Modbus Protocol Reference Guide.pdf y Modbus over Serial Line-Specification and Implementation Guide V1.0.pdf.

2. http://www.circuitcellar.com, RS-485 Data Interface.pdf, The Art and Science of RS-485.pdf y Designing RS-485 Circuits.pdf.

CAPÍTULO 3

DISEÑO DEL SOFTWARE DEL SISTEMA

3.1 INTRODUCCIÓN

Ocho son los Códigos de Función del Protocolo Modbus implementados en este proyecto, 01H a 06H, 0FH y 10H, la mayoría de ellos interactúan directamente con el medio físico, entre ellos, se pueden mencionar la escritura de bobinas, la lectura de entradas digitales, bobinas y entradas analógicas.

Los códigos de función que no interactúan físicamente con el medio son los ligados a la lectura y escritura de registros holding, ya que todo su manejo se hace mediante software. Sin embargo, se utilizaron algunos de ellos para manejar ciertas características propias del PIC, como son, el gobierno del Timer1 y la señal PWM, al igual que los registros internos asociados a su funcionamiento.

La figura 3.1 resume en forma de bloques la disposición física de la placa según los códigos de función implementados en ella, el bloque denominado LCD no solo ayudará a la lectura de los registros internos del microcontrolador, sino también a la configuración de los parámetros de comunicación, y a la visualización de cualquier variable enlazada a un registro holding reservado para esta tarea, y el bloque de comunicaciones, representa el circuito integrado MAX232 y el conector RJ-45 que servirán para comunicar la placa con un dispositivo maestro.

La nomenclatura que identifica a cada uno de los dispositivos electrónicos a los que se hace referencia en este capítulo, es la misma utilizada en el diagrama del

circuito de la tarjeta (ANEXO B). La tabla 3.1 muestra en forma general las características que posee la placa para la cual se diseñó el software.

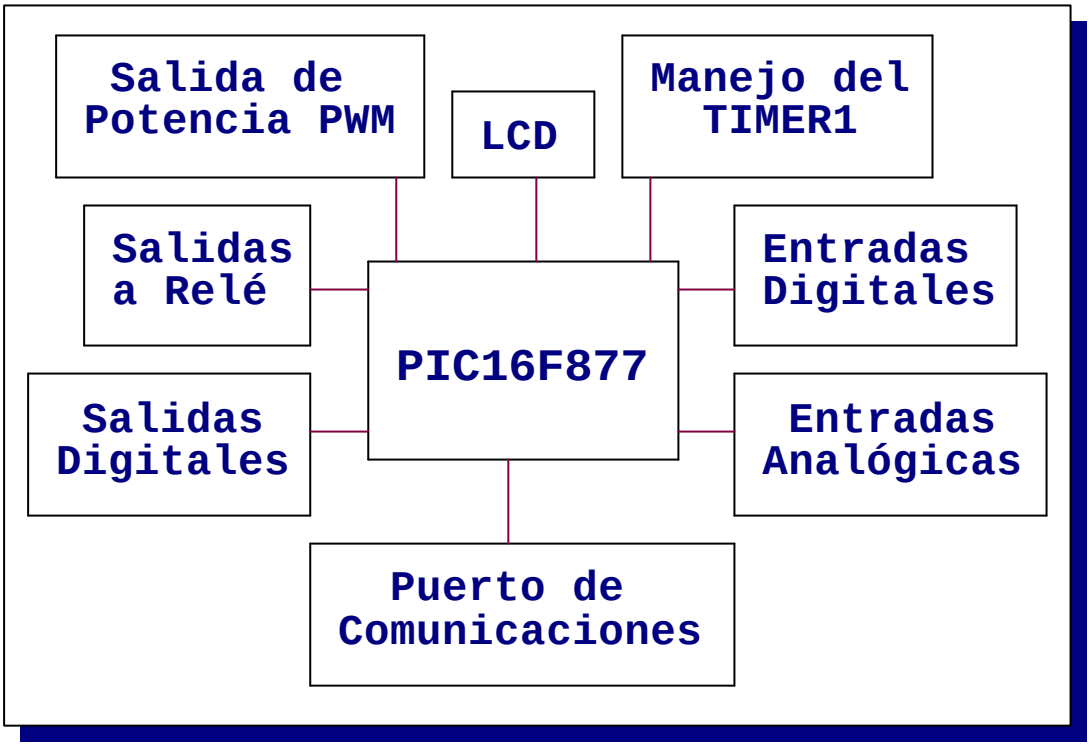

Figura 3.1. Diagrama de bloques de la tarjeta implementada

Tabla 3.1. Características de salidas, entradas y registros de la tarjeta microcontrolada

Características generales de la tarjeta implementada		
S / E / R	**Tipo**	**Dirección**
Salidas (10)	4 a relé	1 - 4
	6 digitales	5 - 10
Entradas (10)	6 digitales	10001 - 10006
	4 analógicas	30006 - 30009
Registros (11)	6 de propósito general	40001 - 40006
	5 para funciones predeterminadas.	40007 - 40011

Todo el programa alojado en el microcontrolador se escribió utilizando un paquete de software desarrollado por microEngineering, llamado PICBasic Pro[1].

Con respecto a los diagramas de flujo expuestos, aquellos son solo el comienzo a un examen más profundo descrito posteriormente, sin embargo, dan las bases suficientes para crear una idea general de funcionamiento del código de programación diseñado. En el programa implementado en el PIC16F877[2] se describe en forma detallada las variables utilizadas, los registros propios del PIC empleados para la gestión eficiente de las tramas de petición del maestro, y la respectiva construcción de las tramas de respuesta.

Todo el diseño de la Interfase Humano Máquina (HMI) se realizó utilizando la versión 3.8 de un paquete de software desarrollado por la National Instruments, empleado en automatización industrial, llamado Lookout[3]. Se inicia explicando el objeto Modbus alrededor del cual toda la interfase se desarrolla. Continúa con los diferentes objetos necesarios para controlar las funciones que el protocolo Modbus ofrece, y analiza más profundamente la escritura y lectura de los registros internos del PIC, T1CON, TMR1H:TMR1L, T2CON y PR2.

3.2 DISEÑO DEL PROGRAMA ALOJADO EN EL MICROCONTROLADOR

3.2.1 Salidas (Bobinas) y Entradas Digitales

El manejo de bobinas y entradas digitales se hace por medio de los arreglos est y est1, respectivamente. A cada bobina y entrada digital se le asigna el elemento de un arreglo, de este modo su direccionamiento es muy sencillo de implementar y fácilmente escalable, suponiendo que se requiere un mayor número de bobinas o entradas digitales a las declaradas en este proyecto.

3.2.2 Entradas Analógicas[4]

El microcontrolador PIC16F877 posee un conversor análogo digital de 10 bits de resolución (el valor máximo de conversión es 1023 para una entrada analógica de 5V) y 8 canales de entrada, de los que solo se utilizan 4 para este proyecto. El registro ADCON0 controla la operación del conversor A/D, los bits ADCON0<7:6> sirven para seleccionar la frecuencia de reloj que se emplea en la conversión.

Se designa como T_{AD} el tiempo que dura la conversión de cada bit. En el caso de trabajar con valores digitales de 10 bits, se requiere un tiempo mínimo de $12*T_{AD}$. El valor de T_{AD} seleccionado nunca debe ser menor de 1,6 microsegundos. El valor escogido para este proyecto de tesis fue un $T_{AD} = 32 * T_{OSC}$, utilizando un cristal de 4 MHz.

El valor que debe adoptar el registro ADCON0 para que la lectura de una señal analógica de interés por parte del maestro sea correctamente escogida, se realiza direccionando un arreglo reservado únicamente para este propósito, llamado Ch.

El bit ADFM del registro ADCON1 selecciona el formato del resultado de la conversión, empleándose para este caso justificación a la derecha. Por lo tanto, el valor completo del registro ADRESL se coloca en uno de los elementos del arreglo peticion y el valor de los dos bits de menos peso del registro ADRESH, en otro elemento de mismo arreglo contiguo al anterior.

Los bits PCFG3-0 de ADCON1 se usan para configurar las patitas de los canales de entrada al conversor. Si un dispositivo maestro requiere de la placa implementada el valor de alguna de sus entradas analógicas, los bits PCFG3-0 toman el valor de 0000 (todas las patitas se convierten en entradas analógicas) caso contrario el valor de 0110 (todas las patitas se convierten en entradas o salidas digitales).

3.2.3 Registros Holding[4]

El microcontrolador maneja once registros holding gestionados internamente mediante software. Los primeros seis registros son de propósito general y pueden ser utilizados para almacenar variables de interés en un determinado proceso. Los cinco registros restantes guardan relación con el despliegue de información tanto interna como externa, así como del control de algunas características propias del microcontrolador.

La tabla 3.2 muestra con más detalle todo lo descrito anteriormente.

Tabla 3.2. Asignación de registros holding al microcontrolador

Asignación de registros holding al microcontrolador	
Registro	**Descripción**
40001 a 40006	Propósito general
40007	Escritura de TMR1H:TMR1L
40008	Salida de Potencia PWM
40009	Visualización en LCD de una variable enlazada a este registro
40010	Escritura de T1CON, T2CON y PR2
40011	Lectura de T1CON, TMR1H:TMR1L, T2CON y PR2

Los registros holding tienen una longitud de 16 bits, siendo necesarios dos arreglos (registro1 y registro) para pasar adecuadamente sus datos y poder direccionarlos cómodamente.

3.2.3.1 Timer1

El TMR1 es el único Temporizador / Contador ascendente con un tamaño de 16 bits, lo que requiere el uso de dos registros concatenados de 8 bits: TMR1H:TMR1L, que son los encargados de guardar el valor de conteo de cada momento.

El valor contenido en TMR1H:TMR1L puede ser leído o escrito, y los impulsos de reloj que originan el conteo ascendente pueden provenir del exterior o de la frecuencia de funcionamiento del microcontrolador (F_{osc} / 4).

El TMR1 es capaz de funcionar de las tres formas siguientes:

1. Como temporizador.
2. Como contador síncrono
3. Como contador asíncrono

El funcionamiento del TMR1 está gobernado por el valor con el que se programan los bits del registro T1CON.

En el modo temporizador el valor concatenado TMR1H:TMR1L se incrementa con cada ciclo de instrucción (F_{osc} / 4).

En el modo contador síncrono, el incremento se puede producir con los flancos ascendentes de un reloj externo, cuya entrada se aplica a las líneas PORTC.0 y PORTC.1 del microcontrolador.

En el modo contador asíncrono, el incremento se puede producir por los impulsos aplicados en la línea PORTC.0 del microcontrolador.

3.2.3.2 Timer2[5]

Se trata de un temporizador ascendente de 8 bits que se puede leer y escribir, y que también puede realizar funciones especiales para la Puerta Serie Síncrona (SSP) y con los módulos de captura y comparación.

La señal de reloj del TMR2 es interna: F_{osc} / 4, y antes de ser aplicada pasa por un predivisor de frecuencia con rangos de 1:1, 1:4 y 1:16. La salida del TMR2 atraviesa un postdivisor de frecuencia con rangos de división desde 1:1 a 1:16, pasando por los 16 valores posibles.

Para controlar el funcionamiento del TMR2 se usa el registro T2CON.

3.2.3.2.1 Modo PWM

En este modo de trabajo se consiguen impulsos lógicos cuya duración del ancho del nivel alto es variable. Esta característica es de enorme aplicación en el control de dispositivos tan populares como motores y triacs.

El pin RC2 / CCP1 del PIC16F877 está configurado como salida y bascula entre los niveles lógicos 0 y 1 a intervalos variables de tiempo. Lo que se intenta es obtener un impulso cuyo nivel alto tenga una anchura variable (ciclo de trabajo) dentro del intervalo del periodo de trabajo. El TMR2 está íntimamente relacionado con el funcionamiento de la salida PWM.

3.2.3.2.2 Periodo PWM

El periodo PWM se escribe en el registro PR2 y está calculado usando la siguiente fórmula:

$$T_{PWM} = [(PR2) + 1] * 4 * T_{OSC} * PD$$

3.2.3.2.3 Ciclo de Trabajo PWM

El ciclo de trabajo PWM se escribe en el registro CCPR1L y los bits CCP1CON<5:4>. La longitud (bits) que la señal PWM puede manejar es de 10 bits, permaneciendo contenidos en el registro CCPR1L los ocho bits más significativos y en CCP1CON<5:4> los dos bits menos significativos. La siguiente ecuación es usada para calcular el ciclo de trabajo PWM:

$$CT = (CCPR1L : CCP1CON < 5 : 4 >) * T_{OSC} * PD$$

Otra forma de escribir las mismas ecuaciones es la siguiente:

$$PR2 = [F_{OSC} / (4 * PS * F_{PWM})] - 1$$

$$CCPR1L : CCP1CON < 5 : 4 >= (PR2 + 1) * 4 * CT\%$$

La máxima resolución PWM (bits) para una Frecuencia PWM, está dada por la siguiente fórmula:

$$\text{Re } solución = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} bits$$

Donde:

F_{osc} = Frecuencia de Reloj

PD = Valor del Predivisor de Frecuencia del Timer2 (T2CON<1:0>)

F_{PWM} = Frecuencia PWM

T_{PWM} = Periodo PWM

CT% = Ciclo de Trabajo (Ej 20% = 0.2).

Este proyecto utiliza una Frecuencia PWM igual a 500Hz y el Predivisor de frecuencia del TMR2 igual a 1:16. Por lo tanto, PR2 es igual a 124.

El valor máximo de CCPR1L:CCP1CON<5:4> sucede cuando el Ciclo de Trabajo es del 100%. De esta manera, CCPR1L:CCP1CON<5:4> máximo es igual a 500. Esto quiere decir, que los valores enviados al microcontrolador no deben superar el valor de 500. Para lograr esto, las propiedades del registro holding encargado de manejar la salida PWM (registro 40008 en Lookout) deben cumplir con estas características.

3.2.4 Diagramas de Flujo

Son ocho los diagramas de flujo que se presentan a continuación (figuras, 3.2 a 3.9) y dan una idea general de como el programa implementado en el PIC16F877 está diseñado.

Para mejorar la comprensión de dichos diagramas, se detallan algunas de las variables involucradas en el código del programa con una muy breve descripción de las mismas.

Variables y banderas de la subrutina de interrupción

p(i) : Arreglo donde se almacena la pregunta del Maestro

Temp : Backup de RCREG

Temp1 : Temp desplazado cuatro posiciones

I : Índice del almacenaje de datos en el arreglo p(i)

DP : Bandera de detección del caracter dos puntos (:)

Unir : Bandera que une dos nibbles en un byte

esclavo : Dirección de esclavo

Variables y banderas de propósito general

config y espera : Registros asociados a la configuración de los parámetros de comunicación.

POR o PCON.1 : Bit que permite conocer si ha ocurrido un Power-on Reset

ByC : Byte Count

STimer : Bandera que determina la configuración del Timer1

Entrar: Bandera que permite la configuración de los parámetros de comunicación

MTS : Bandera que establece el Modo de Transmisión Serial

k : Variable de propósito general

num : Número de elementos

pos : Posición de un elemento

La figura 3.2 muestra el diagrama de flujo del programa principal, el cual inicia cargando la configuración de puertos de entrada y salida, interrupciones y la información almacenada en la EEPROM para los registros SPBRG, T1CON, T2CON, PR2, dirección del esclavo, red y modo de transmisión serial. La bandera Entrar complementada con el contador espera y la variable config se encargarán de la configuración de los parámetros de comunicación; dirección del esclavo, velocidad de transmisión, modo de transmisión serial y habilitación o no del driver de red. Luego, la bandera MTS determina si en el análisis de tramas enviadas por el maestro y almacenadas en el arreglo p(i) será utilizando el modo de transmisión RTU o ASCII y finalmente por medio de la subrutina Comprobar evaluar la integridad de cada unos de los datos que conforman dicha trama.

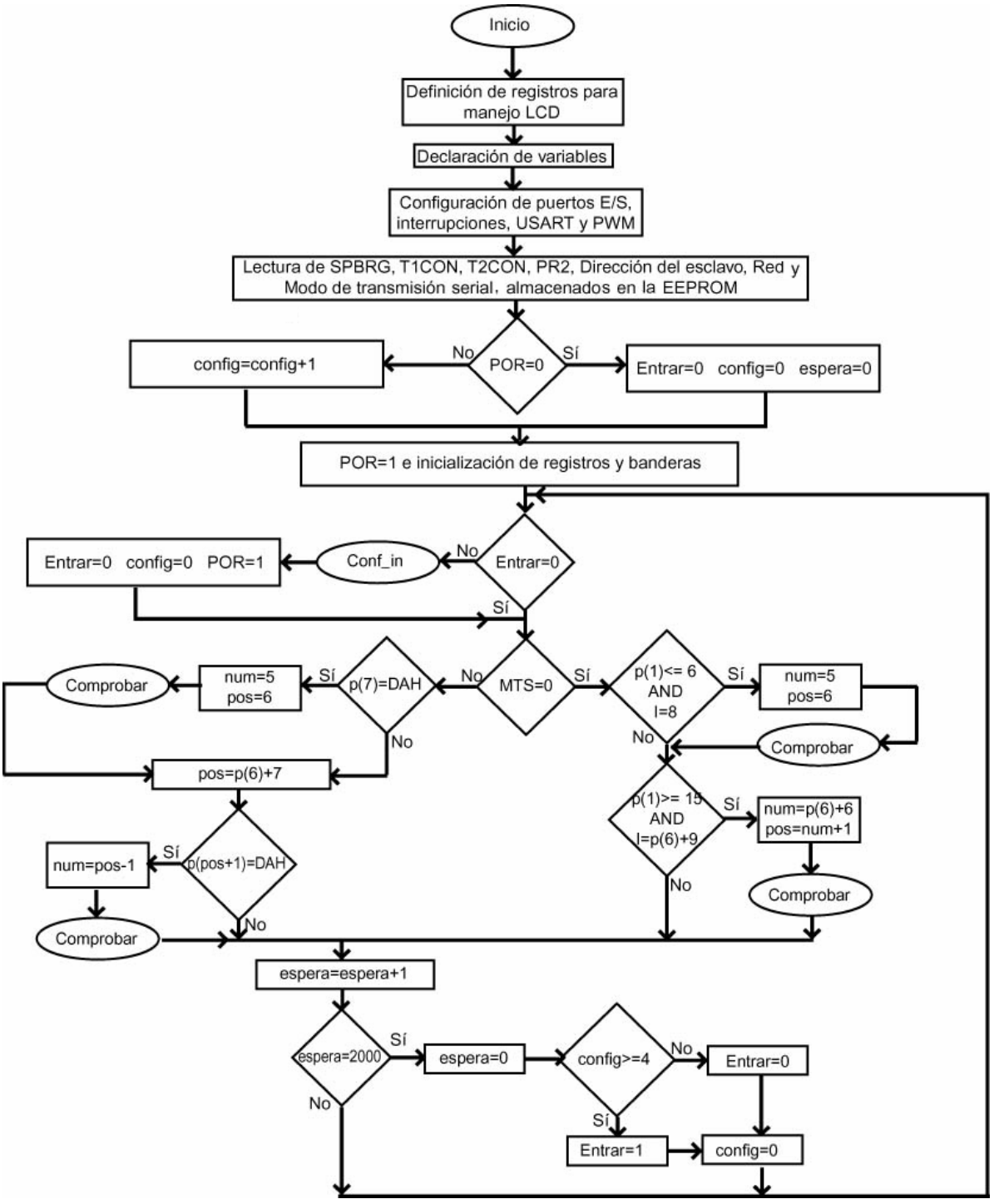

Figura 3.2. Diagrama de Flujo del Programa Principal

En la figura 3.3 se muestra el diagrama de flujo de la subrutina Comprobar, la cual utiliza dos subrutinas; CRCLRC y Ch_CRCLRC, que en conjunto determinan si la trama enviada por un dispositivo maestro es una trama válida.

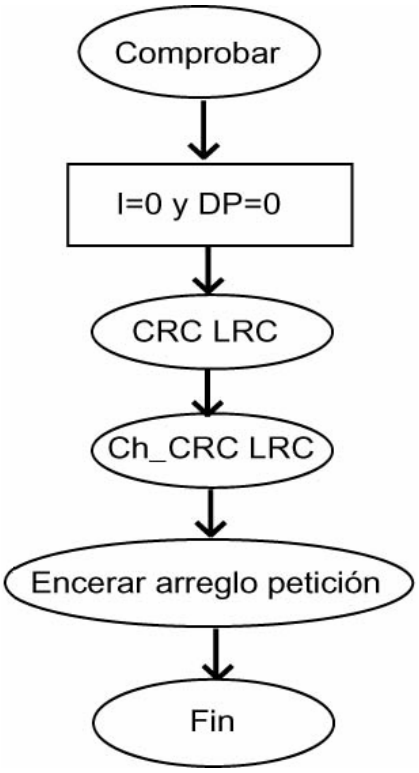

Figura 3.3. Diagrama de Flujo de la Subrutina Comprobar

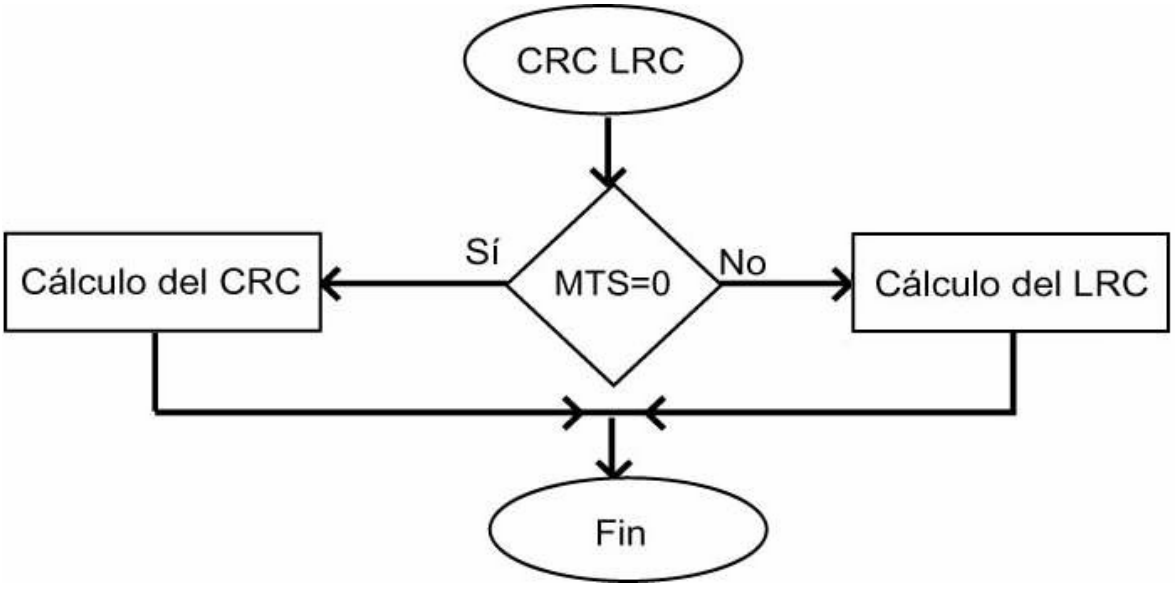

Figura 3.4. Diagrama de Flujo de la Subrutina CRCLRC

En el diagrama de flujo de la subrutina CRCLRC (figura 3.4) se calcula el chequeo de redundancia cíclica o longitudinal según lo indique la bandera MTS

utilizando para ello los datos enviados por un dispositivo maestro almacenados en el arreglo p(i).

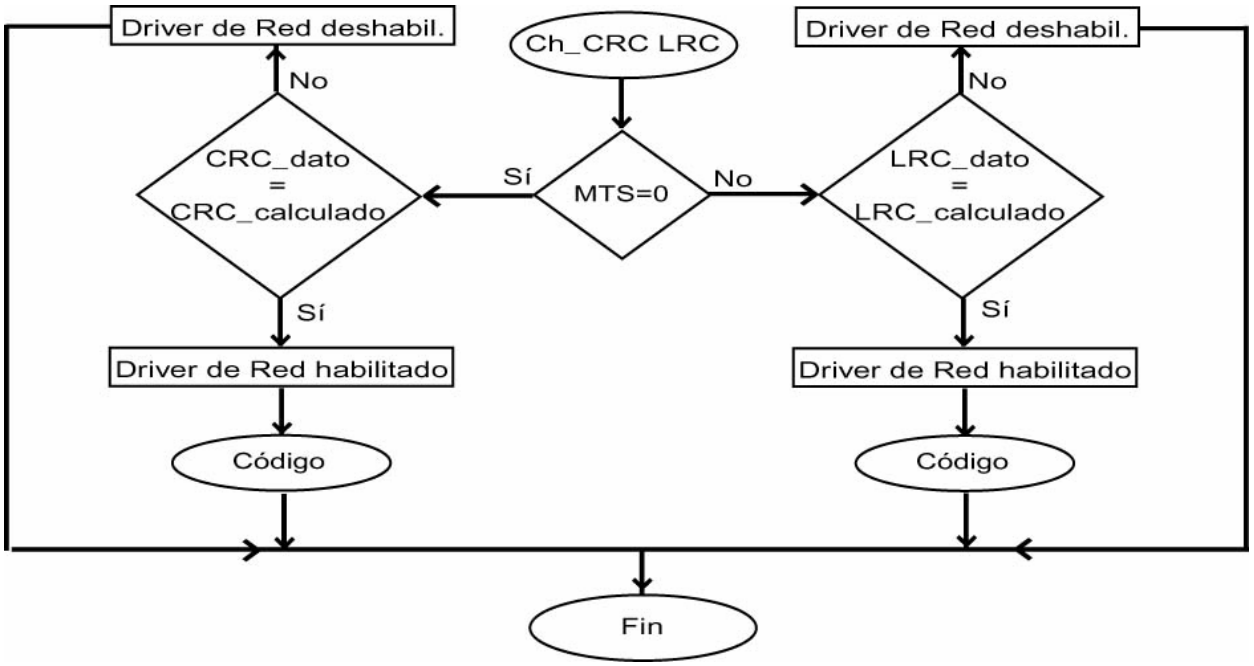

Figura 3.5. Diagrama de Flujo de la Subrutina Ch_CRCLRC

En la figura 3.5 el diagrama de flujo de la subrutina Ch_CRCLRC compara el chequeo de redundancia cíclica o longitudinal calculado en la subrutina CRCLRC y lo compara con el CRC o LRC contenido en el arreglo p(i), dato que envía el dispositivo maestro dentro de su trama de pregunta. Si existe una coincidencia entre los datos mencionados se analiza el campo código de función contenido en el arreglo p(i) por medio de la subrutina Codigo.

En la figura 3.6 el diagrama de flujo de la subrutina Codigo analiza el campo código de función [p(1)] para determinar la operación a realizarse dentro las ocho posibles; lectura de las salidas a relé y digitales [p(1)=1], lectura de entradas digitales [p(1)=2], lectura de registros holding [p(1)=3], lectura de registros de entrada [p(1)=4], escribir una bobina individual [p(1)=5], escribir un registro individual [p(1)=6], escribir múltiples bobinas [p(1)=15] y escribir múltiples registros holding [p(1)=16]. La operación siguiente es generar una trama de

respuesta de que va de acuerdo al código de función gestionado y a uno de los dos modos de transmisión serial ASCII o RTU.

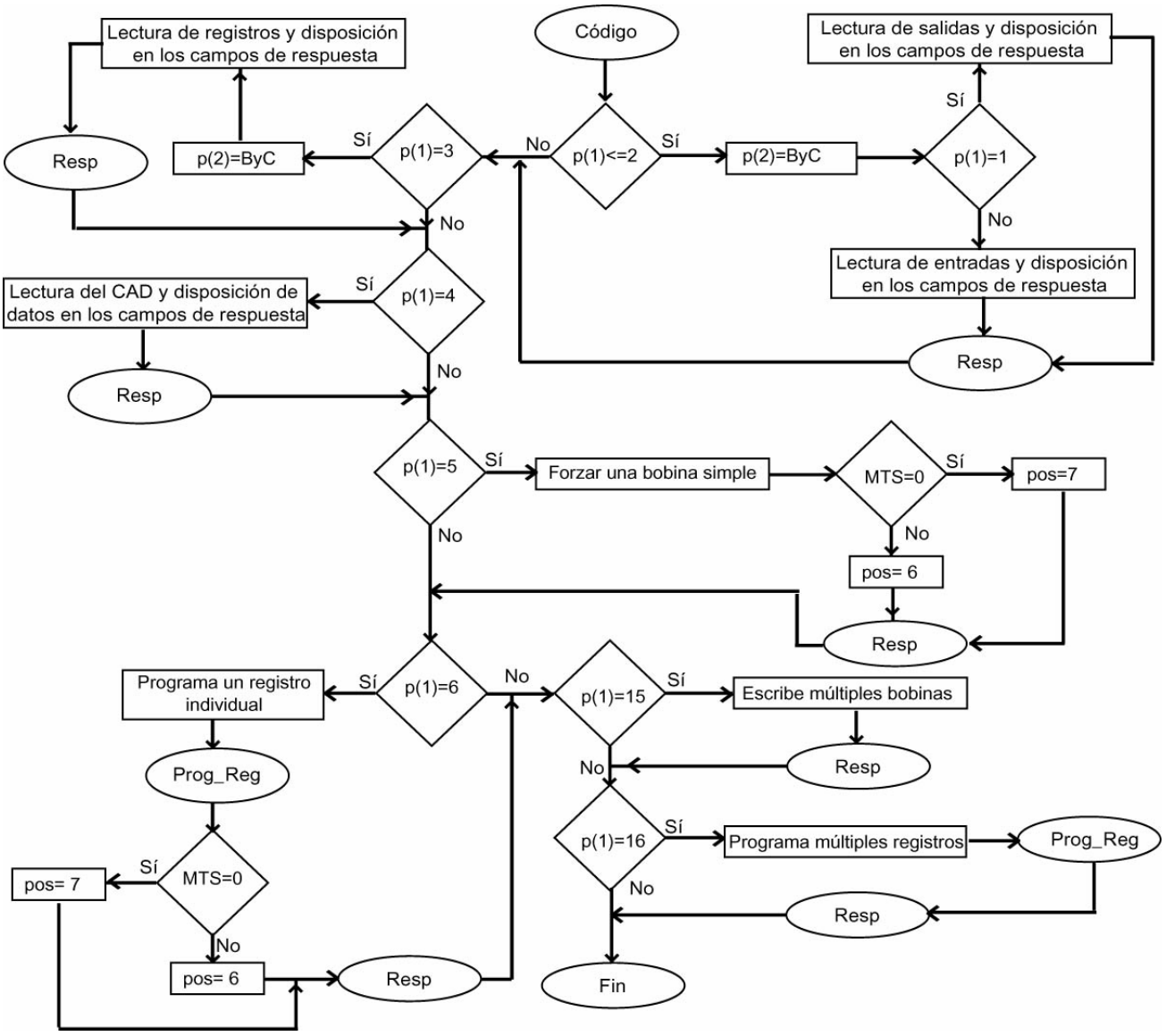

Figura 3.6. Diagrama de Flujo de la Subrutina Codigo

En la figura 3.7 se muestra el diagrama de flujo de la subrutina Prog_reg que sirve para configurar los registros T1CON, TMR1, T2CON y PR2. Además se maneja la activación o no de la señal PWM y el registro de visualización (LCD) Esta configuración y activación de registros internos se realiza mediante la HMI realizada en Lookout.

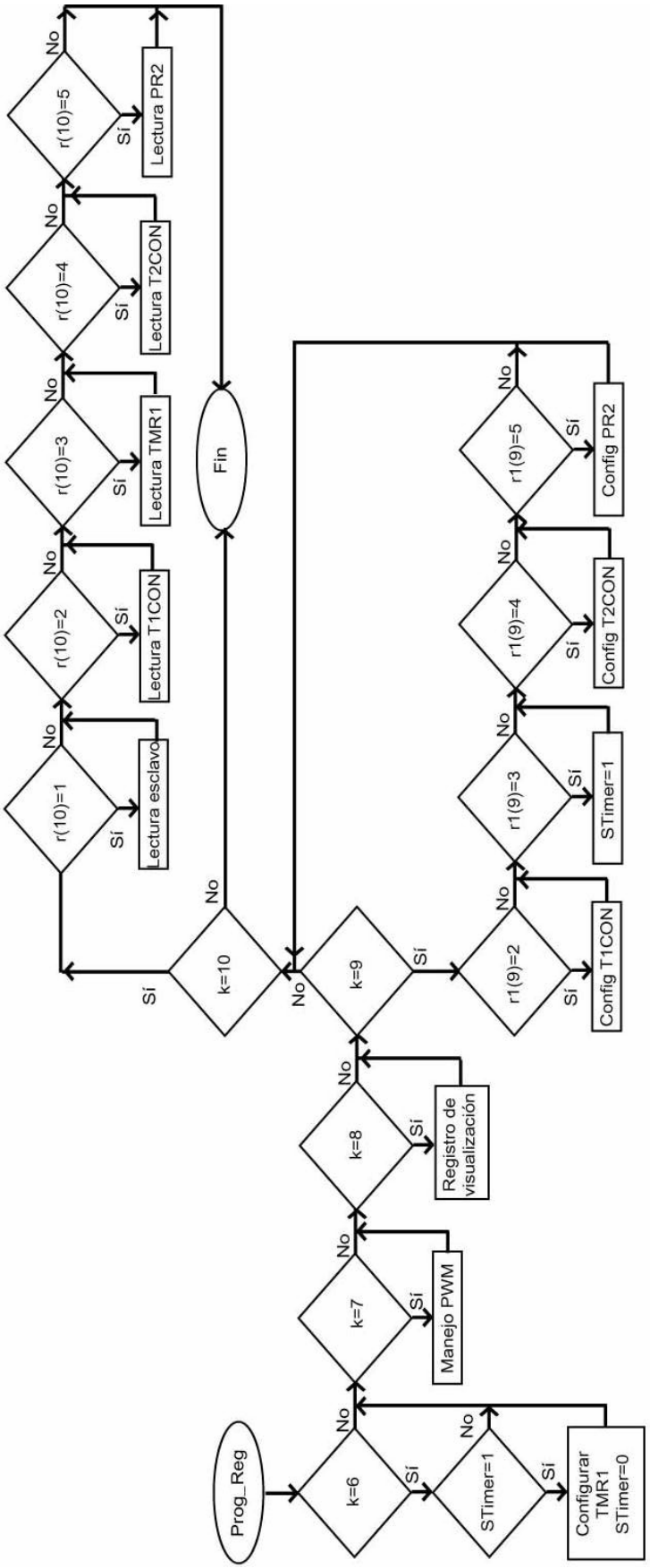

Figura 3.7. Diagrama de Flujo de la Subrutina Prog_Reg

En la figura 3.8 el diagrama de flujo de la subrutina Resp construye el cuerpo del mensaje a una petición enviada por el dispositivo maestro y calcula su chequeo de redundancia cíclica o longitudinal dependiendo de modo de transmisión serial utilizado, el cual se agregará como parte de la respuesta total que emitirá el dispositivo esclavo hacia el maestro.

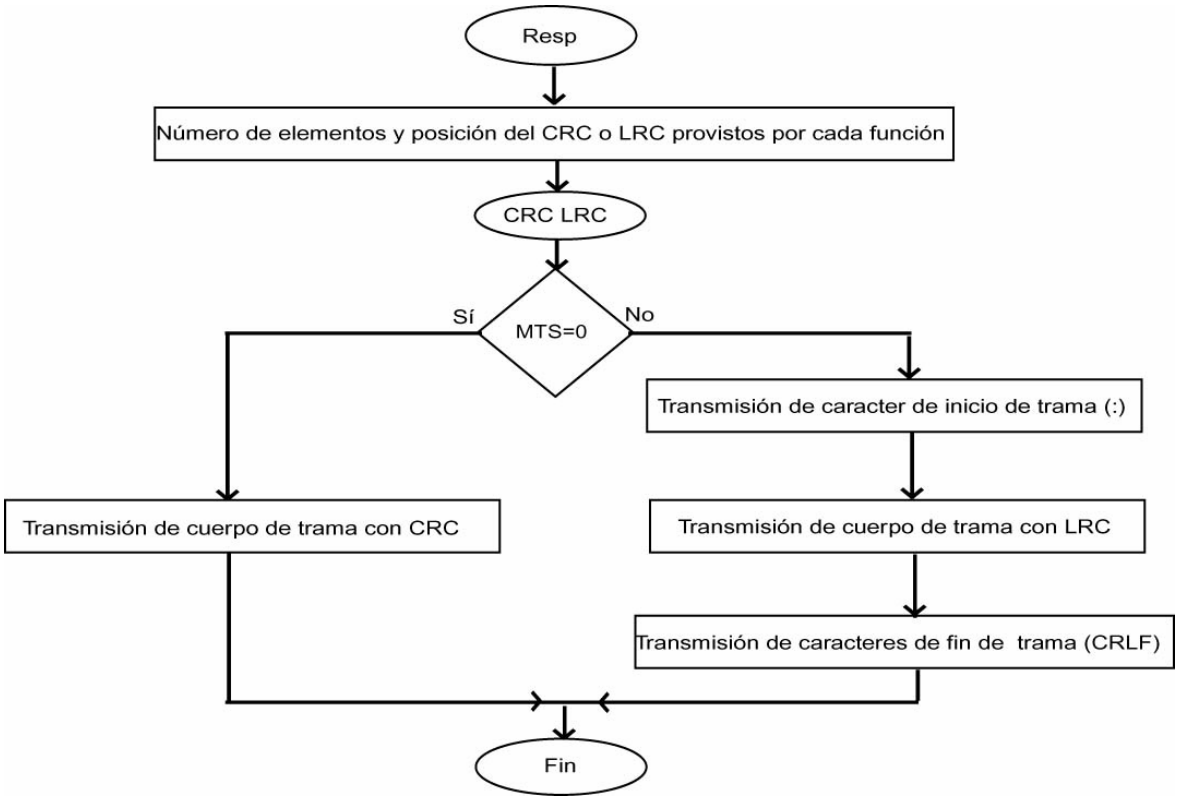

Figura 3.8. Diagrama de Flujo de la Subrutina Resp

En la figura 3.9 el diagrama de flujo de la subrutina de interrupción Recibir recepta los datos enviados serialmente (RTU o ASCII) por el dispositivo maestro y los almacena en el arreglo p(i). En el modo de transmisión ASCII se espera recibir el carácter dos puntos (:) y que el campo correspondiente a la dirección de esclavo sea la misma que lo identifica, antes de empezar a almacenar los datos y los caracteres CRLF para terminar el almacenaje de los mismos. En el modo de transmisión RTU se espera que el campo correspondiente a la dirección de esclavo sea la misma que lo identifica, que los códigos de función sean válidos y que el índice del arreglo p(i) (tamaño de la trama de petición del maestro) vaya de acuerdo al código de función a gestionarse.

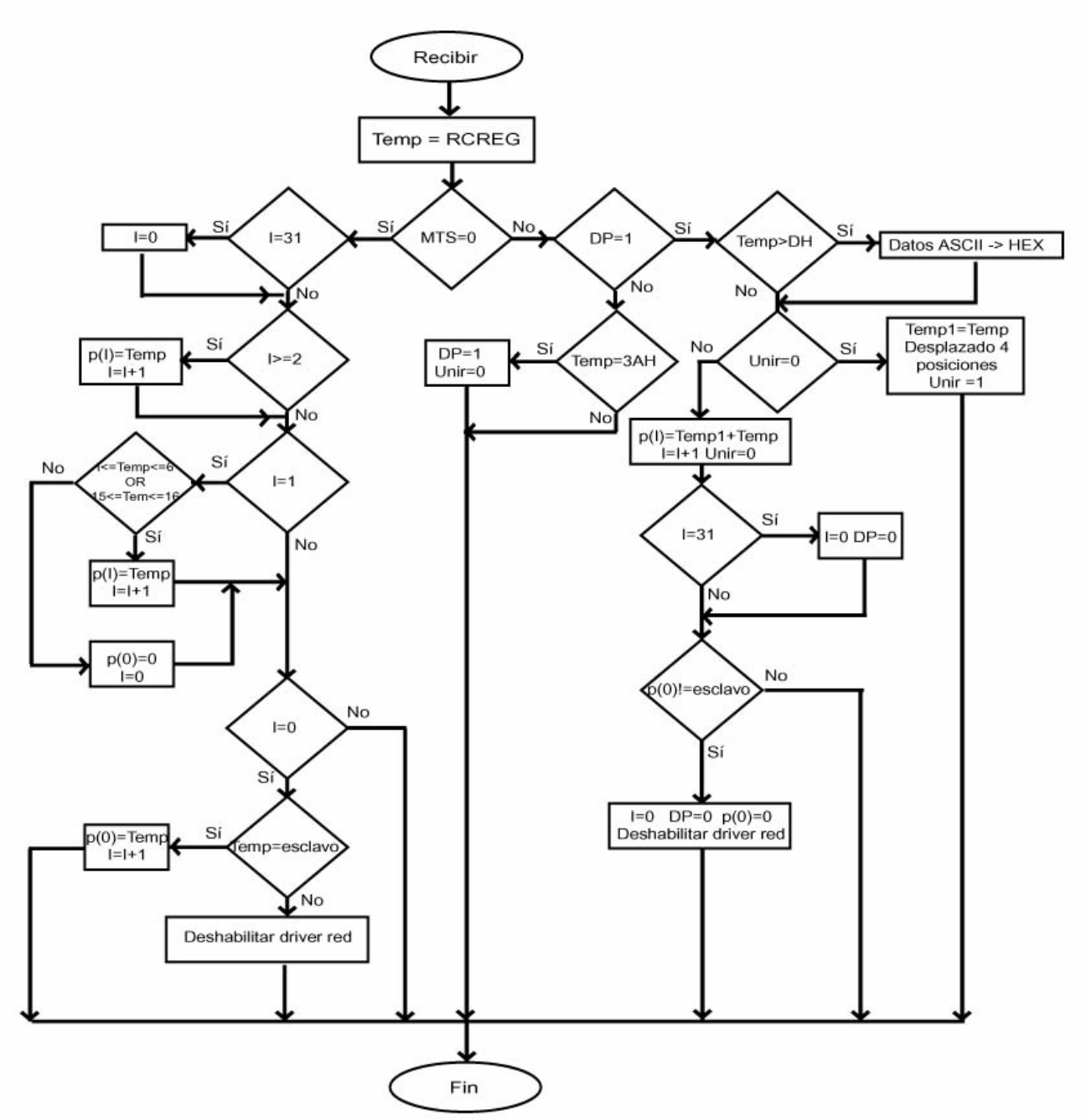

Figura 3.9. Diagrama de Flujo de la Subrutina de Interrupción Recibir

3.2.5 Programa Implementado en el PIC16F877

```
'*************************************************************************************************
'* Nombre:   Modbus.bas
'* Autor:      Rodrigo Cárdenas
'*
'* Tema:      Implementación del protocolo Modbus utilizando el microcontrolador
'*              PIC16F877
'* Versión:   1.0
'*************************************************************************************************

'*************************************************************************************************'
‘    DEFINICIÓN DE REGISTROS PARA MANEJO DE LCD
'*************************************************************************************************
DEFINE LCD_DREG PORTB       ' Puerto de datos del LCD
DEFINE LCD_DBIT 0               ' Puerto LCD de 4 bits comenzando en B0
DEFINE LCD_RSREG PORTC     ' Puerto de selección de registro LCD
DEFINE LCD_RSBIT 3              ' Bit de selección de registro LCD (RS)

DEFINE LCD_RWREG PORTC    ' Puerto de Lectura/escritura del LCD
DEFINE LCD_RWBIT 4              ' Bit de lectura / escritura LCD (R/W)
DEFINE LCD_EREG PORTC       ' Puerto de habilitación LCD

DEFINE LCD_EBIT 5      ' Bit de habilitación LCD (E)
DEFINE LCD_BITS 4      ' Bus LCD de 4 bits
DEFINE LCD_LINES 2     ' Número de lineas del LCD

DEFINE CHAR_PACING 100  ' Paso de la salida serial en us

'*************************************************************************************************
'    DATOS ALMACENADOS EN LA EEPROM AL MOMENTO DE GRABAR EL PIC
'*************************************************************************************************
```

‘Tanto los valores de dirección del esclavo, modo de transmisión serial, velocidad de ‘transmisión, estado de la red, registros asociados a los Timers y PWM, y mensajes ‘de despliegue para el LCD, son almacenados en la EEPROM de datos. La

‘instrucción "Data" se ejecuta solamente al grabar el PIC, dando valores iniciales de ‘trabajo al dispositivo esclavo

```
'*****************************************************************************************************
‘    DATOS DE INICIALES DE REGISTROS Y BANDERAS
'*****************************************************************************************************
Dato_Esclavo Data 1  ' Dirección del dispositivo esclavo 01H
Dato_MTS Data 0  ' Modo de Transmisión Serial ->MTS = 0-> RTU  MTS = 1 -> ASCII
Dato_SPBRG Data 25  ' Velocidad de Transmisión = 9600 Baudios -> SPBRG = 25
Dato_Hab Data 2  ' Driver deshabilitado
Dato_T1CON Data 6  ' Timer1 apagado
Dato_T2CON Data 2  ' Timer2 apagado
Dato_PR2 Data 124  ' Registro PR2 = 124

'*****************************************************************************************************
‘    MENSAJES PARA LECTURA DE REGISTROS Y DESPLIEGUE DE CARÁTULA
'*****************************************************************************************************
Mensj_T1CON Data " REGISTRO T1CON ",0
Mensj_TMR1HL Data "   TMR1H :TMR1L  ",0
Mensj_T2CON Data " REGISTRO T2CON ",0
Mensj_PR2 Data "  REGISTRO PR2  ",0
Mensj_Tesis Data " TESIS DE GRADO ",0
Mensj_Implementacion Data " IMPLEMENTACION ",0
Mensj_Protocolo Data "  PROTOCOLO MB  ",0
Mensj_Integrantes Data "  INTEGRANTES   ",0
Mensj_Mi_nombre Data "Rodrigo Cardenas",0
Mensj_Esclavo Data " ESCLAVO ",0
'*****************************************************************************************************
'        DECLARACIÓN DE VARIABLES
'*****************************************************************************************************
'*****************************************************************************************************
'    VARIABLES Y BANDERAS DE INTERRUPCIÓN DEL USART (no reutilizables)
'*****************************************************************************************************
peticion var byte [31]  ' Arreglo donde se almacena la petición del Maestro
Temp1 var byte BANK1  ' Backup de RCREG<<4
```

```
Indice var byte BANK1  ' Índice del almacenaje de datos en peticion
Bandera var byte BANK1  ' Registro en que cada bit es usado como una bandera
DP var Bandera.0  ' Bandera de detección del caracter dos puntos (:)
Unir var Bandera.1  ' Bandera que une dos nibbles en un byte

'*****************************************************************************************************
'    VARIABLES Y BANDERAS DE PROPÓSITO GENERAL (no reutilizables)
'*****************************************************************************************************
est var bit [10]  ' Estado de las bobinas
est1 var bit [6]  ' Estado de las entradas digitales
Ch var byte [4]  ' Canales del CAD (ADCON0)
registro var byte [11] : registro1 var byte [11]  ' Para manejo de registros internos
config var byte BANK1 : espera var word BANK1 ' Registros asociados a la
‘configuración de parámetros de comunicación
esclavo var byte BANK1  ' Dirección de esclavo
Hab var byte BANK1  ' Estado de la Red
LSB var Bandera.2  ' Bandera que examina el LSB de "pg" para el cálculo del CRC
STimer var Bandera.3  ' Bandera que determina la configuración del Timer1
Nop var Bandera.4 'Bandera No operación y despliegue de mensaje "Esclavo XX OK"
Entrar var Bandera.5  ' Bandera que permite la configuración de comunicaciones
MTS var Bandera.6  ' Bandera que establece el Modo de Transmisión Serial
Driver var PORTD.5  ' Nombre dado a la salida que servirá de Driver de red (S10D)
POR var PCON.1  ' Nombre del bit que permite conocer si ha ocurrido un Power-on
‘ Reset

'*****************************************************************************************************
'    VARIABLES DE PROPÓSITO GENERAL (reutilizables)
'*****************************************************************************************************
k var byte BANK1 : m var byte BANK1 : pg var word BANK1 : pgw var word BANK1
pg1 var byte BANK1 : pg2 var byte BANK1 : pg3 var byte BANK1
num var byte BANK1  ' Número de elementos
pos var byte BANK1  ' Posición de un elemento
direcc var byte BANK1  ' Dirección de un elemento
ind var byte BANK1  ' Índice de arreglo
        GoTo Inicio  ' Salta alrededor del handler de interrupciones
```

```
'*************************************************************************************************
'  INTERRUPCIÓN DEL USART
'*************************************************************************************************
'En el Modo de Transmisión Serial RTU, los mensajes comienzan y finalizan con un
‘intervalo silencioso de al menos 3.5 tiempos de caracter, esto hace que el código
‘implementado para este modo deba ser riguroso, evitando así que el dispositivo
‘esclavo reaccione a peticiones que no correspondan a la dirección que lo identifica.
'Son dos las coincidencias que continuamente se buscan, que la dirección del
‘dispositivo esclavo (Indice=0) sea igual a la configurada por el usuario y que el
‘campo que le sigue a continuación (Indice=1) sea un código de función válido. Si
‘ambas condiciones se cumplen, en el arreglo peticion se guarda directamente la
‘pregunta del Maestro (Indice>=2), ya que cada byte de 8 bits en un mensaje tiene
‘dos caracteres  hexadecimales de 4 bits.

'En el Modo de Transmisión Serial ASCII todos los caracteres entrantes son
‘comparados con el caracter dos puntos (:) cuyo valor en hexadecimal es 3AH. Si se
‘encuentra dicho valor esto significa que se ha hallado el inicio de la Trama Modbus,
‘seteando la bandera DP, procediendo luego al almacenaje de los datos en el arreglo
‘peticion. Los caracteres entrantes de 8 bits son convertidos a su equivalente
‘hexadecimal de 4 bits, de esta forma se guarda dos caracteres en un solo byte,
‘optimizando su almacenamiento. Para realizarlo se utiliza una bandera llamada Unir.
‘Cuando es 0 desplaza el dato entrante cuatro posiciones a la izquierda y cuando es 1
‘suma el valor actual con el anteriormente desplazado.
'"Temp>$D" sirve para guardar los caracteres de fin de trama (CR y LF) sin
‘conversión alguna porque sus valores son DH y AH respectivamente.

'Para los dos Modos de Transmisión Serial, peticion[0]=esclavo es el único valor de
‘dirección que el dispositivo esclavo asumirá como válido.

'Si la bandera Hab es 1 o 0, quiere decir que la red está habilitada y que el estado de
‘habilitación del Driver de Red corresponde al flanco elegido por el usuario, ya sea
‘este flanco ascendente o flanco descendente.

'El número máximo del arreglo peticion es 31. Se eligió esté considerando el tamaño
‘de la trama que envía el Maestro de acuerdo a los 8 códigos de función
```

‘implementados. Se estableció que la trama más grande es la de la función 10H la ‘cual maneja la programación de múltiples registros, el código de función que le sigue ‘en tamaño es el 0FH. Como el programa posee 11 registros y estos podrán ser ‘manejados por el Maestro al mismo tiempo, se realiza el siguiente análisis:

'Los primeros 7 campos en la función 10H son constantes e independientes del ‘número de registros, siendo los restantes el número de registros multiplicados por 2 ‘(ya que cada uno de ellos tiene dos campos de 8 bits) a lo que luego se suma 1 ‘campo correspondiente al LRC o dos campos si se tratará del CRC, y otro para los ‘caracteres de fin de trama (CR y LF) para el Modo de Transmisión Serial ASCII.

'A continuación se nuestra la asignación del tamaño del arreglo petición de acuerdo al 'número de registro que manejará el dispositivo Esclavo.

```
' Modo ASCII    7 + (2*11) + 1 (LRC) + 1 (CRLF) = 31
' Modo RTU      7 + (2*11) + 2 (CRC) = 31
```

'Se puede observar que 31 serán los elementos constitutivos del arreglo petición, si ‘se necesitara implementar dicho código en otro PIC, es estrictamente necesario ‘tener en cuenta el tamaño máximo de trama entre los códigos de función que se ‘deseen implementar en este supuesto programa.

```
Recibir: Temp = RCREG
        If MTS = 0 Then ' Modo de Transmisión Serial RTU
          If Indice = 31 Then
            Indice = 0
          End If
          If Indice >= 2 Then
            peticion [Indice] = Temp : Indice = Indice + 1
          End If
          If Indice = 1 Then
            If (Temp>=1 And Temp<=6) Or (Temp>=$F And Temp<=$10) Then
              peticion [1] = Temp : Indice = Indice + 1
            Else
              peticion [0] = 0 : Indice = 0
```

```
      End If
    End If
    If Indice = 0 Then
      If Temp = esclavo Then
        peticion [0] = Temp : Indice = Indice + 1
      Else
        If Hab < 2 Then
          Driver = Hab ^ 1 ' Deshabilito driver
        End If
      End If
    End If
  Else  ' MTS = 1
    If DP = 1 Then ' Modo de Transmisión Serial ASCII
      ' Detección de CRLF
      If Temp>$D Then
        ' Conversión de ASCII a DEC
        Lookdown RCREG, ["0123456789ABCDEF"], Temp
      End If
      If Unir = 0 Then
        Temp1=Temp<<4 : Unir=1
      Else
        peticion [Indice] = Temp1 + Temp : Indice = Indice + 1: Unir = 0
        If Indice = 31 Then
          Indice = 0 : DP = 0
        End If
        If peticion[0]!=esclavo Then
          DP = 0 : Indice = 0 : peticion [0] = 0
          If Hab < 2 Then
            Driver = Hab ^ 1 ' Deshabilito driver
          End If
        End If
      End If
    Else
      If Temp=$3A Then
        DP = 1 : Unir = 0
```

```
            End If
          End If
        End If

        PIR1 0.5 = 0
        Resume
        Enable

On INTERRUPT GoTo Recibir

'*****************************************************************************************************
'    PUERTOS DE I/O Y REGISTRO OPTION
'*****************************************************************************************************
Inicio:  ADCON1=%00000110  ' Puerto A como I/O digitales
        TRISA=%00100000  ' Out = 0 / In = 1
        TRISB=%11110000
        TRISC=%10000011
        TRISD=%11000000
        TRISE=%00000111
        OPTION_REG=%10000011  ' Pull-up desactivado

'*****************************************************************************************************
'    REGISTROS DE INTERRUPCIONES
'*****************************************************************************************************
        INTCON=%11000000  ' Otros periféricos SI!!!!
        PIE1=%00100000  ' Permiso de interrupción del USART
        PIR1 0.5 = 0 ' Borra señalizador de RX  RCIF

'*****************************************************************************************************
'    INICIALIZACIÓN DE LCD
'*****************************************************************************************************
        Lcdout $FE,$6  ' Establece modo de funcionamiento
        Lcdout $FE,$C  ' Control ON / OFF
        Lcdout $FE,1 : direcc=Mensj_Esclavo : Gosub LCD
        Read Dato_Esclavo, esclavo ' Lee la dirección asignada a este dispositivo
                                   ' esclavo
```

```
        Lcdout,$FE,$89,dec esclavo," OK   "

'*****************************************************************************************************
'     INICIALIZACIÓN DE REGISTROS PARA USART
'*****************************************************************************************************
        TXSTA=%00100100  ' 8 bits, Asincrónico, High speed
        RCSTA=%10010000  ' Habilita puerto serie y rx continua
        Read Dato_SPBRG, SPBRG ' Este registro nos permitirá escoger las diferentes
                                ' velocidades de transmisión
'*****************************************************************************************************
'     INICIALIZACIÓN DE REGISTROS PARA MANEJO DE LA SEÑAL PWM
'*****************************************************************************************************
        ' Fosc = Frecuencia del reloj (4.0MHz)
        ' PD = Predivisor del Timer2 (T2CON<1:0>)
        ' Fpwm = Frecuencia PWM
        ' CT% = Ciclo de trabajo (20% = 0.2)

        ' Fórmulas:
        ' PR2=(Fosc/(4*PD*Fpwm))-1
        ' CCPR1L:CCP1CON<5:4>=(PR2+1)*4*Duty%

        ' Fpwm = 500Hz, Predividor del TMR2 = 1:16    PR2=124
        ' CCPR1L:CCP1CON<5:4>=(PR2+1)*4*Duty%  = 500

        CCP1CON=%00001100  ' Modo PWM habilitado
        CCP1CON 0.4 = 0 : CCP1CON 0.5 = 0 : CCPR1L = 0
        Read Dato_PR2, PR2

'*****************************************************************************************************
'     INICIALIZACIÓN DE LA CONFIGURACIÓN DE TIMERS
'*****************************************************************************************************
'Lee valores asignados a T1CON y T2CON que fueron almacenados en la EEPROM
‘de datos.
        Read Dato_T1CON, T1CON : Read Dato_T2CON, T2CON
```

```
'*************************************************************************************************
'    DRIVER DE RED
'*************************************************************************************************
'Lee en Dato_Hab el estado de la Red que fue almacenada en la EEPROM de datos.

      Read Dato_Hab, Hab
      If Hab = 2 Then
        Driver = 0 ' PORTD.5=0
      Else
        Driver = Hab ^ 1
      End If

'*************************************************************************************************
'    MODO DE TRANSMISIÓN SERIAL
'*************************************************************************************************
'Lee en Dato_MTS el estado del Modo de Transmisión Serial almacenado en la
'EEPROM de datos.

      Read Dato_MTS, MTS ' MTS = 0 -> RTU   MTS = 1 -> ASCII

'*************************************************************************************************'*************
'    ESTABLECE SI HA OCURRIDO UN POWER-ON RESET (POR)
'*************************************************************************************************
      If POR = 0 Then ' Si ha ocurrido un Power-on Reset
        Entrar = 0 : config = 0 : espera = 0
      Else  ' POR=1 Si ha ocurrido un MCLR Reset durante una operación normal
        config = config + 1
      End If
      POR = 1

'*************************************************************************************************
'    INICIALIZACIÓN DE REGISTROS Y BANDERAS
'*************************************************************************************************
      PORTA.4 = 0 : STimer = 0 : DP = 0 : Indice = 0
      PORTA.0 = 0 : PORTA.1 = 0 : PORTA.2 = 0 : PORTA.3 = 0 : PORTD.0 = 0
```

```
PORTD.1 = 0 : PORTD.2 = 0 : PORTD.3 = 0 : PORTD.4 = 0 : GoSub Ence_P
For k = 0 To 9
   est [k] = 0
   If k <= 5 Then
      est1 [k] = 0
   End If
Next k

'*****************************************************************************************************
'        PROGRAMA PRINCIPAL
'*****************************************************************************************************
'El Modo de Transmisión Serial RTU no posee un caracter de inicio de trama o
‘caracteres de fin de trama, por lo tanto se necesita saber en que momento se debe
‘analizar el contenido de todos sus campos para el cálculo del CRC, debido a ello se
‘utiliza la variable Indice con ayuda del elemento petición[1] que corresponde al
‘código de función enviado por el Maestro en su trama de pregunta. En las variables
‘num y pos se cargan los valores, número de elementos para el cálculo del CRC y la
posición del CRC en la trama almacenada en el arreglo peticion.

'En el Modo de Transmisión Serial ASCII la posición en la cual los caracteres de fin
‘de trama (CR y LF) se almacenaron en el arreglo peticion determina el mejor
‘momento para el cálculo del LRC. En las variables num y pos se cargan los valores,
‘número de elementos para el cálculo del LRC y la posición del LRC en la trama
‘almacenada en el arreglo peticion.

'Tanto el cálculo del CRC como del LRC en la trama de petición del maestro sirve
‘para verificar que la integridad de la información contenida en sus campos es
‘correcta, comparándola con el dato CRC o LRC que envía el maestro dentro de su
‘trama. Los valores propuestos en el programa del número de elementos para el
‘cálculo del CRC o LRC y la posición de los mismos en la trama almacenada se
‘obtuvieron por simple inspección, de acuerdo a las similitudes y diferencias entre los
‘diferentes códigos de función.
'El campo correspondiente a la dirección del esclavo está almacenada siempre en la
'posición 0 del arreglo peticion.
```

'espera=2000 establece un tiempo corto para que el usuario presionando cuatro o ‘más veces el botón Reset (config>=4) pueda ingresar a la configuración interna de ‘los parámetros de comunicación (Entrar=1), subrutina llamada Conf_in.

```
Esperar:  If Entrar = 1 Then
            GoSub Conf_in : Entrar = 0 : config = 0 : POR = 1 : Pause 300
          End If
          If MTS = 0 Then
            If peticion[1]<=6 And Indice=8 Then
               num = 5 : pos = 6 : GoSub Comprobar
            End If
            If peticion[1]>=$F And Indice=(peticion[6]+9) Then
              num=peticion[6]+6 : pos=num+1 : Gosub Comprobar
            End If
          Else
            If peticion[7]=$DA Then
              num = 5 : pos = 6 : GoSub Comprobar
            End If
            pos=peticion[6]+7
            If peticion[pos+1]=$DA Then
              num = pos – 1 : GoSub Comprobar
            End If
          End If
              espera = espera + 1
          If espera = 2000 Then
            espera = 0
            If config >= 4 Then
              Entrar = 1
            Else
              Entrar = 0
            End If
           config = 0
          End If

        GoTo Esperar
```

```
'*****************************************************************************************************
'       SUBRUTINAS
'*****************************************************************************************************
'*****************************************************************************************************
'    SUBRUTINA QUE IMPRIME EN EL LCD UN MENSAJE ALMACENADO EN
'    EEPROM
'*****************************************************************************************************
'En direcc se carga la dirección de inicio del mensaje, imprimiéndose en el LCD
'caracter a caracter de acuerdo a como direcc se va incrementando. El fin del
'mensaje sucede cuando un cero es encontrado, llevado a cabo esto la subrutina
'finaliza.

LCD:    Read direcc, pg1
        If pg1 = 0 Then Sigue
        Lcdout pg1 : direcc = direcc + 1 : GoTo LCD
Sigue:  Return

'*****************************************************************************************************
'    SUBRUTINA QUE ENCERA EL ARREGLO PETICION
'*****************************************************************************************************
Ence_P:  For k = 0 To 30
            peticion [k] = 0
         Next k
         Return

'*****************************************************************************************************
'    SUBRUTINA QUE CONFIGURA PARÁMETROS DE COMUNICACIÓN
'*****************************************************************************************************
'Presionando cuatro veces o más el botón de Reset en un intervalo pequeño de
'tiempo se logra acceder a la configuración de los parámetros de comunicación.
'Las entradas digitales 10001 y 10002 sirven para cambiar los valores del dato a
'configurarse, mientras que con la entrada discreta 10003 se cambia de parámetro.

'Los parámetros que pueden ser configurados son:
'Dirección del Esclavo: 1 a 247
```

```
'Modo de Transmisión Serial: RTU o ASCII
'Velocidad de Transmisión: 1200, 2400, 4800, 9600 o 19200 Baudios
'Red: Deshabilitada, Habilitada en bajo o Habilitada en alto

Conf_in: RCSTA.7=0 : Lcdout $FE,1 : Read Dato_Esclavo,pg1
        Lcdout $FE,$80,"  DIREC ESCLAVO "
Escl:    Lcdout $FE,$C6,dec pg1," Dec   "
        If PORTB.4=1 Then  ' Bornera etiquetada como 10001
          Pause 110
          If pg1 = 247 Then
            pg1 = 0
          End If
          pg1 = pg1 + 1 : Pause 110
        End If
        If PORTB.5=1 Then  ' Bornera etiquetada como 10002
          Pause 110
          If pg1 = 1 Then
            pg1 = 248
          End If
          pg1 = pg1 – 1 : Pause 110
        End If
        If PORTB.6=1 Then  ' Bornera etiquetada como 10003
          Pause 450 : Write Dato_Esclavo,pg1 : Goto Modts
        End If
        GoTo Escl

Modts:   Read Dato_MTS, pg1
        Lcdout $FE,$80,"MODO TRANSMISION" : Lcdout $FE,$C0," Serial  "
Modts1:  If pg1 = 0 Then
          Lcdout $FE,$C9,"RTU    "
        End If
        If pg1 = 1 Then
          Lcdout $FE,$C9,"ASCII  "
        End If
Modts2:  If PORTB.4=1 Then  ' Bornera etiquetada como 10001
```

```
        Pause 200 : pg1 = pg1 ^ 1 : Pause 200 : GoTo Modts1
      End If
      If PORTB.6=1 Then  ' Bornera etiquetada como 10003
        Pause 450  : Write Dato_MTS,pg1 : Goto Velo
      End If
      GoTo Modts2

Velo:   Read Dato_SPBRG,pg1 : Lcdout $FE,$80," VELOCIDAD TRAN "
      Lookdown pg1, [207,103,51,25,12], pg2
Velo1:  Lcdout $FE,$C0,"  ",dec (1200*Dcd(pg2))," Baudios  "
      If PORTB.4=1 Then  ' Bornera etiquetada como 10001
        Pause 200 : pg2 = pg2 + 1
        If pg2 = 5 Then
          pg2 = 0
        End If
        Pause 200 : End If
      If PORTB.5=1 Then  ' Bornera etiquetada como 10002
        Pause 200
        If pg2 = 0 Then
          pg2 = 5
        End If
        pg2 = pg2 – 1 : Pause 200
      End If
      If PORTB.6=1 Then  ' Bornera etiquetada como 10003
        Pause 450 : Lookup pg2,[207,103,51,25,12],SPBRG : Write
Dato_SPBRG,SPBRG
        GoTo Netw
      End If
      GoTo Velo1

Netw:   Read Dato_Hab, pg1
      Lcdout $FE,$80,"     RED      " : Lcdout $FE,$C0," Driver  "
Netw1:  If pg1 = 2 Then
        Lcdout $FE,$C9,"Deshab "
      End If
```

```
        If pg1 = 0 Then
          Lcdout $FE,$C9,"Hab Lo "
        End If
        If pg1 = 1 Then
          Lcdout $FE,$C9,"Hab Hi "
        End If
Netw2:   If PORTB.4=1 Then  ' Bornera etiquetada como 10001
          Pause 200 : pg1 = pg1 + 1
          If pg1 = 3 Then
            pg1 = 0
          End If
          Pause 200 : GoTo Netw1
        End If
        If PORTB.5=1 Then  ' Bornera etiquetada como 10002
          Pause 200
          If pg1 = 0 Then
            pg1 = 3
          End If
          pg1 = pg1 – 1 : Pause 200 : GoTo Netw1
        End If
        If PORTB.6=1 Then  ' Bornera etiquetada como 10003
          Pause 450 : Write Dato_Hab,pg1 : Goto Fin
        End If
        GoTo Netw2

Fin:    Lcdout $FE,1 : direcc=Mensj_Esclavo : Gosub LCD
        Lcdout,$FE,$89,dec esclavo," OK   " : RCSTA.7=1

        Return

'*****************************************************************************************
'    SUBRUTINA QUE REALIZA UNA SERIE DE COMPROBACIONES
'*****************************************************************************************'
```

’Esta subrutina calcula el CRC o LRC de la trama entrante y lo compara con el

```
‘campo donde el maestro coloca su CRC o LRC. La posición de este campo varía con
‘cada código de función, y se carga anticipadamente en el barrido principal.
'Si tal comprobación es exitosa se procede a determinar que código de función debe
‘ejecutarse.

Comprobar: Indice = 0 : DP = 0 : GoSub CRCLRC : GoSub Ch_CRCLRC
                GoSub Ence_P
        Return

'*****************************************************************************************
'    SUBRUTINA QUE CALCULA EL CHEQUEO DE ERROR CÍCLICO O
‘    LONGITUDINAL
'*****************************************************************************************'
’Esta subrutina calcula el chequeo de error cíclico o longitudinal, de acuerdo al Modo
‘de Transmisión Serial elegido por el usuario, siendo solo necesario el número de
‘elementos que estarán involucrados en tal cálculo (num).

CRCLRC:  If MTS = 0 Then
          pg=$FFFF
          For k = 0 To num
             pg=pg^peticion[k]
             For m = 0 To 7
                LSB=pg&1 : pg=pg>>1
                If LSB = 1 Then
                  pg=pg^$A001
                End If
             Next m
          Next k
        Else
          pg1 = 0
          For k = 0 To num
             pg1=pg1+peticion[k]
          Next k
          pg2 = 256 - pg1
        End If : Return
```

```
'*************************************************************************************************
'SUBRUTINA QUE REVISA EL CHEQUEO DE ERROR CÍCLICO O LONGITUDINAL
'*************************************************************************************************
'Establece una comprobación exitosa o no del CRC o LRC de la trama entrante ‘calculado en el programa, con el CRC o LRC enviado por el maestro.

'Si el resultado es verdadero se procede a analizar la clase de función que deberá ‘realizar el dispositivo esclavo (subrutina Codigo), al mismo tiempo que determina si ‘se habilita o no el Driver de Red.

Ch_CRCLRC: If MTS = 0 Then
          pgw=(peticion[pos+1]<<8)+peticion[pos]
          If pg = pgw Then
            If Hab < 2 Then
              Driver = Hab ' Se habilita el driver según su estado de
            End If          ' de activación (ascendente o descendente)
            GoSub Codigo
          Else
            If Hab < 2 Then
              Driver = Hab ^ 1
            End If
          End If
        Else
          If pg2=peticion[pos] Then
            If Hab < 2 Then
              Driver = Hab ' Se habilita el driver según su estado de
            End If          ' de activación (ascendente o descendente)
            GoSub Codigo
          Else
            If Hab < 2 Then
              Driver = Hab ^ 1
            End If
          End If
        End If
        Return
```

Codigo:

```
'*************************************************************************************************
 'SUBRUTINA QUE BASADA EN UNA PETICION GENERA ACCIÓN Y RESPUESTA
'*************************************************************************************************
'Una vez reconocida la dirección del esclavo (peticion[0]=esclavo), el próximo campo
‘a analizar es el de código de función (peticion[1]).

'Cada código de función posee un etiqueta que va a la par con su valor

'CÓDIGOS DE FUNCIÓN              ETIQUETA

'Lectura de bobinas y             UnoDos
'Lectura del estado de entrada
'Lectura de registros holding      Tres
'Lectura de un registro de entrada  Cuatro
'Forzar una bobina simple          Cinco
'Programar un registro individual   Seis
'Forzar múltiples bobinas          Quince
'Programar múltiples registros     Dyseis

'El valor de peticion[1] se compara con los 8 valores posibles, o sea con los 8 códigos
‘de función implementados en el programa, hasta que encuentre uno igual, si no lo
‘halla regresará a la subrutina "Comprobar" cargando el arreglo petición con ceros,
‘para luego permanecer en "Esperar" en busca de una nueva trama del maestro.

'*************************************************************************************************
'    CÓDIGOS DE FUNCIÓN: LECTURA DE BOBINAS (01H)  Y
'                                LECTURA DEL ESTADO DE ENTRADA (02H)
'*************************************************************************************************
'Los códigos de función 01H y 02H son muy similares, debido a eso se incluyeron en
‘una sola subrutina, diferenciándose solamente en la lectura de sus estados.

'Lectura del estado de las salidas a relé y digitales   Función 01H
'Bobina1=PORTA.0 : Bobina2=PORTA.1 : Bobina3=PORTA.2
'Bobina4=PORTA.3 : Bobina5=PORTD.0 : Bobina6=PORTD.1
'Bobina7=PORTD.2 : Bobina8=PORTD.3 : Bobina9=PORTD.4
'Bobina10=PORTD.5
```

```
'Lectura del estado de las entradas discretas   Función 02H
'10001=PORTB.4 : 10002=PORTB.5 : 10003=PORTB.6
'10004=PORTB.7 : 10005=PORTD.6 : 10006=PORTD.7

'Como la cantidad máxima de entradas o salidas no superan el valor de 15, no es
‘necesario que el campo "Número de salidas (H)" sea tomado en cuenta. Por lo tanto
‘el campo "Número de salidas (L)" (peticion[5]) será suficiente para tal tarea.
'Byte Count será 1 o 2 para la función 01H ya que se leerán menos de 8 o más de 8
‘salidas respectivamente, siendo su máximo valor 10 salidas, y 1 para la función 02H
‘ya que solo existen 6 entradas discretas. Una vez calculado el Byte Count lo próximo
‘a hacer es colocarlo en una posición adecuada en la trama de respuesta, para eso
‘reutilizamos el arreglo petición localizando este valor calculado en peticion[2].
‘peticion[3] y peticion[4] corresponden a los datos de los estados de las bobinas para
la función 01H o solamente peticion[3] para los datos de los estados de las entradas
‘discretas en la función 02H.
'La subrutina respuesta se explicará en su momento.

UnoDos:  If peticion[1]<=2 Then
        ' Byte count (N) = Cantidad de salidas / 8, si el resto != 0 -> N = N + 1
        num=peticion[5]//8  ' Cantidad de salidas Lo
        If num! = 0 Then ' Si Byte count (N) es difernte de 0 entonces
          peticion[2]=(peticion[5]/8)+1  ' N = N + 1  Byte count de respuesta
        Else
          peticion[2]=(peticion[5]/8)  ' Byte count de respuesta
        End If
        ' Dirección de bobina o entradas Lo + Número de bobinas o entradas Lo
        num=peticion[3]+peticion[5]-1  ' Número de bobinas o e desde dirección Lo
        pg = 0 : pg1 = 0
        If peticion[1]=1 Then  ' Función 01H
          est[0]=PORTA.0 : est[1]=PORTA.1 : est[2]=PORTA.2
          est[3]=PORTA.3 : est[4]=PORTD.0 : est[5]=PORTD.1
          est[6]=PORTD.2 : est[7]=PORTD.3 : est[8]=PORTD.4
           If Hab = 2 Then
             est[9]=PORTD.5
          Else
```

```
            est [9] = 0
          End If
          For k=peticion[3] to num
             pg=pg+Dcd(pg1)*est[k] : pg1=pg1+1
          Next k
        Else  ' Función 02H
          est1[0]=PORTB.4 : est1[1]=PORTB.5 : est1[2]=PORTB.6
          est1[3]=PORTB.7 : est1[4]=PORTD.6 : est1[5]=PORTD.7
          For k=peticion[3] to num
             pg=pg+Dcd(pg1)*est1[k] : pg1=pg1+1
          Next k
        End If

        peticion[3]=pg&$FF
        If peticion[5]<=8 Then
          peticion [4] = 0
        Else
          peticion[4]=(pg>>8)&3
        End If
        GoSub Resp
      End If

'*************************************************************************************************
'    CÓDIGO DE FUNCIÓN: LECTURA DE REGISTROS HOLDING (03H)
'*************************************************************************************************
'Como la cantidad de registros holding no supera el valor de 11 el campo "Número de
‘registros (H)" no se tomará en cuenta. El campo "Número de registros (L)" servirá
‘para calcular nuestro Byte Count.
'El valor calculado deberá escribirse en peticion[2] que corresponde a la posición 2 de
‘la trama de respuesta. peticion[3] es el campo "Dirección de inicio (L)" y num un valor
‘que restado del contenido de peticion[3] dará como resultado el número de registros
‘a leerse. De esta forma se accede a las posiciones y al número de los registros de
‘interés por parte del maestro. Para guardar o acceder al contenido de un registro se
‘utilizan dos arreglos, registro1 (parte alta del registro, o en otras palabras el byte más
‘significativo) y registro (parte baja del registro, o en otras palabras el byte menos
```

‘significativo), se realiza esto ya que la trama Modbus maneja una longitud de registro ‘de 16 bits.

'Los valores de lectura por parte del Timer1 deben actualizarse constantemente, ‘debido a ello por cada llamada de lectura de algún registro, en la posición 6 de los ‘arreglos registro1 y registro se almacenan los valores de TMR1H y TMR1L.

```
Tres:    If peticion[1]=3 Then
         peticion[2]=2*peticion[5]  ' Byte count = (Número de puntos Lo)*2
         ind=4 : num=peticion[3]+peticion[5]-1
         For k=peticion[3] to num
            registro1 [6] = TMR1H ' Lectura de TMR1H -> registro 40007
            registro [6] = TMR1L ' Lectura de TMR1L -> registro 40007
            peticion[ind-1]=registro1[k]
            peticion[ind]=registro[k]  ' Guardo los datos bajos
            ind = ind + 2
         Next k
         GoSub Resp
       End If

'*************************************************************************************************
'    CÓDIGO DE FUNCIÓN: LECTURA DE UN REGISTRO DE ENTRADA (04H)
'*************************************************************************************************
```

' Existen 4 canales de conversión correspondientes a los registros de entrada 30006 ‘a 30009. Cada canal utiliza una posición del arreglo Ch, de esta forma su ‘direccionamiento se vuelve más sencillo. El campo "Número de registros (L)" sirve ‘para el cálculo del Byte Count, para luego almacenarlo en la posición 2 de la trama ‘de respuesta. peticion[3] almacena el campo "Dirección de inicio (L)", las posibles ‘direcciones de inicio asumiendo que el campo "Número de registros (L)" (peticion[5]) ‘propuesta por el maestro siempre es igual a 1, y a su vez comparándolas con su ‘respectivo registro de entrada, son las siguientes:

'Dirección de inicio (L)	Número de registros (L)	Registro de entrada
' 05H	01H	30006
' 06H	01H	30007
' 07H	01H	30008
' 08H	01H	30009

'De esta manera se establece direcc=peticion[3]-5, ya que si se resta 5 del campo '"Dirección de inicio (L)" los resultados concuerdan con los índices del arreglo Ch 'asociados a cada canal (índices 0 a 3 para los canales 4 a 7).
'num, no es más que el número de canales que serán leídos.
'Ind, sirve para posicionar correctamente el resultado de la conversión (10 bits) del 'canal elegido en la trama de respuesta sobre el mismo arreglo peticion.

```
Cuatro:  If peticion[1]=4 Then
          ' Asignación de canales al arreglo
          Ch[0]=%10100001  ' Canal 4 -> RA5
          Ch[1]=%10101001  ' Canal 5 -> RE0
          Ch[2]=%10110001  ' Canal 6 -> RE1
          Ch[3]=%10111001  ' Canal 7 -> RE2
          peticion[2]=2*peticion[5]  ' ByteCount
          direcc=peticion[3]-5  ' Dirección baja (posibles direc 0 a 3)
          num=peticion[5]-1  ' Número de puntos Lo -> 0 a 3
          For k = 0 To num
             ADCON0=Ch[direcc+k] : ADCON1=%10000000 : PIR1.6=0  ' ADIF=0
             Pauseus 50 : ADCON0 0.2 = 1 ' GO=1
ConvAD:  If PIR1.6=0 Then ConvAD
             ind=2*k+3 : peticion[ind+1]=ADRESL : peticion[ind]=ADRESH&%00000011
          Next k
          Gosub Resp : ADCON1=%00000110  ' Puerto A como I/O digitales
        End If

'*****************************************************************************************************
'    CÓDIGO DE FUNCIÓN: FORZAR UNA BOBINA SIMPLE (05H)
'*****************************************************************************************************
```

'Cada elemento del arreglo est tiene asignada una bobina, el campo "Dirección de 'bobina (L)" (peticion[3]) permite acceder a cada una de estas bobinas por medio de 'direcc que se utiliza como índice del arreglo. El campo "Dato forzado (L)" (peticion[4]) 'establece si dicha bobina estará encendida o apagada, si el dato es FFH o 00H 'respectivamente.

```
Cinco:   If peticion[1]=5 Then
          direcc=peticion[3]  ' Dirección de la bobina a forzarse
          If peticion[4]=$FF Then
            est [direcc] = 1
          Else
            est [direcc] = 0
          End If
          GoSub Escri_B
          If MTS = 0 Then
            pos = 7
          Else
            pos = 6
          End If
          GoSub Resp
        End If

'*****************************************************************************************************
'     CÓDIGO DE FUNCIÓN: PROGRAMAR UN REGISTRO INDIVIDUAL (06H)
'*****************************************************************************************************
'El campo "Dirección de registro (H)" no se utiliza ya que se direccionarán solamente
‘11 registros. El campo "Dirección de registro (L)" (peticion[3]) sirve para acceder al
‘contenido de los arreglos registro1 y registro, que contienen la parte alta y baja del
‘registro solicitado por el maestro. Las posiciones 4 y 5 del arreglo petición se
‘sobrescriben con dichos valores, correspondiendo a las posiciones que deben tener
‘en la trama de respuesta.

Seis:    If peticion[1]=6 Then
          k=peticion[3]  ' Dirección del registro
          registro1[k]=peticion[4]  ' Data hi reg
          registro[k]=peticion[5]  ' Data lo reg
          GoSub Prg_Reg
          If MTS = 0 Then
            pos = 7
          Else
            pos = 6
```

```
        End If
    GoSub Resp
     End If

'*************************************************************************************************
'    CÓDIGO DE FUNCIÓN: FORZAR MULTIPLES BOBINAS (0FH)
'*************************************************************************************************
```

'Se direccionarán un máximo de 10 bobinas, por lo tanto el campo "Dirección de ‘bobina (H)" no será utilizado. Los campos "Dirección de bobina (L)" y "Número de ‘bobinas (L) sirven para determinar el número de elementos desde una dirección ‘específica. Dos son los posibles valores de Byte Count, 0 o 1 dependiendo de la ‘cantidad de bobinas a forzarse (campos "Datos forzados(H)" - peticion[7] y "Datos ‘forzados(L)" peticion[8]), este dato se almacenará en pg para luego trasformarlo a su ‘equivalente en binario y guardar cada bit de este número en una posición del arreglo ‘est, para terminar asignando cada elemento del arreglo a una salida física ‘correspondiente a cada bobina.

```
Quince:  If peticion[1]=$F Then
          ' Dirección de bobina Lo + Número de bobinas Lo
          num=peticion[3]+peticion[5]-1  ' Número de bobinas desde dirección Lo
          If peticion[6]=1 Then  ' Examino el valor de Byte Count
            pg=peticion[7] ' Bobinas a forzarse (byte)
          Else
            pg=(peticion[8]<<8)+peticion[7] ' Bobinas a forzarse (palabra)
          End If
          For k=peticion[3] to num
             est[k]=pg//2 : pg=pg/2
          Next k
          GoSub Escri_B : GoSub Resp
        End If

'*************************************************************************************************
'    CÓDIGO DE FUNCIÓN: PROGRAMAR MÚLTIPLES REGISTROS (10H)
'*************************************************************************************************'
```

’Algo similar a lo ocurrido en la función 0FH sucede en esta función. Los campos

‘asociados a peticion[3] y peticion[5] se utilizan con la misma finalidad, pero aquí sus ‘nombres cambian, siendo "Dirección de inicio (L)" y "Número de registros (L)".
'Los valores de escritura que manda el maestro son almacenados en registro1 y ‘registro, luego la subrutina "Prog_reg" determina la acción a tomar.

```
Dyseis:  If peticion[1]=$10 Then
          ' Dirección de inicio Lo + Número de registros - 1
          num=peticion[3]+peticion[5]-1 : ind=7
          For k=peticion[3] to num
             registro1[k]=peticion[ind]  ' Dato alto
             registro[k]=peticion[ind+1]  ' Dato bajo
             GoSub Prog_Reg : ind = ind + 2
          Next k
          GoSub Resp
        End If
      Return

'*****************************************************************************************************
'    SUBRUTINA QUE IMPRIME LA RESPUESTA A UNA PETICIÓN
'*****************************************************************************************************
```

' El número de elementos necesarios para el cálculo del CRC o LRC para los códigos ‘de función 01H a 04H es el mismo, y se encuentran relacionados con el Byte Count ‘calculado en cada una de ellas. La variable pos no es más que la posición en la que ‘dicho valor va a ser colocado sobre la trama de respuesta.
‘Para los códigos de función 05H y 06H el cálculo del CRC o LRC es innecesario ya ‘que su respuesta es solamente un eco de la petición del maestro.

'Para los códigos de función 0FH y 10H el número de elementos necesarios para el ‘cálculo del CRC o LRC, así como su posición en la trama de respuesta, es la misma.

'La próxima acción a realizarse es elaborar la respuesta en si, para eso se genera el ‘caracter de inicio de trama (:), el cuerpo de la trama y por último los caracteres de fin ‘de trama (CR y LF) si el Modo de Transmisión Serial elegido hubiera sido el ASCII, ‘de lo contrario solo será necesario enviar el cuerpo de la trama.

```
Resp:    If peticion[1]<=4 Then
           num=peticion[2]+2
           GoSub CRCLRC : pos = num + 1
         End If
         If peticion[1]>=$F Then
           num = 5 : GoSub CRCLRC
         End If

         If MTS = 0 Then
           If peticion[1]<=4 Then
             peticion[pos]=pg&$0FF  ' Guardo CRC calculado en peticion
             peticion[pos+1]=(pg&$FF00)>>8 : pos=pos+1
           End If
           If peticion[1]>=$F Then
             peticion[6]=pg&$0FF : peticion[7]=(pg&$FF00)>>8 : pos=7
           End If
           ' Imprime el cuerpo de la trama
           For k = 0 To pos
              pg1=peticion[k] : Hserout[pg1]
           Next k
         Else
           If peticion[1]<=4 Then
             peticion [pos] = pg2 ' Guardo LRC calculado en peticion
           End If
           If peticion[1]>=$F Then
             peticion [6] = pg2 : pos = 6
           End If
           ' Imprime el caracter de inicio de trama (:)
           Hserout [":"]
           ' Imprime el cuerpo de la trama
           For k = 0 To pos
              pg1=(peticion[k]&$F0)>>4 : Gosub DAsccih  : Hserout [pg2]
              pg1=peticion[k]&$F : Gosub DAsccih : Hserout [pg2]
           Next k
           ' Imprime los caracteres de fin de trama (CR y LF)
```

```
        Hserout [13,10]
      End If  :  Return
'*****************************************************************************************
'    SUBRUTINA QUE PASA DATOS DEl ESTADO DE LAS BOBINAS
'*****************************************************************************************
'Cada elemento del arreglo est guarda el estado de una bobina cuya dirección no es
'más que el índice del arreglo.
'Como se ve estos elementos se asignan a salidas físicas del PIC, sean estas salidas
‘a relé o digitales. La escritura de la bobina10 también se pierde si el estado de la
‘Red está habilitado.

Escri_B: PORTA.0=est[0] : PORTA.1=est[1] : PORTA.2=est[2]
      PORTA.3=est[3] : PORTD.0=est[4] : PORTD.1=est[5]
      PORTD.2=est[6] : PORTD.3=est[7] : PORTD.4=est[8]
      If Hab = 2 Then
        PORTD.5=est[9]
      End If
      Return

'*****************************************************************************************
'    SUBRUTINA QUE PASA DATOS PARA ESCRIBIR Y LEER REGISTROS
'*****************************************************************************************
'En k se guarda el valor de peticion[3] (Dirección de registro (L)).
'La escritura del Registro 40007 (k=6) asigna un valor al Timer1 siempre y  cuando la
‘bandera STimer esté habilitada.
'El manejo de la señal PWM (k=7) es simple, se realiza seteando los bits 4 y 5 del
‘registro CCP1CON y todo el registro CCPR1L.

'El registro de visualización 40009 (k=8) muestra en el LCD el valor enlazado al
‘mismo. Para ver su contenido se transforma este valor hexadecimal a su equivalente
‘en BCD.
'El registro holding 40010 (k=9) direcciona la carga a los registros T1CON, T2CON y
‘PR2, se puede observar que a parte de esta carga estos valores son almacenados
‘en la EEPROM de datos.
```

'El registro holding 40011 (k=10) de lectura de registros puede alternar entre el ‘despliegue del contenido de T1CON, T2CON, TIMER1 y PR2.

```
Prog_Reg: If k = 6 Then ' Registro 40007 - Escritura de TMR1H:TMR1L
          If STimer = 1 Then
            TMR1H=registro1[6] : TMR1L=registro[6] : STimer=0
          End If
        End If
        If k = 7 Then ' Registro 40008 - Manejo de PWM
          pg1=registro[7] : CCP1CON.4=pg1.0 : CCP1CON.5=pg1.1
          CCPR1L=(registro1[7]<<6)+(pg1>>2)
        End If
        If k = 8 Then ' Registro de visualización 40009 (LCD)
          If Nop = 0 Then
            Lcdout $FE,$80," REGISTRO 40009 " : pg=(registro1[8])<<8+registro[8]
            pg1=pg/10000 : pg=pg//10000 : pg2=pg/1000 : pg=pg//1000
            pg3=pg/100 : pg=pg//100
            Lcdout $FE,$C5,dec pg1,dec pg2," ",dec pg3,dec (pg/10),dec (pg//10),"    "
          End If
        End If
        If k = 9 Then ' Registro 40010 - Escritura de Timers y PR2
          If registro1[9]=2 Then  ' Configuración de T1CON
            Write Dato_T1CON,registro[9] : T1CON=registro[9]
          End If
          If registro1[9]=3 Then  ' Escritura de TMR1H:TMR1L
            STimer = 1: Nop = 1
          End If
          If registro1[9]=4 Then  ' Configuración de T2CON
            CCP1CON=%00001100  ' Modo PWM habilitado
            Write Dato_T2CON,registro[9] : T2CON=registro[9]
            If T2CON.2=0 Then  ' Modo PWM deshabilitado
              CCP1CON = 0: PORTC 0.2 = 0
            End If
          End If
          If registro1[9]=5 Then  ' Escritura de PR2
```

```
      Write Dato_PR2,registro[9] : PR2=registro[9]
    End If
  End If
  If k = 10 Then ' Registro 40011 - Lectura de registros
    If registro[10]=3 Then
      Lcdout $FE,1 : direcc=Mensj_TMR1HL : Gosub LCD
      Lcdout $FE,$C5,dec (TMR1H<<8+TMR1L)," H    " : Nop=1
    Else
      If registro[10]=1 And Nop=1 Then
        Lcdout $FE,1 : direcc=Mensj_Esclavo : Gosub LCD
        Lcdout,$FE,$89,dec esclavo," OK   " : Nop=0
      End If
      If registro[10]>=2 Then
        If registro[10]=2 Then
          direcc = Mensj_T1CON: pg2 = Dato_T1CON
        End If
        If registro[10]=4 Then
          direcc = Mensj_T2CON: pg2 = Dato_T2CON
        End If
        If registro[10]=5 Then
          direcc = Mensj_PR2 : pg2 = Dato_PR2
        End If
        Lcdout $FE,1 : Gosub LCD : Read pg2,pg3
        Lcdout $FE,$C2,hex pg3," H - ", dec pg3," D" : Nop=1
      End If
    End If
    If registro[10]=6 Then
      pg=0 : Lcdout $FE,1 : direcc=Mensj_Tesis : Gosub Lazo
      Lcdout $FE,$80 : direcc=Mensj_Implementacion : Gosub LCD
      Lcdout $FE,$C0 : direcc=Mensj_Protocolo : Gosub Lazo
      Lcdout $FE,1 : direcc=Mensj_Integrantes : Gosub Lazo
      Lcdout $FE,$80 : direcc=Mensj_Mi_nombre : Gosub LCD
      Lcdout $FE,$C0 : direcc=Mensj_Nombre_de_Panchito : Gosub Lazo
      Lcdout $FE,1 : direcc=Mensj_Esclavo : Gosub LCD
      Lcdout,$FE,$C8,dec esclavo," OK   "
```

```
        End If
      End If
      Return
'*************************************************************************************************
'    SUBRUTINA QUE CONVIERTE DE DEC A ASCII HEX
'*************************************************************************************************
DAsccih: Lookup pg1, ["0123456789ABCDEF"], pg2

      Return

'*************************************************************************************************
'    SUBRUTINA QUE GENERA UN RETARDO (NO SE PUEDE EMITIR
'    RESPUESTA)
'*************************************************************************************************
'Esta subrutina da una medida de tiempo para el despliegue de mensajes de la
'carátula. Como se utilizan tiempos demasiado grandes una respuesta por parte del
'esclavo es nula.

Lazo:  GoSub LCD : pg = pg + 1
      If pg = 35000 Then
        pg = 0
      Else
        GoTo Lazo
      End If

      Return

'*************************************************************************************************
'    FIN DEL PROGRAMA
'*************************************************************************************************
```

3.3 DISEÑO DE LA INTERFASE HUMANO MÁQUINA (HMI)[3]

3.3.1 El Objeto MODBUS

Modbus es una clase de driver de protocolo que Lookout usa para comunicarse con equipos tales como controladores lógicos programables (PLCs), unidades terminales remotas (RTUs) u otros equipos que utilicen Modbus Serial (ASCII o RTU) o el protocolo de comunicaciones Modbus Plus.

Los parámetros del objeto Modbus a configurarse se muestran en la figura 3.10.

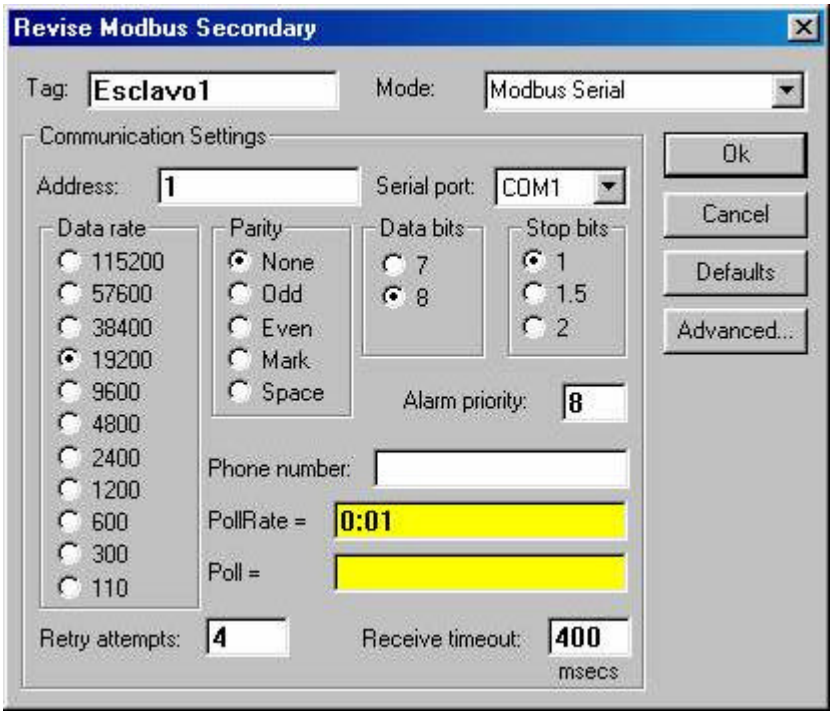

Figura 3.10. Parámetros de configuración Modbus

Muchos de estos parámetros son familiares: la configuración del puerto serial (Bits por segundo o Data rate, Bits de datos, Paridad y Bits de parada), Dirección del Esclavo y Modo de Transmisión Serial (Modbus Serial en este caso).

PollRate. Es una expresión numérica que determina cuan a menudo se sondeará al dispositivo en un determinado intervalo de tiempo. La medida más utilizada corresponde a 1 segundo, (0:01).

Retry attempts. Especifica el número consecutivo de veces que Lookout tratará de establecer comunicaciones con un dispositivo. Si no se obtiene una respuesta válida después de dicho número de veces, el objeto Modbus generará una alarma.

Receive timeout. Es el tiempo de demora que Lookout usa en espera de una respuesta del dispositivo antes de reintentar la petición.

El driver Modbus procura agrupar en bloques las lecturas y escrituras de bobinas, registros de entrada y registros holding. El objetivo es mejorar la eficiencia de la comunicación.

Utilizando el cuadro de diálogo de la figura 3.11, el usuario puede controlar los valores máximos por mensaje que el objeto Modbus manejará dentro de la aplicación, de acuerdo a los códigos de función implementados.

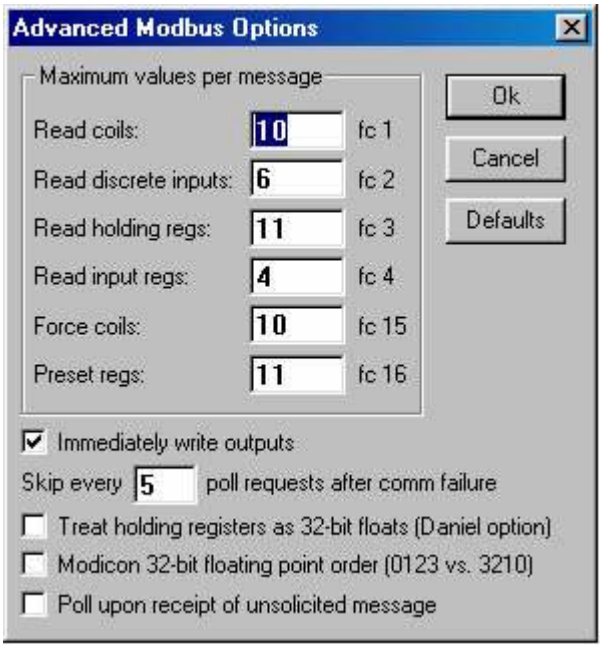

Figura 3.11. Configuración avanzada Modbus

Los valores máximos para este proyecto fueron llenados con el contenido de la tabla 3.1.

3.3.2 Salidas (Bobinas)

La interfase en Lookout que maneja las salidas a relé, salidas digitales y sus respectivos estados es la que se muestra en la figura 3.12

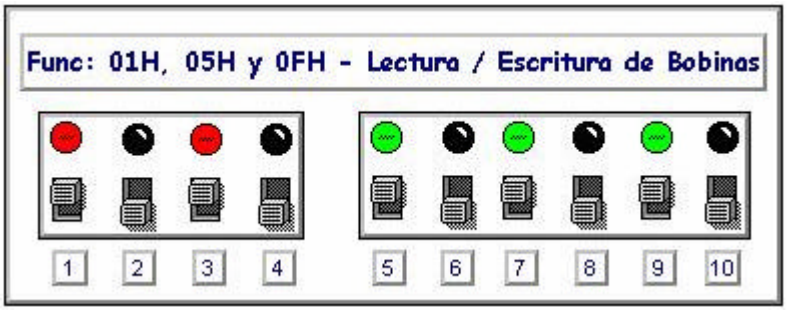

Figura 3.12. Visualización de bobinas en Lookout

Dichos estados se perciben en la interfase como indicadores luminosos (Revise expression) del encendido o apagado de la bobina 1 a bobina 10.

La figura 3.13 muestra como debe llenarse el campo de un Revise expression para determinar el estado de la bobina 1.

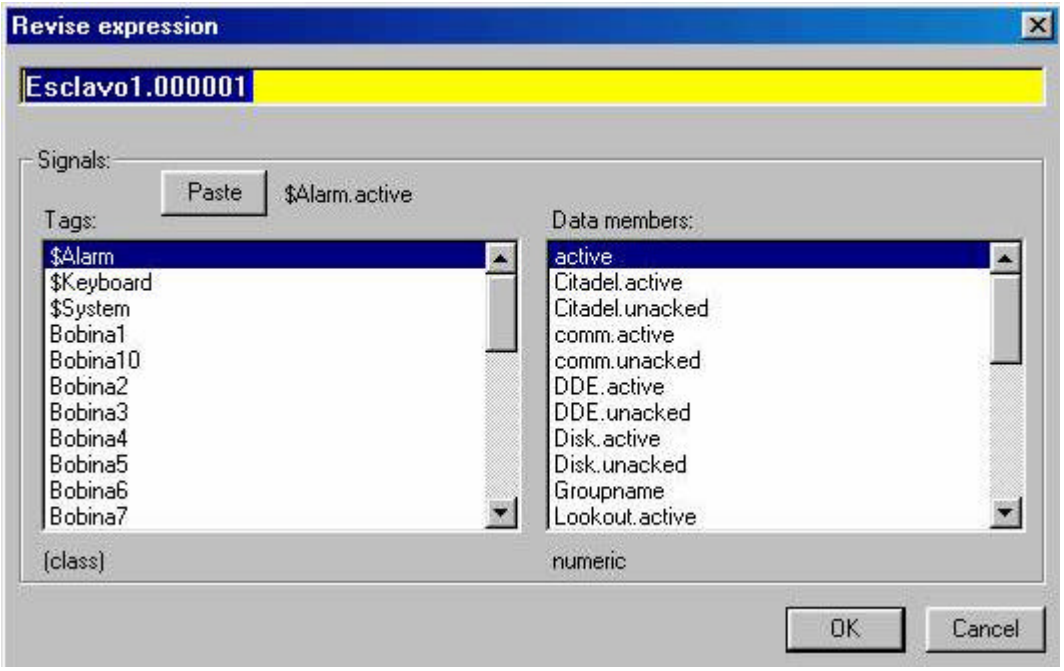

Figura 3.13. Forma de llenar el campo de un Revise expression para conocer el estado de una bobina

Para la lectura del resto de las bobinas el procedimiento es similar.

Para controlar el encendido y apagado de bobinas, se enlazan objetos tipo Switch a cada una de las salidas del objeto Modbus (000001 a 000010). Un objeto Switch genera una señal lógica cuyo estado cambia al hacer clic sobre el botón del mouse o la barra espaciadora del teclado.

Los objetos tipo Switch están etiquetados (tag) en la interfase implementada como Bobina1 a Bobina10.

La figura 3.14 muestra como debe llenarse el campo de un Revise expression para tomar el control de la bobina 1 empleando un Switch etiquetado como Bobina1.

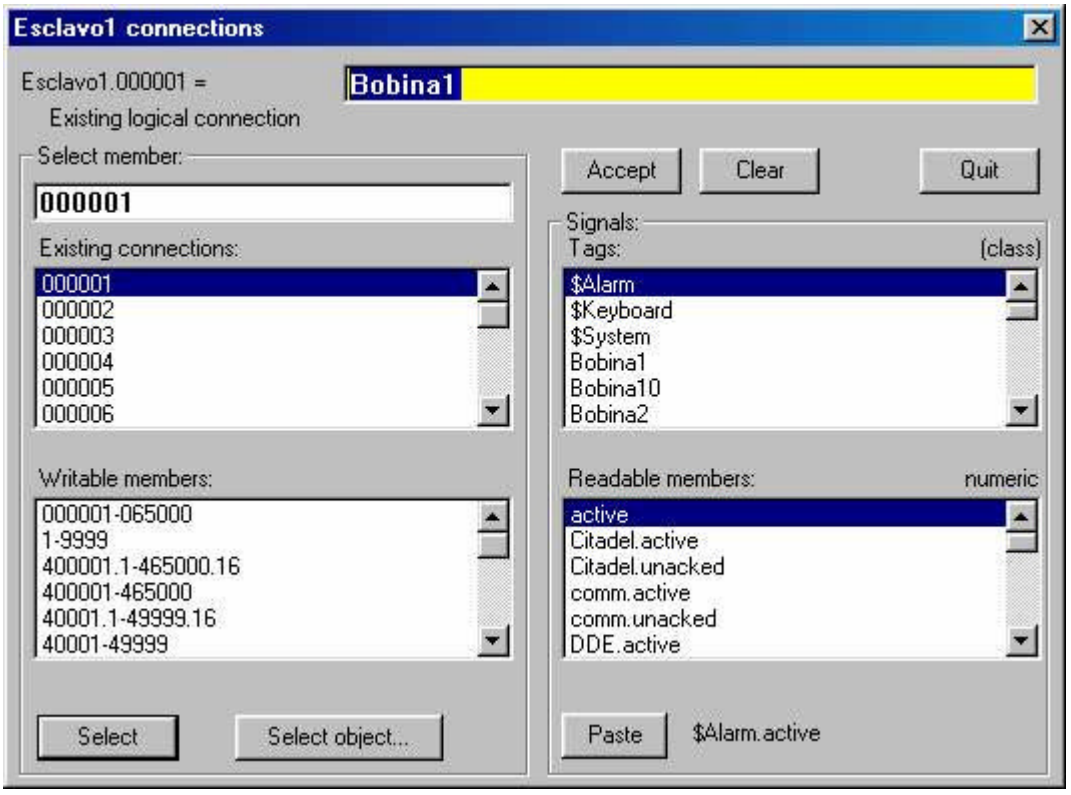

Figura 3.14. Enlace de una bobina a un objeto Switch

El enlace de las bobinas restantes es similar a lo anteriormente expuesto.

3.3.3 Entradas Digitales

La interfase en Lookout que maneja los estados de las entradas discretas es mostrada en la figura 3.15.

Figura 3.15. Visualización de entradas digitales en Lookout

Los indicadores luminosos (Revise expression) presentan los estados de las entradas discretas 10001 a 10006 (Función 02H).

La figura 3.16 muestra como debe llenarse el campo de un Revise expression para determinar el estado de la entrada 10001.

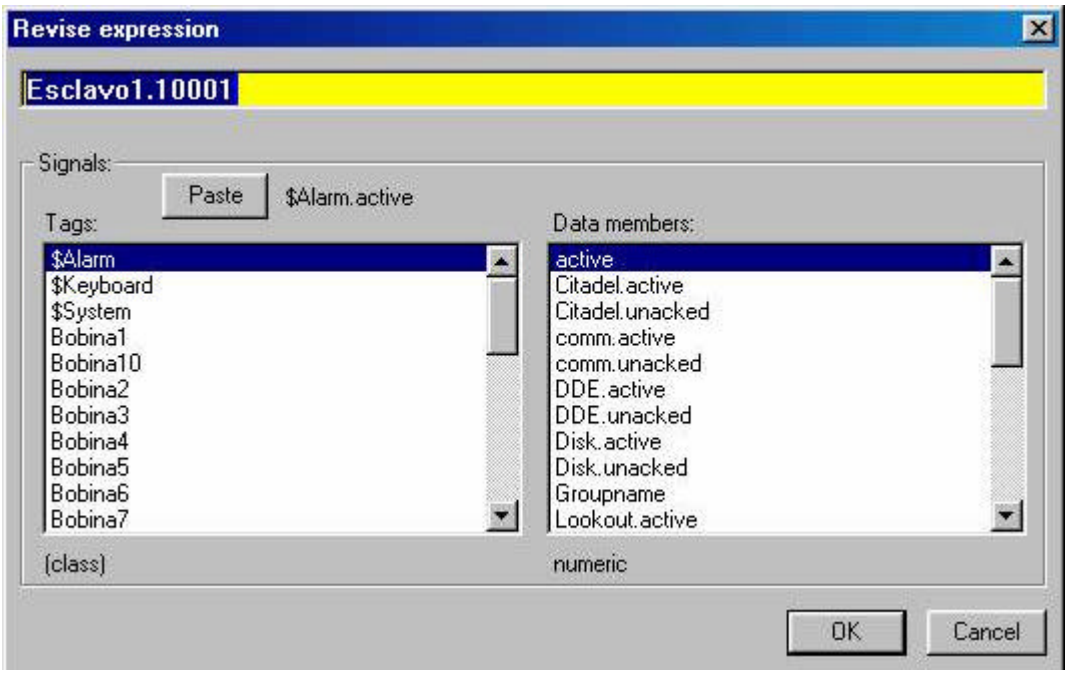

Figura 3.16. Forma de llenar el campo de un Revise expression para conocer el estado de una entrada digital

Para la lectura de las demás entradas digitales el proceso es similar.

3.3.4 Entradas Analógicas

La interfase en Lookout que maneja los registros de entrada 30006 a 30009 (Función 04H) se presenta en la figura 3.17.

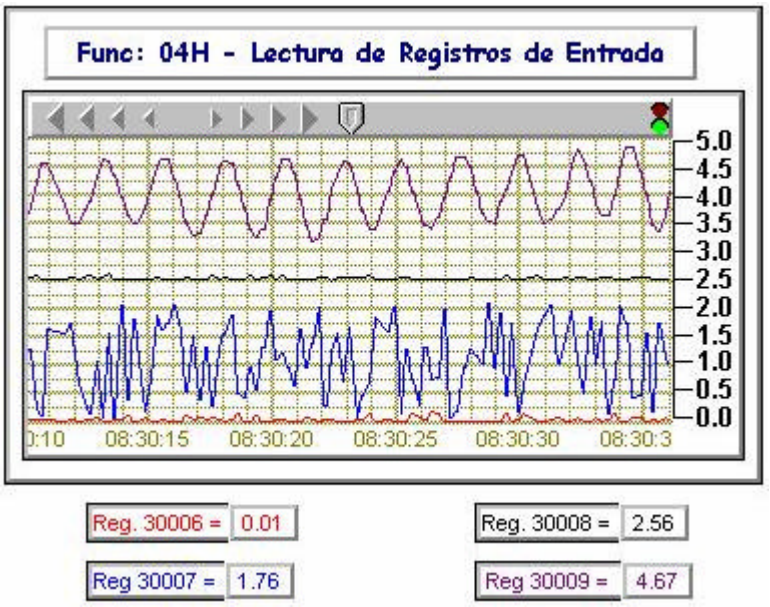

Figura 3.17. Visualización de registros de entrada en Lookout

Figura 3.18. Enlace de un registro de entrada a un objeto Hyper Trend

Se utilizaron un objeto Hyper Trend y cuatro Revise expressions para el despliegue gráfico y numérico de las señales analógicas, que entran al microcontrolador.

El registro de entrada 30006 enlazado al objeto Hyper Trend se muestra en la figura 3.18. Los demás registros de entrada están enlazados de la misma manera, siendo en total cuatro Trendlines (Esclavo1.30006 a Esclavo1.30009).

La figura 3.19 muestra como debe llenarse el campo de un Revise expression para determinar el valor del registro de entrada 30006.

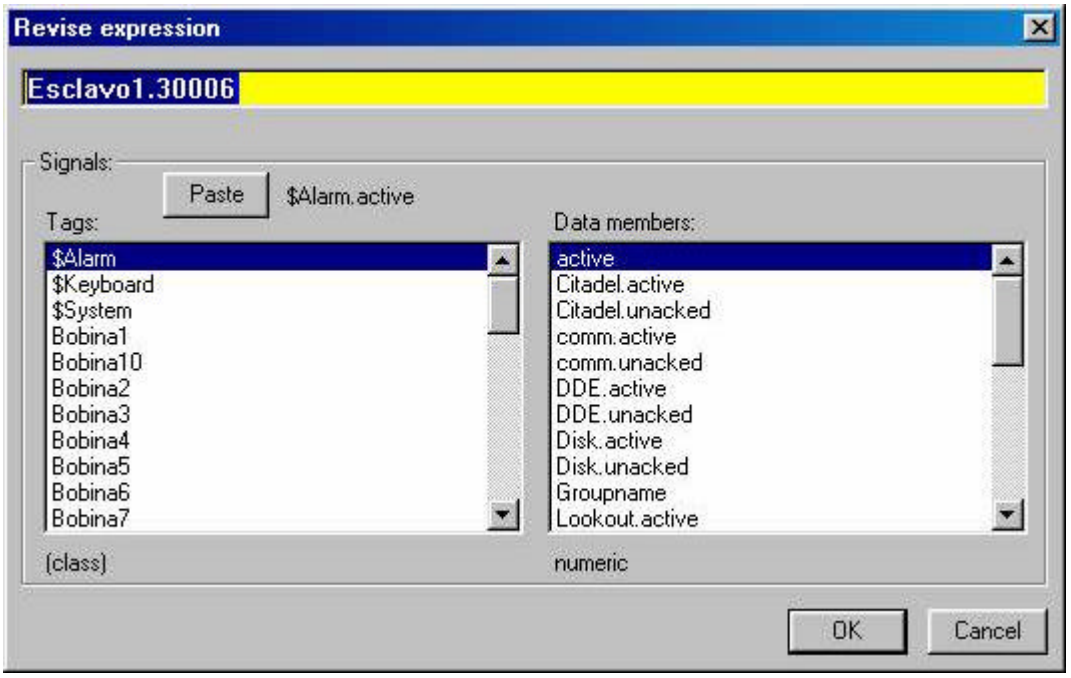

Figura 3.19. Forma de llenar el campo de un Revise expression para conocer el valor de un registros de entrada

Para la lectura de los valores de los registros de entrada restantes el proceso es similar.

Como los registros de entrada son de tipo numérico y no lógico como las usadas en la lectura de los estados de las entradas discretas y salidas, es necesario definir sus valores máximo y mínimo de despliegue, siendo 0 el mínimo

valor y 1023 el valor máximo (tal como se muestra en la figura 3.20) ya que la resolución del módulo de conversión análogo digital del PIC es de 10 bits

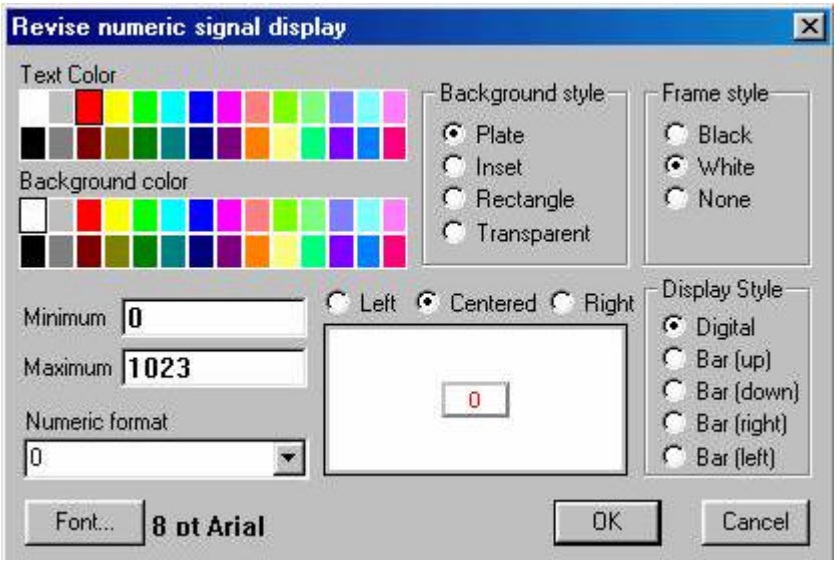

Figura 3.20. Valores mínimo y máximo de despliegue para un registro de entrada

Utilizando como referencia la figura 3.19, figura 3.20 y ANEXO B (diagrama del circuito de la placa microcontrolada) se detalla lo siguiente:

Si se conoce que el voltaje de saturación de un amplificador operacional (en este caso el integrado KIA324P), es aproximadamente inferior en dos unidades al voltaje de alimentación (en este caso el voltaje proporcionado por el potenciómetro R25), se puede inferir fácilmente que el voltaje de polarización del amplificador operacional deberá ser de aproximadamente 7V, para que una señal de 5V en su entrada se vea correctamente reflejada a su salida.

En lugar de utilizar las características eléctricas propuestas por el fabricante para el integrado KIA324P (cálculo del voltaje de alimentación con respecto al de saturación), se propone a continuación un método práctico más sencillo para llevar a cabo esta tarea.

3.3.4.1 Obtención del voltaje de alimentación para el integrado KIA324P

1. Mover el potenciómetro R25 etiquetado en la placa como LM324 en contra de las manecillas del reloj hasta llegar al tope mínimo (figura 3.21).

Figura 3.21. Poteciómetro del integrado KIA324P

2. Conectar 5V en la entrada analógica 30006. Esta tensión puede ser obtenida fácilmente por el borne etiquetado en la placa implementada como 5V (figura 3.22).

Figura 3.22. Bornera 5V-GND

3. Utilizando un Revise expression enlazar ésta con el registro de entrada 30006 tal como se muestra en la figura 3.19, luego definir sus valores máximo y mínimo de despliegue tal como lo indica la figura 3.20.

4. Girar el potenciómetro según las manecillas del reloj y mirar como se incrementa el Revise expression. Seguir de esta forma hasta obtener la lectura 1023 (5V). Una vez logrado esto el microcontrolador se encontrará protegido de posibles señales que superen el máximo permitido (5V).

3.3.5 Registros Holding

La interfase en Lookout que maneja la lectura y escritura de registros holding se muestra en la figura 3.23.

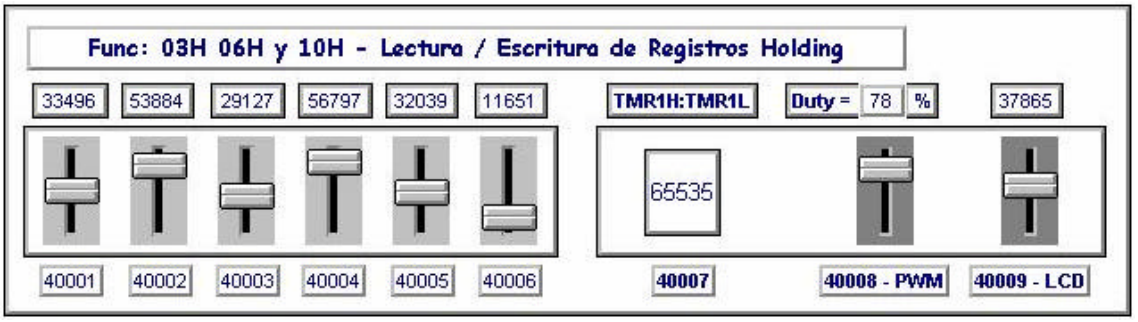

Figura 3.23. Visualización de registros holding en Lookout

Se utilizaron Revise expressions para la lectura de los registros 40001 a 40007 y 40009 (función 03H) enlazándose con el objeto Modbus tal como se presenta en la figura 3.24.

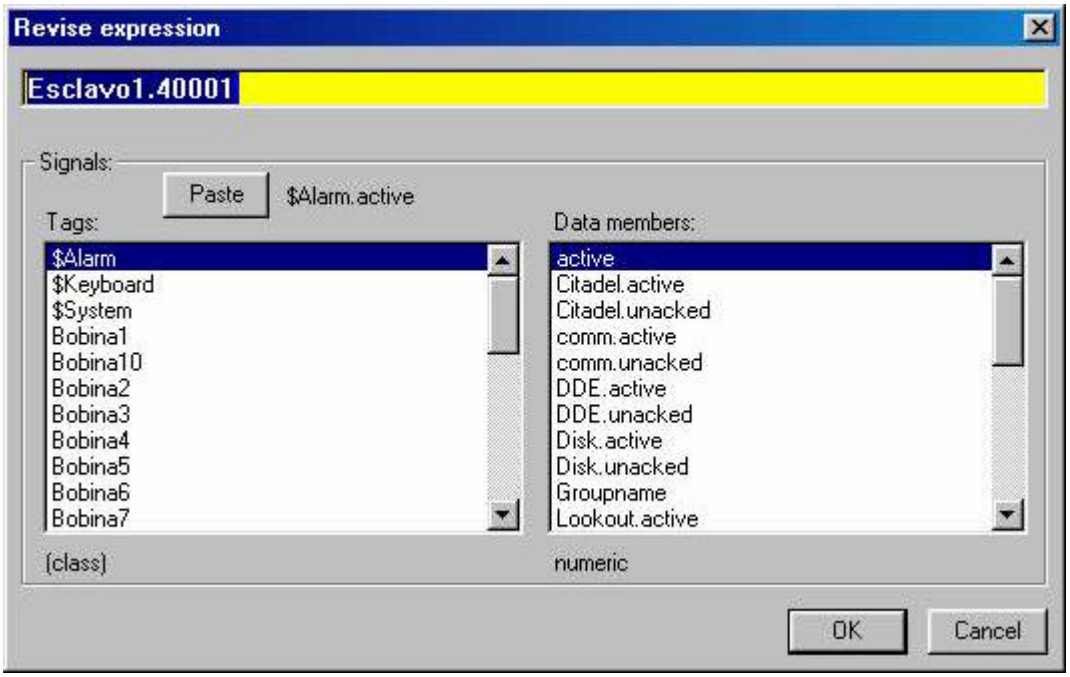

Figura 3.24. Forma de llenar el campo de un Revise expression para conocer el valor de un registro holding

La lectura de los registros holding restantes se realiza de manera similar.

Todos lo registros holding tienen una longitud de 16 bits. Por lo tanto, el rango de variación puede ser desde 0 a un máximo de 65535 para cada uno de ellos.

Para la escritura de todos los registros holding (funciones 06H y 10H) se utilizaron Pots en forma de deslizadores verticales, exceptuando los registros 40007 y 40010, donde se emplearon Pots en forma de entradas digitales y el registro 40011 que utiliza un Radio Buttons.

Todos los Pots tienen un nombre que corresponde a la función que realiza, de esta forma los registros holding 40001 a 40006, 40008 y 40009 están etiquetados como Reg_hold1 a Reg_hold6, Reg_hold8 y Reg_hold9.

En la figura 3.25 se muestra el enlace del registro 40001 al objeto Modbus cuya escritura se lleva a cabo por medio de un Pot etiquetado como Reg_hold1.

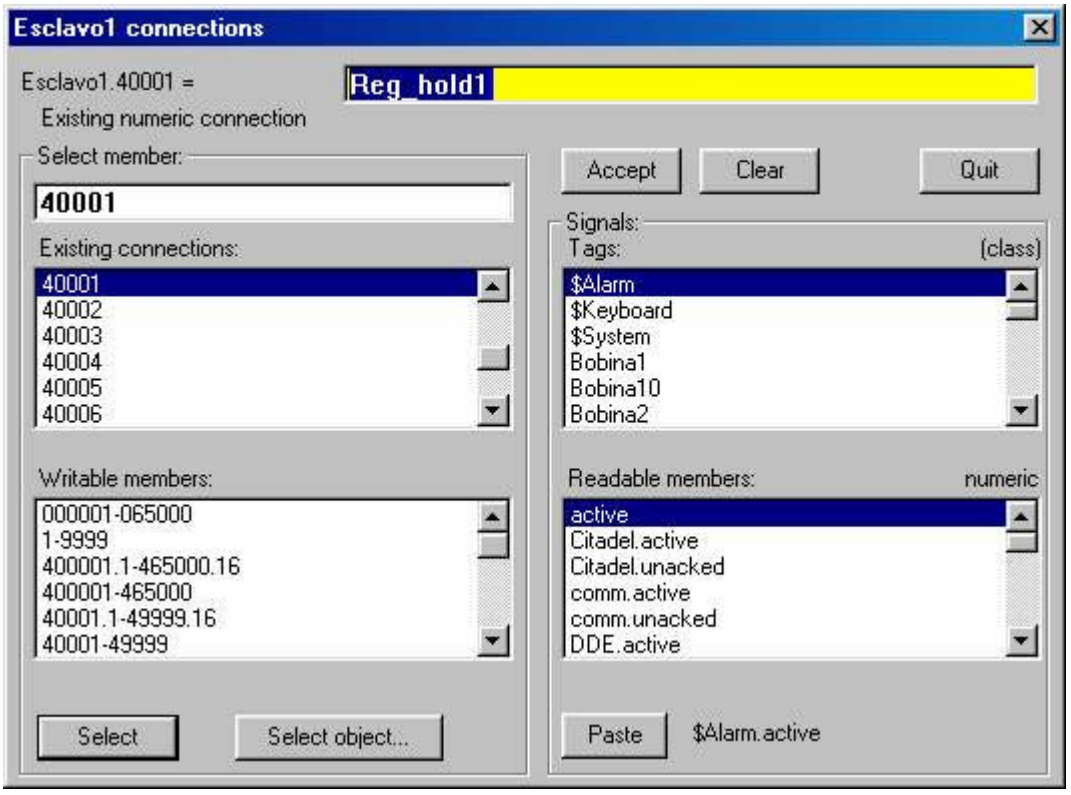

Figura 3.25. Enlace de un registro holding a un objeto Pot

La escritura de los demás registros holding está hecha de manera análoga.

El Pot llamado Reg_hold8 está unido al registro holding 40008, que es el encargado de enviar valores a los registros internos del microcontrolador

asociados al control de la señal PWM (CCPR1L:CCP1CON<5:4>), sus valores mínimo y máximo (0 y 500) son consecuencia de los cálculos hechos con los parámetros, Frecuencia PWM, Predivisor de frecuencia del TMR2 y PR2.

El registro holding 40009 muestra en la pantalla de cristal líquido ubicada en la placa implementada, cualquier variable que se haya enlazado a el, sean registros de entrada, registros holding o incluso un simple Pot.

La interfase en Lookout que maneja la lectura y escritura de los registros internos del PIC se muestra en la figura 3.26.

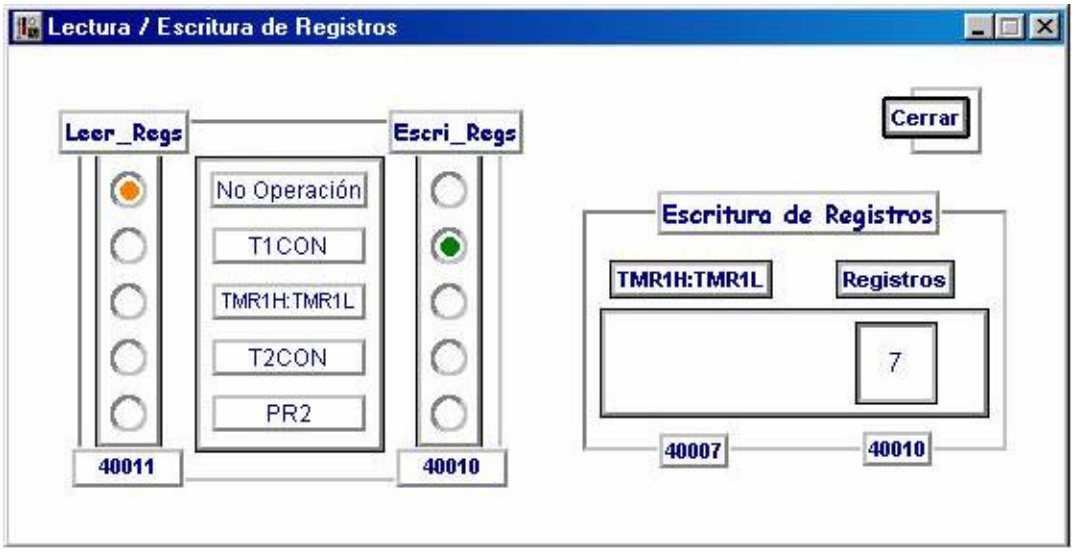

Figura 3.26. Interfase en Lookout para la lectura y escritura de registros internos del PIC

3.3.5.1 Escritura de los registros internos del PIC

Los registros holding 40007 y 40010 están encargados de la escritura de los registros internos del PIC (T1CON, TMR1H:TMR1L, T2CON y PR2) teniendo asociadas las etiquetas Direc_Escri (Dirección de Escritura) y Escri_Regs (Escritura de Registros) para el registro 40010, y Escri_TMR1 (Escritura del TMR1) para el registro 40007.

La longitud del contenido de los registros en las funciones 06H y 10H son de 16 bits y sus valores se almacenan en dos campos de 8 bits, Dato (H) y Dato (L),

dentro de la trama Modbus. Aprovechando esta característica podemos escribir los diferentes registros propios del PIC16F877 utilizando el registro holding 40010 reservado para este propósito. De esta forma el campo Dato (H) sirve de direccionamiento de carga del registro interno T1CON, TMR1H:TMR1L, T2CON o PR2, mientras que el campo Dato (L) se destina al almacenaje del valor de carga del registro T1CON, T2CON o PR2.

La única excepción es cuando se quiere pasar un valor a TMR1H:TMR1L, el campo Dato (L) por lo tanto será insuficiente para cumplir dicho propósito ya que la longitud de TMR1H:TMR1L es de 16 bits, para llevar a cabo tal fin se utiliza toda la longitud del registro holding 40007.

3.3.5.1.1 No Operación

Lookout actualiza las variables cada cierto tiempo (lectura y escritura) o cuando sucede una transición en las mismas. No hay problema si se rescribe continuamente con un mismo valor T1CON, T2CON o PR2 ya que no interferirían en el trabajo del microcontrolador.

El problema radica cuando se carga el contenido del TMR1 debido a que el valor de carga inicial será actualizado en intervalos establecidos por Lookout, llegando a perderse por ejemplo el conteo de eventos externos, suponiendo que la aplicación en uso sea esa. Por tal motivo se creó un estado adicional al de escritura de registro que es No Operación. Lookout entonces actualiza el valor que tiene el registro holding 40010, direccionando la petición del maestro a una posición del código dentro del PIC en la cual no gestiona registro interno o periférico, generando solamente una respuesta.

3.3.5.1.2 Valores que adopta el registro de direccionamiento de carga 40010

El estado de No Operación y cada registro interno del microcontrolador, tienen asignado un valor de direccionamiento que el maestro enviará al dispositivo esclavo dentro de la trama Modbus en el campo Dato (H), cuando exista una

petición de escritura del registro holding 40010. En pocas palabras, el campo Dato (H) viene a ser equivalente al byte más significativo del registro 40010.

La tabla 3.3 muestra los diferentes valores que puede adoptar el campo Dato (H) de acuerdo a los registros antes mencionados.

Tabla 3.3. Valores que adopta el campo Dato (H) del registro 40010

Registros - PIC	Registro 40010 Dato (H)
No Operación	00000001
T1CON	00000010
TMR1	00000011
T2CON	00000100
PR2	00000101

Para No Operación y TMR1 el campo Dato (L) (byte menos significativo) carece de importancia, en los registros restantes el campo señalado será cargado con el valor de escritura elegido por el usuario en la interfase en Lookout.

Volviendo a la figura 3.26. La columna Escri_Regs permite seleccionar el registro a escribirse utilizando para esto el Radio Buttons llamado Direc_Escri, su valor actual (Direc_Escri.current) es multiplicado por 256 para direccionar la carga (byte más significativo) de No Operación, T1CON, TMR1H:TMR1L, T2CON o PR2, mientras el Pot etiquetado como Escri_Regs pasa un valor de carga (byte menos significativo) al registro T1CON, T2CON o PR2, tal como se muestra en la figura 3.27.

Para escribir los registros TMR1H:TMR1L el registro 40010 solo sirve para direccionar la carga, mientras que el valor de carga se lo hace mediante el Pot Escri_TMR1 tal como se observa en la figura 3.28. Los Pots etiquetados como Escri_TMR1 y Escri_Regs se hacen o no visibles de acuerdo a la opción escogida en el Radio Buttons Direc_Escri.

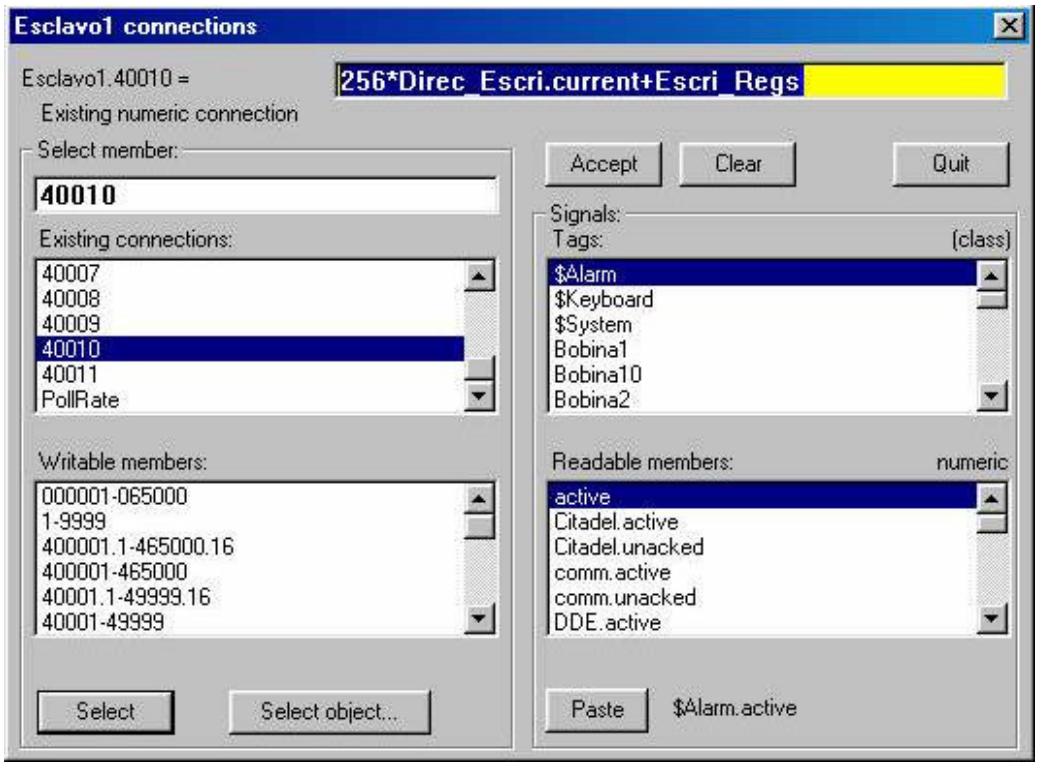

Figura 3.27. Enlace del registro de direccionamiento de carga 40010

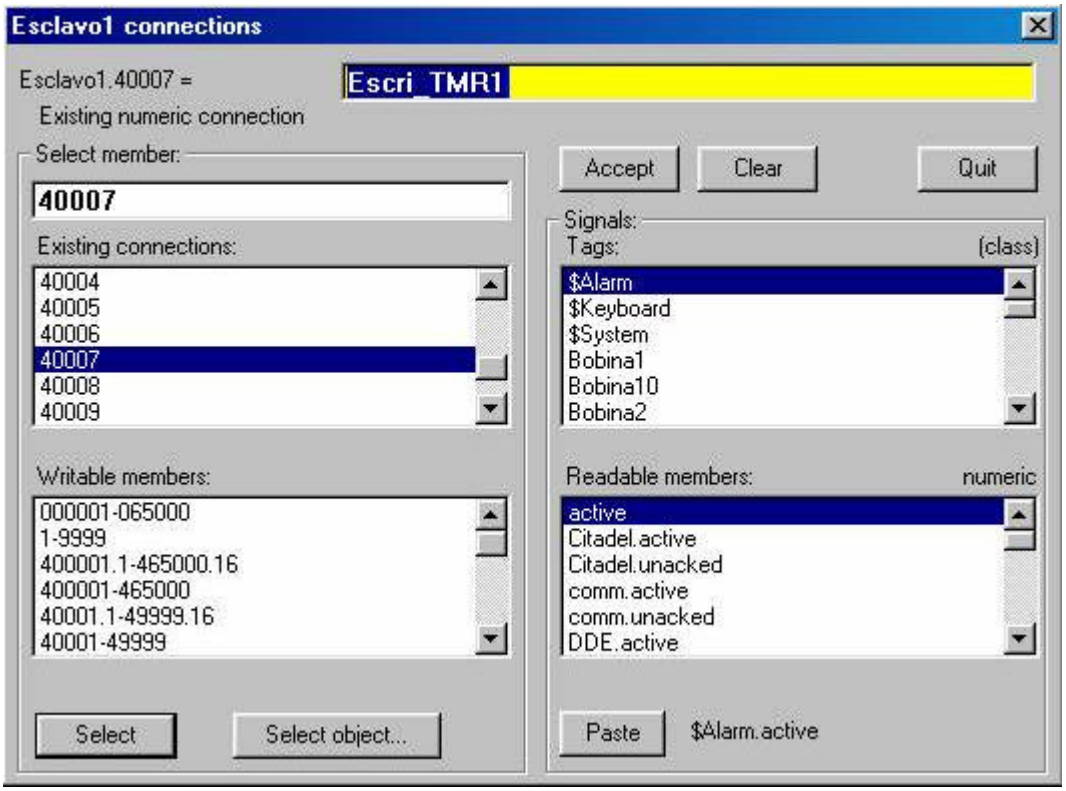

Figura 3.28. Enlace del registro holding 40007 al objeto Pot etiquetado como Escri_TMR1

3.3.5.1.3 Procedimiento para la escritura de los registros internos del PIC

1. Dentro de las opciones que presenta la columna Escri_Regs debe asegurarse que la opción No operación es la que está actualmente presente.

2. Escoger cualquier registro que se desee escribir bajo la columna Escri_Regs, cargar con un valor dicho registro utilizando el Pot Escri_Regs o el Pot Escri_TMR1 según sea el caso. Recordar que Lookout toma el valor de direccionamiento (Direc_Escri.current) como un cambio en el valor del registro 40010. Por lo tanto, el valor que posee en ese instante Escri_Regs o Escri_TMR1 pasará directamente al dispositivo esclavo. En pocas palabras, una selección del registro a escribirse implica ya una escritura del registro escogido, perdiéndose el valor anteriormente almacenado en el mismo sin que el usuario tenga control sobre tal procedimiento. Se recomienda especial atención cuando se direcciona la escritura de un registro porque una equivocación implicaría un valor de carga no deseado.

3. Escoger nuevamente la opción No operación. Si se desea escribir otro registro solamente repetir los pasos 2 y 3.

Un caso especial puede ocurrir para la escritura de los registros TMR1H:TMR1L, se trata de que el valor de carga no se ve reflejado en la lectura del registro 40007, generalmente sucede cuando dicho valor de carga es el mismo que el que tenía anteriormente el Pot etiquetado como Escri_TMR1.

Para resolver este problema se recomienda cargar este par de registros concatenados con un valor cualquiera diferente al que se trató de escribir en un principio siguiendo los pasos 1, 2 y 3.

Repetir dicho procedimiento ahora si con el valor que se necesita escribir.

3.3.5.2 Lectura de los registros internos del PIC

Análogamente a lo realizado en el direccionamiento del registro 40010, el campo Dato forzado (L) del registro holding 40011 (byte menos significativo) direcciona la opción a desplegarse en el LCD entre las cinco existentes, No Operación, T1CON, TMR1, T2CON y PR2 tal como se muestra en la tabla 3.4.

Tabla 3.4. Valores que adopta el campo Dato (L) del registro 40011

Registros - PIC	Registro 40011 Dato (L)
No Operación	00000001
T1CON	00000010
TMR1	00000011
T2CON	00000100
PR2	00000101

Volviendo a la figura 3.26. La columna Leer_Regs por medio del Radio Buttons etiquetado como Leer_Regs (Lectura de Registros) permite seleccionar el registro interno del PIC a desplegarse en la pantalla de cristal líquido, utilizando para esto el registro holding 40011, el cual se enlaza al Objeto Modbus tal como se muestra en la figura 3.30.

Figura 3.29. Despliegue del registro 40009 en el LCD

Mientras la opción de despliegue sea diferente a No Operación el registro 40011 tiene preferencia sobre la variable enlazada al registro 40009 (su despliegue es mostrado en la figura 3.29), de esta forma evitamos que la información de estos dos registros se muestren alternadamente en el display de cristal líquido.

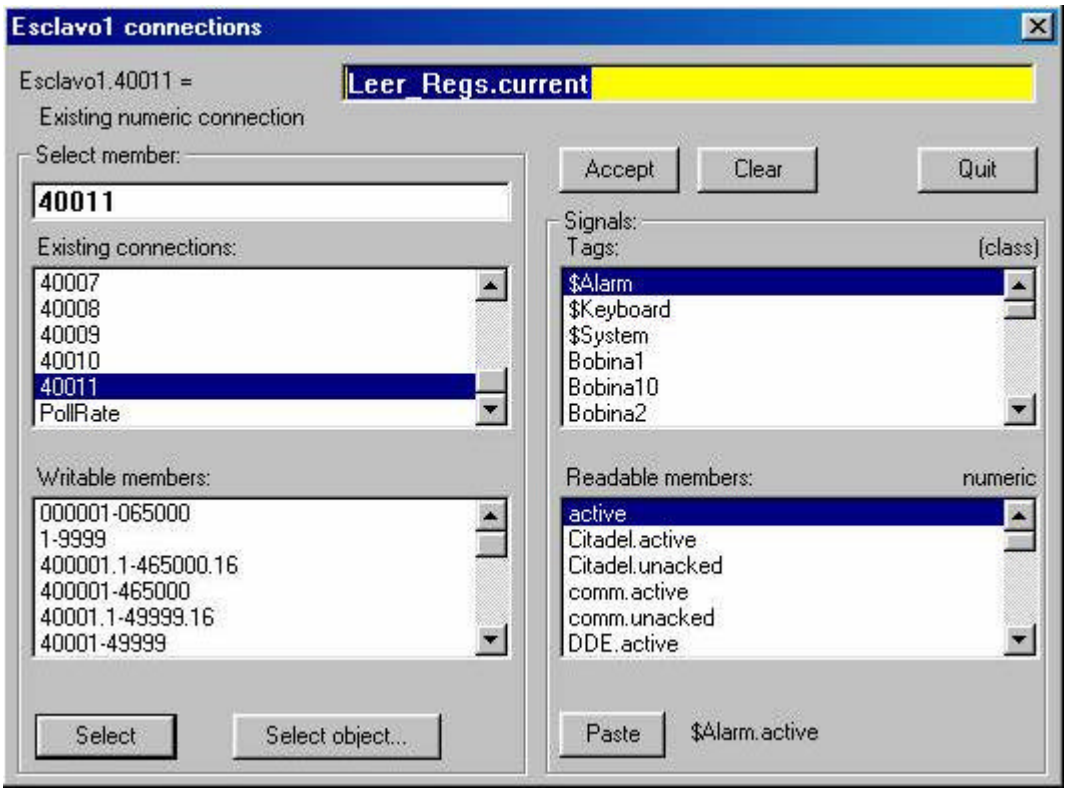

Figura 3.30. Enlace del registro holding 40011 al objeto Radio Buttons etiquetado como Leer_Regs

Para facilitar al usuario la lectura de los registros internos del microcontrolador en el LCD esta se muestra en dos formatos, hexadecimal y decimal, excepto para el par de registros concatenados TMR1H:TMR1L cuya lectura es solo hexadecimal.

La figura 3.31 muestra el despliegue del contenido de los registros internos del PIC, T1CON, TMR1H:TMR1L, T2CON y PR2 en el LCD.

(a)

(b)

(c)

(d)

Figura 3.31. Lectura de los registros internos del PIC en el LCD: (a) Registro T1CON; (b) Registros TMR1H:TMR1L; (c) Registro T2CON; (d) Registro PR2

Referencias:

1. PICBASICPRO, Reference guide.htm.

2. MICROCHIP TECHNOLOGY INC, PIC16F87X Family Reference Manual.pdf.

3. NATIONAL INSTRUMENTS, Lookout Manual.pdf.

4. Angulo Usátegui, José Ma. y Angulo Martínez, Ignacio, Microcontroladores PIC. Diseño Práctico de Aplicaciones, 2da edición, Editorial McGraw -Hill, 1999.

5. Angulo Usátegui, José Ma. y Angulo Martínez, Ignacio, Microcontroladores PIC. La solución en un chip, 4ta edición, Editorial ITP Paraninfo, 2000.

CAPÍTULO 4

IMPLEMENTACIÓN

4.1 INTRODUCCIÓN

La figura 4.1 muestra cuatro vistas de la placa microcontrolada para la cual se diseño el software, tanto el programa alojado en el microcontrolador como la interfase humano máquina realizada en Lookout.

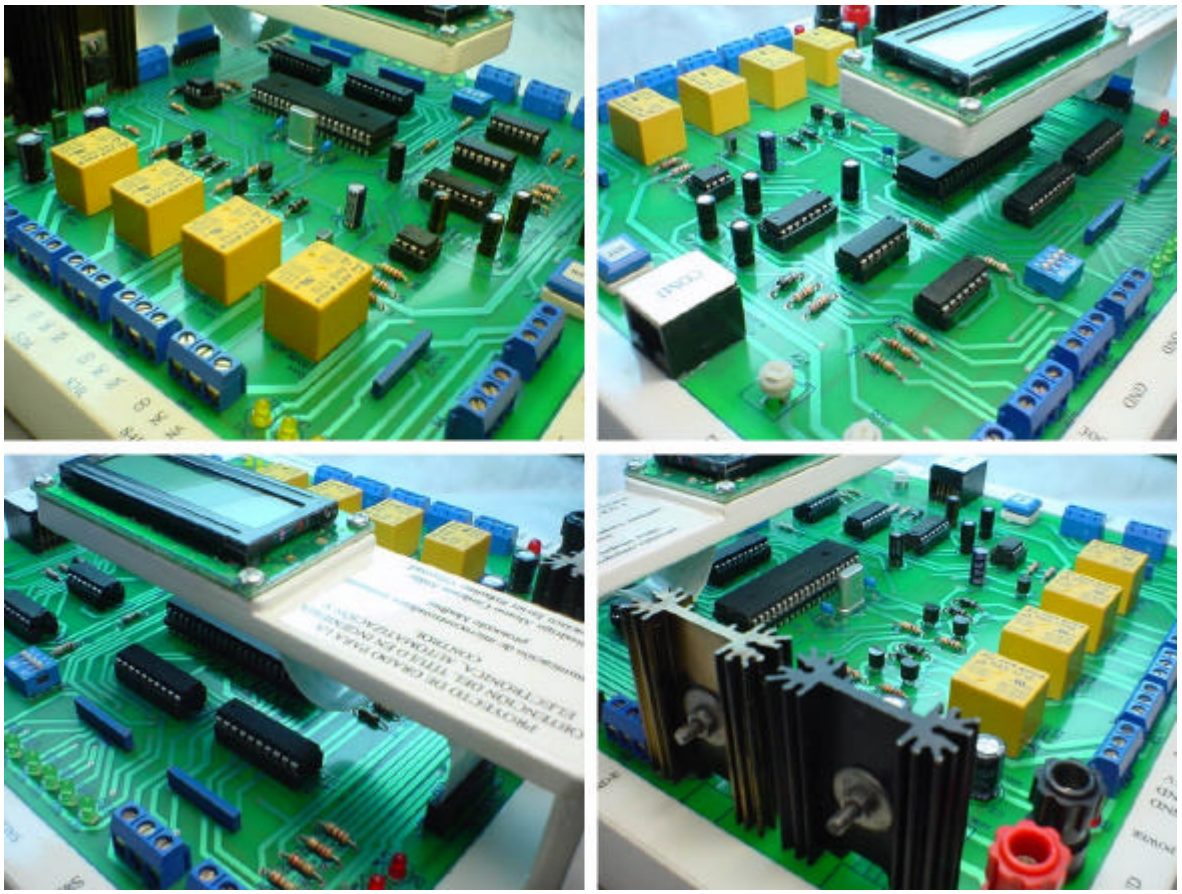

Figura 4.1. Vistas de la placa microcontrolada

La sección que corresponde a la Implementación de una Red Serial Modbus TTL, es una aplicación real realizada con un dispositivo maestro (PC), la placa microcontrolada y un segundo esclavo (los tres colocados en red), demostrando que compartir información entre nodos de red es posible. Para el segundo esclavo no se construyó placa alguna, solo se armó su circuito en un protoboard, con el único fin de probar la validez del programa de red diseñado para el microcontrolador, y la interfase humano máquina.

Se expone además, la facilidad con que el código del programa diseñado para el microcontrolador PIC16F877, es manipulado para cargarlo en un microcontrolador de menores prestaciones (PIC16F870).

4.2 IMPLEMENTACIÓN DE UNA RED SERIAL MODBUS TTL

La conexión de varios dispositivos esclavos a un nodo maestro es muy difundido en el ámbito industrial. En este apartado se implementa una red que consta de un nodo Maestro y dos nodos esclavos, cuyas señales de comunicación son de tipo TTL. Se recurre a ello no como una interfase física alternativa a una implementación de red a la cual irían conectados numerosos esclavos, sino como un medio idóneo para exponer conceptos y entenderlos mediante circuitos integrados de fácil adquisición.

4.2.1 Diseño del Hardware de la Red Serial Modbus TTL

La figura 4.2 presenta el diagrama de bloques de la Red Serial Modbus TTL propuesta para este proyecto.

4.2.1.1 Dispositivo Maestro

El dispositivo Maestro es un computador personal trabajando bajo la interfase hecha en Lookout con ayuda del integrado MAX232N. La señal del transmisor del dispositivo Maestro luego de pasar por el MAX se conecta al receptor de todos los dispositivos esclavos. Cada uno de los transmisores de los dispositivos esclavos se conectan al receptor del Maestro.

La figura 4.3 presenta el diagrama de bloques del dispositivo Maestro.

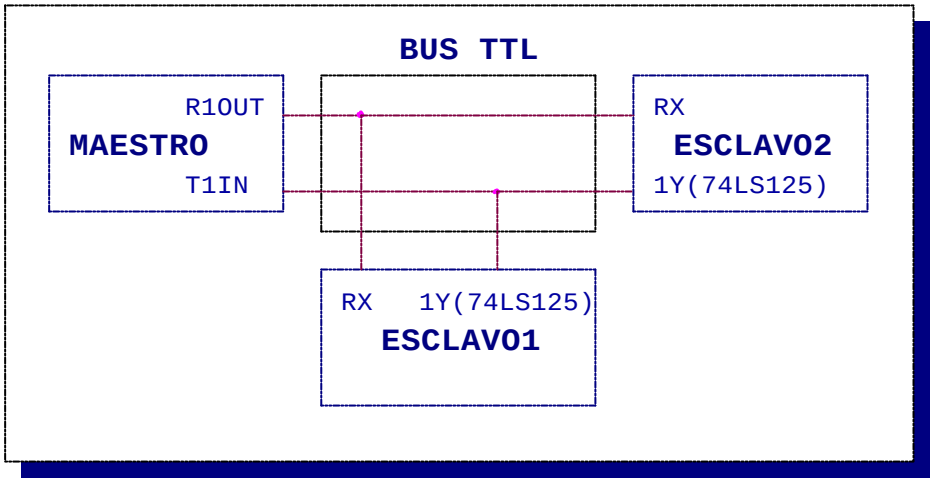

Figura 4.2. Diagrama de bloques de la Red Serial Modbus TTL

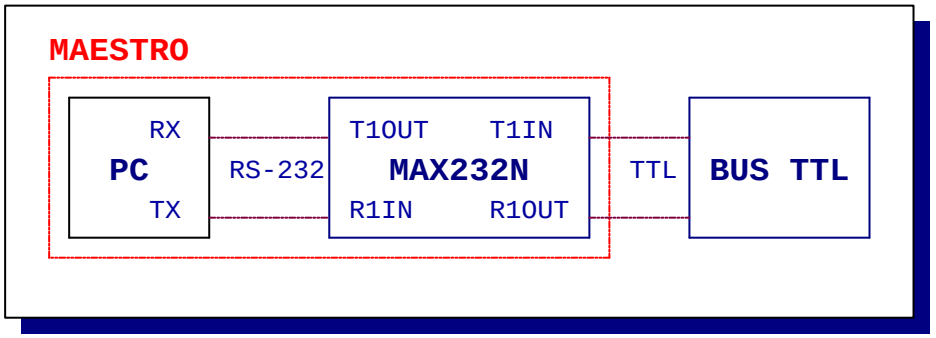

Figura 4.3. Diagrama de bloques del dispositivo Maestro

4.2.1.2 Dispositivo Esclavo1

Para evitar que un dispositivo esclavo al transmitir una trama de respuesta al dispositivo maestro, dañe la salida del transmisor de otro esclavo, se utiliza el circuito integrado 74LS125 (tal como se aprecia en la figura 4.4), cuyas salidas poseen tres estados: 1 lógico, 0 lógico y alta impedancia. La forma como cada esclavo protege la salida de su transmisor se hace mediante software, habilitando o deshabilitando al buffer correspondiente (estados lógicos o alta impedancia respectivamente) por medio de G1 (Salida 10 Digital -> S10D – ver ANEXO B). El manejo de dicha señal depende de la dirección que el dispositivo Maestro envía

en su trama de petición. De esta forma solo un esclavo tiene el control a la vez de la línea de transmisión (Driver de Red) hacia el receptor del dispositivo Maestro.

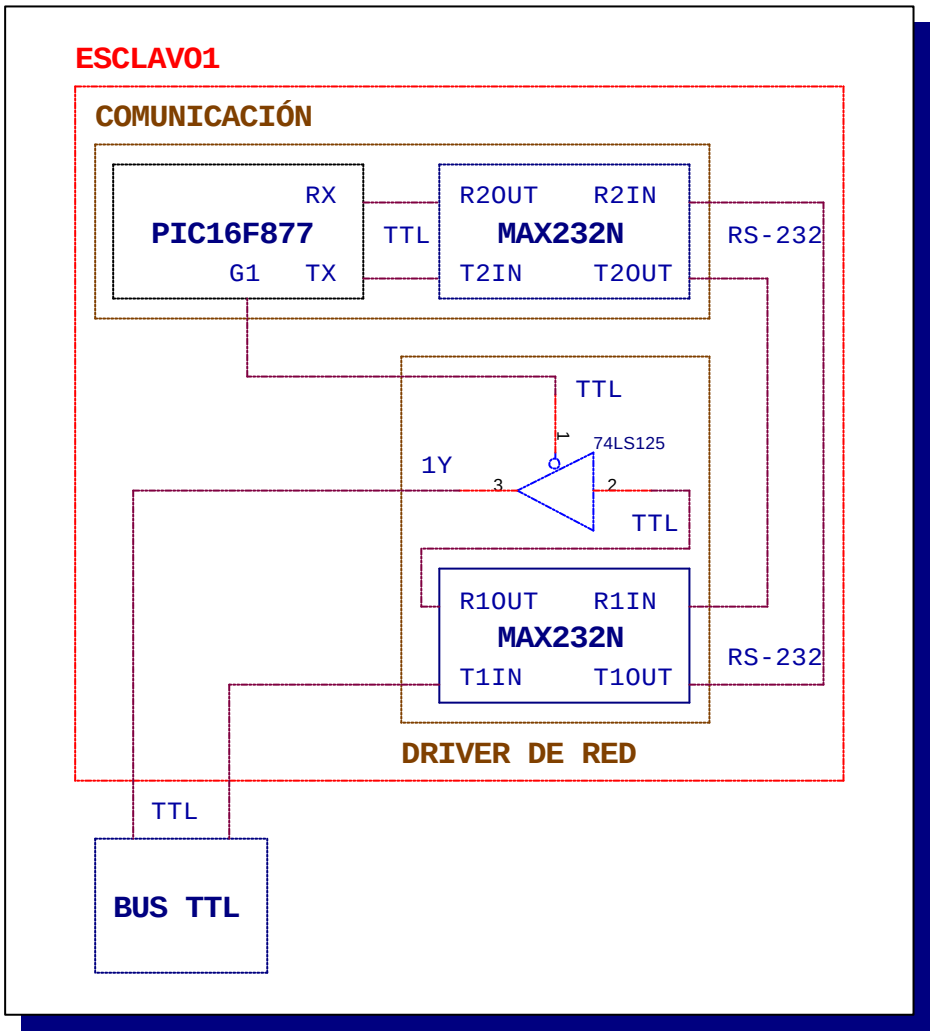

Figura 4.4. Diagrama de bloques del dispositivo Esclavo1

El bloque Comunicación, corresponde a los pines que utiliza el microcontrolador de la placa implementada para comunicarse con el exterior, en conjunción con el integrado MAX232N, y la señal con la que se accede al medio físico (G1). El bloque Driver de Red, consta del integrado 74LS244 y otro MAX232N para adaptar las señales de entrada / salida RS-232 a señales TTL y viceversa. Es necesario hacerlo ya que la placa posee un MAX para la conexión directa a una computadora.

4.2.1.3 Dispositivo Esclavo2

El Esclavo2 puede ser cualquier microcontrolador PIC cuyas señales TTL de transmisión y recepción sean conectadas como se muestra en la figura 4.5, y que además tenga cargado en el, una versión igual o de menores prestaciones (se refiere al número de periféricos que podrá manejar) al programa diseñado para el Esclavo1.

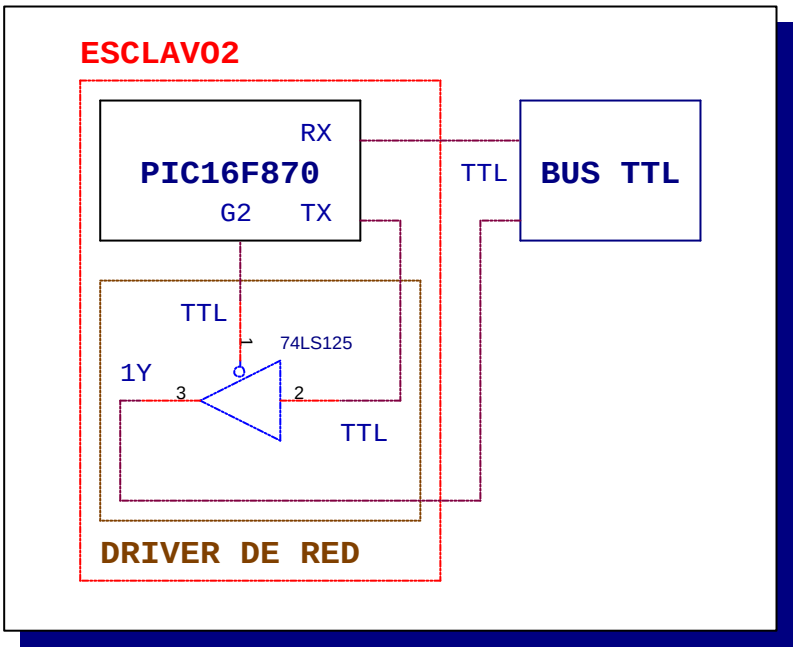

Figura 4.5. Diagrama de bloques del dispositivo Esclavo2

La señal G2 es una salida física del microcontrolador, PORTC.1.

4.2.2 Diseño del Software de la Red Serial Modbus TTL

Se modificaron los programas originalmente implementados para la placa (el realizado en Basic y Lookout) utilizando un microcontrolador más pequeño. El esclavo2 (PIC16F870) podrá manejar 3 bobinas, 1 registro holding, 2 entradas digitales y 1 registro de entrada.

Haberle asignado un manejo tan limitado obedece al único propósito de establecer conceptos de funcionamiento, a nivel de programación y de interfase en Lookout, mientras queda en el ingenio del diseñador, la puesta en marcha de

proyectos mucho más complejos, que satisfagan las necesidades que un entorno industrial requiere.

4.2.2.1 Programa Implementado en el PIC16F870

```
'*****************************************************************************************
'*  Nombre:   Modbus-Red.bas
'*  Autor:      Rodrigo Cárdenas
‘*
*   Tema:      Implementación del protocolo Modbus utilizando un microcontrolador
'*              PIC16F870
‘*  Versión:  1.0
'*****************************************************************************************

DEFINE CHAR_PACING 100  ' Paso de la salida serial en us
'*****************************************************************************************
'        DECLARACIÓN DE VARIABLES
'*****************************************************************************************
'*****************************************************************************************
'    VARIABLES Y BANDERAS DE INTERRUPCIÓN DEL USART (no reutilizables)
'*****************************************************************************************
peticion var byte [10]  ' Arreglo donde se almacena la petición del Maestro
Temp var byte  ' Backup de RCREG
Indice var byte  ' Índice en el almacenaje de datos en peticion
Bandera var byte  ' Registro en que cada bit es usado como una bandera
DP var Bandera.0  ' Bandera de detección del caracter dos puntos (:)
Unir var Bandera.1  ' Bandera que une dos nibbles en un byte

'*****************************************************************************************
'     VARIABLES Y BANDERAS DE PROPÓSITO GENERAL (no reutilizables)
'*****************************************************************************************
est var bit [3]  ' Estado de las bobinas
est1 var bit [3]  ' Estado de las entradas digitales
Ch var byte [1]  ' Canales del CAD (ADCON0)
registro var byte [1] : registro1 var byte [1]  ' Para manejo de registros internos
```

```
esclavo var byte  ' Dirección de esclavo
LSB var Bandera.2  ' Bandera que examina el LSB de "pg" para el cálculo del CRC
MTS var Bandera.6  ' Bandera que establece el Modo de transmisión serial
Driver var PORTC.1  ' Nombre dado a la salida que servirá de Driver de red (S10D)

'*****************************************************************************************************
'     VARIABLES DE PROPÓSITO GENERAL (reutilizables)
'*****************************************************************************************************
k var byte : m var byte : pg var word : pgw var word : pg1 var byte
pg2 var byte : pg3 var byte
num var byte  ' Número de elementos
pos var byte  ' Posición de un elemento
direcc var byte  ' Dirección de un elemento
ind var byte  ' Índice de arreglo

        GoTo Inicio  ' Salta alrededor del handler de interrupciones

'*****************************************************************************************************
'     INTERRUPCIÓN DEL USART
'*****************************************************************************************************
Recibir: Temp = RCREG
        If MTS = 0 Then ' Modo de Transmisión Serial RTU
          If Indice = 10 Then
            Indice = 0
          End If
          If Indice >= 2 Then
            peticion [Indice] = Temp: Indice = Indice + 1
          End If
          If Indice = 1 Then
            If (Temp>=1 And Temp<=6) Or (Temp>=$F And Temp<=$10) Then
              peticion [1] = Temp: Indice = Indice + 1
            Else
              peticion [0] = 0: Indice = 0
            End If
          End If
          If Indice = 0 Then
```

```
      If Temp = esclavo Then
         peticion [0] = Temp: Indice = Indice + 1
      Else
         Driver = 1 ' Deshabilito driver
      End If
   End If
Else  ' MTS = 1
   If DP = 1 Then ' Modo de Transmisión Serial ASCII
      ' Detección de CRLF
      If Temp>$D Then
         ' Conversión de ASCII a DEC
          Lookdown RCREG, ["0123456789ABCDEF"], Temp
      End If
      If Unir = 0 Then
         Temp1=Temp<<4 : Unir=1
      Else
         peticion [Indice] = Temp1 + Temp: Indice = Indice + 1: Unir = 0
         If Indice = 10 Then
            Indice = 0: DP = 0
         End If
         If peticion[0]!=esclavo Then
            DP = 0: Indice = 0: peticion [0] = 0
            Driver = 1
         End If
      End If
   Else
      If RCREG=$3A Then
         DP = 1: Unir = 0
      End If
   End If
End If

PIR1 0.5 = 0
Resume
Enable
```

```
On INTERRUPT GoTo Recibir

'*****************************************************************************************************
'    PUERTOS DE I/O Y REGISTRO OPTION
'*****************************************************************************************************
Inicio:  ADCON1=%00000110  ' Puerto A como I/O digitales
        TRISA=%00101111  ' Out = 0 / In = 1
        TRISB=%11000000
        TRISC=%10000000
        OPTION_REG=%10000011  ' Pull-up desactivado

'*****************************************************************************************************
'    REGISTROS DE INTERRUPCIONES
'*****************************************************************************************************
        INTCON=%11000000  ' Otros periféricos SI!!!!
        PIE1=%00100000  ' Permiso de interrupción del USART
        PIR1 0.5 = 0 ' Borra señalizador de RX  RCIF

'*****************************************************************************************************
'    DIRECCIÓN DEL ESCLAVO
'*****************************************************************************************************
        esclavo=(PORTA.1<<2+PORTA.0)+2
'*****************************************************************************************************
'    INICIALIZACIÓN DE REGISTROS PARA USART
'*****************************************************************************************************
        TXSTA=%00100100  ' 8 bits, Asincrónico, High speed
        RCSTA=%10010000  ' Habilita puerto serie y rx continua
                                ' velocidades de transmisión
        pg1=PORTA.3
        Lookup pg1, [25,12], SPBRG

'*****************************************************************************************************
'    DRIVER DE RED
'*****************************************************************************************************
        Driver = 1
```

```
'*************************************************************************************************
'    MODO DE TRANSMISIÓN SERIAL
'*************************************************************************************************
       MTS=PORTA.2  ' MTS = 0 -> RTU   MTS = 1 -> ASCII

'*************************************************************************************************
'    INICIALIZACIÓN DE REGISTROS
'*************************************************************************************************
       DP = 0: Indice = 0: PORTC 0.3 = 0: PORTC 0.4 = 0: PORTC 0.5 = 0
       GoSub Ence_P
       For k = 0 To 2
          est [k] = 0: est1 [k] = 0
       Next k

'*************************************************************************************************
'    PROGRAMA PRINCIPAL
'*************************************************************************************************
Esperar:  If MTS = 0 Then
            If peticion[1]<=6 And Indice=8 Then
              num = 5: pos = 6: GoSub Comprobar
            End If
            If peticion[1]>=$F And Indice=(peticion[6]+9) Then
              num=peticion[6]+6 : pos=num+1 : Gosub Comprobar
            End If
          Else
            If peticion[7]=$DA Then
              num = 5: pos = 6: GoSub Comprobar
            End If
            pos=peticion[6]+7
            If peticion[pos+1]=$DA Then
              num = pos - 1: GoSub Comprobar
            End If
          End If

       GoTo Esperar
```

```
'*************************************************************************************************
'        SUBRUTINAS
'*************************************************************************************************
'*************************************************************************************************
'    SUBRUTINA QUE ENCERA EL ARREGLO PETICION
'*************************************************************************************************
Ence_P:  For k = 0 To 9
           peticion [k] = 0
         Next k

         Return

'*************************************************************************************************
'    SUBRUTINA QUE REALIZA UNA SERIE DE COMPROBACIONES
'*************************************************************************************************
Comprobar: Indice = 0: DP = 0: GoSub CRCLRC: GoSub Ch_CRCLRC: GoSub
Ence_P

          Return

'*************************************************************************************************
'    SUBRUTINA QUE CALCULA EL CHEQUEO DE ERROR CÍCLICO O
‘    LONGITUDINAL
'*************************************************************************************************
CRCLRC:  If MTS = 0 Then
           pg=$FFFF
           For k = 0 To num
              pg=pg^peticion[k]
              For m = 0 To 7
                 LSB=pg&1 : pg=pg>>1
                 If LSB = 1 Then
                   pg=pg^$A001
                 End If
              Next m
           Next k
         Else
```

```
        pg1 = 0
        For k = 0 To num
           pg1=pg1+peticion[k]
        Next k
        pg2 = 256 - pg1
      End If

      Return

'*************************************************************************************************
'    SUBRUTINA QUE DICE SI EL CHEQUEO DE ERROR CÍCLICO O
‘    LONGITUDINAL ES CORRECTO
'*************************************************************************************************
Ch_CRCLRC: If MTS = 0 Then
         pgw=(peticion[pos+1]<<8)+peticion[pos]
         If pg = pgw Then
           Driver = 0: GoSub Codigo
         Else
           Driver = 1
         End If
       Else
         If pg2=peticion[pos] Then
           Driver = 0: GoSub Codigo
         Else
           Driver = 1
         End If
       End If
      Return

Codigo:

'*************************************************************************************************
'    SUBRUTINA QUE BASADA EN UNA PETICION GENERA UNA ACCIÓN Y UNA
‘    RESPUESTA
'*************************************************************************************************
```

```
'*************************************************************************************************
'     CÓDIGOS DE FUNCIÓN: LECTURA DE BOBINAS (01H)  Y
'                                    LECTURA DEL ESTADO DE ENTRADA (02H)
'*************************************************************************************************
UnoDos:  If peticion[1]<=2 Then
         ' Byte count (N) = Cantidad de salidas / 8, si el resto != 0 -> N = N + 1
         num=peticion[5]//8  ' Cantidad de salidas Lo
         If num! = 0 Then ' Si Byte count (N) es difernte de 0 entonces
           peticion[2]=(peticion[5]/8)+1  ' N = N + 1  Byte count de respuesta
         Else
           peticion[2]=(peticion[5]/8)  ' Byte count de respuesta
         End If
         ' Dirección de bobina o entradas Lo + Número de bobinas o entradas Lo
         num=peticion[3]+peticion[5]-1  ' Número de bobinas o e desde dirección Lo
         pg = 0: pg1 = 0
         If peticion[1]=1 Then  ' Función 01H
           est[0]=PORTC.3 : est[1]=PORTC.4 : est[2]=PORTC.5
           For k=peticion[3] to num
              pg=pg+Dcd(pg1)*est[k] : pg1=pg1+1
           Next k
         Else  ' Función 02H
           est1[0]=PORTB.6 : est1[1]=PORTB.7
           For k=peticion[3] to num
              pg=pg+Dcd(pg1)*est1[k] : pg1=pg1+1
           Next k
         End If
         peticion[3]=pg&$FF
         If peticion[5]<=8 Then
           peticion [4] = 0
         Else
           peticion[4]=(pg>>8)&3
         End If
         GoSub Resp
       End If
```

```
'*****************************************************************************************************
'     CÓDIGO DE FUNCIÓN: LECTURA DE REGISTROS HOLDING (03H)
'*****************************************************************************************************
Tres:    If peticion[1]=3 Then
         peticion[2]=2*peticion[5]  ' Byte count = (Número de puntos Lo)*2
         ind=4 : num=peticion[3]+peticion[5]-1
         For k=peticion[3] to num
            peticion[ind-1]=registro1[k]
            peticion[ind]=registro[k]  ' Guardo los datos bajos
            ind = ind + 2
         Next k
         GoSub Resp
       End If

'*****************************************************************************************************
'    CÓDIGO DE FUNCIÓN: LECTURA DE UN REGISTRO DE ENTRADA (04H)
'*****************************************************************************************************
Cuatro:  If peticion[1]=4 Then
         ' Asignación de canales al arreglo
         Ch[0]=%10100001  ' Canal 4 -> RA5
         peticion[2]=2*peticion[5]  ' ByteCount
         direcc=peticion[3]-5  ' Dirección baja (posibles direc 0 a 3)
         num=peticion[5]-1  ' Número de puntos Lo -> 0 a 3
         For k = 0 To num
            ADCON0=Ch[direcc+k] : ADCON1=%10000000 : PIR1.6=0  ' ADIF=0
            Pauseus 50: ADCON0 0.2 = 1 ' GO=1
ConvAD:        If PIR1.6=0 Then ConvAD
            ind=2*k+3 : peticion[ind+1]=ADRESL : peticion[ind]=ADRESH&%00000011
         Next k
         Gosub Resp : ADCON1=%00000110  ' Puerto A como I/O digitales
       End If

'*****************************************************************************************************
'    CÓDIGO DE FUNCIÓN: FORZAR UNA BOBINA SIMPLE (05H)
'*****************************************************************************************************
Cinco:   If peticion[1]=5 Then
```

```
        direcc=peticion[3]  ' Dirección de la bobina a forzarse
        If peticion[4]=$FF Then
          est [direcc] = 1
        Else
          est [direcc] = 0
        End If
        GoSub Escri_B
        If MTS = 0 Then
          pos = 7
        Else
          pos = 6
        End If
        GoSub Resp
      End If

'*****************************************************************************************************
'    CÓDIGO DE FUNCIÓN: PROGRAMAR UN REGISTRO INDIVIDUAL (06H)
'*****************************************************************************************************
Seis:   If peticion[1]=6 Then
        k=peticion[3]  ' Dirección del registro
        registro1[k]=peticion[4]  ' Data hi reg
        registro[k]=peticion[5]  ' Data lo reg
        If MTS = 0 Then
          pos = 7
        Else
          pos = 6
        End If : GoSub Resp : End If

'*****************************************************************************************************
'    CÓDIGO DE FUNCIÓN: FORZAR MULTIPLES BOBINAS (0FH)
'*****************************************************************************************************
Quince:  If peticion[1]=$F Then
        ' Dirección de bobina Lo + Número de bobinas Lo
        num=peticion[3]+peticion[5]-1  ' Número de bobinas desde dirección Lo
        If peticion[6]=1 Then  ' Examino el valor de Byte Count
          pg=peticion[7] ' Bobinas a forzarse (byte)
```

```
        Else
          pg=(peticion[8]<<8)+peticion[7] ' Bobinas a forzarse (palabra)
        End If
        For k=peticion[3] to num
          est[k]=pg//2 : pg=pg/2
        Next k
        GoSub Escri_B: GoSub Resp
      End If
      Return

'*************************************************************************************************
'    SUBRUTINA QUE IMPRIME LA RESPUESTA A UNA PETICIÓN
'*************************************************************************************************
Resp:   If peticion[1]<=4 Then
        num=peticion[2]+2
        GoSub CRCLRC: pos = num + 1
      End If
      If peticion[1]>=$F Then
        num = 5: GoSub CRCLRC
      End If
      If MTS = 0 Then
        If peticion[1]<=4 Then
          peticion[pos]=pg&$0FF  ' Guardo CRC calculado en peticion
          peticion[pos+1]=(pg&$FF00)>>8 : pos=pos+1
        End If
        If peticion[1]>=$F Then
          peticion[6]=pg&$0FF : peticion[7]=(pg&$FF00)>>8 : pos=7
        End If
        ' Imprime el cuerpo de la trama
        For k = 0 To pos
          pg1=peticion[k] : Hserout[pg1]
        Next k
      Else
        If peticion[1]<=4 Then
          peticion [pos] = pg2 ' Guardo LRC calculado en peticion
```

```
        End If
        If peticion[1]>=$F Then
          peticion [6] = pg2: pos = 6
        End If
        ' Imprime el caracter de inicio de trama (:)
        Hserout [":"]
        ' Imprime el cuerpo de la trama
        For k = 0 To pos
           pg1=(peticion[k]&$F0)>>4 : Gosub DAsccih : Hserout [pg2]
           pg1=peticion[k]&$F : Gosub DAsccih : Hserout [pg2]
        Next k
        ' Imprime los caracteres de fin de trama (CR y LF)
         Hserout [13,10]
      End If : Return

'*****************************************************************************************************
'    SUBRUTINA QUE PASA DATOS DEl ESTADO DE LAS BOBINAS
'*****************************************************************************************************
Escri_B: PORTC.3=est[0] : PORTC.4=est[1] : PORTC.5=est[2] : Return

'*****************************************************************************************************
'    SUBRUTINA QUE CONVIERTE DE DEC A ASCII HEX
'*****************************************************************************************************
DAsccih: Lookup pg1, ["0123456789ABCDEF"], pg2
          Return
'*****************************************************************************************************
'     FIN DEL PROGRAMA
'*****************************************************************************************************
```

4.2.2.2 Diseño de la Interfase Humano Máquina (HMI)

Se ha hablado que en una Red Serial Modbus los dispositivos esclavos no pueden comunicarse entre sí. Sin embargo, la flexibilidad de Lookout hace que se pueda compartir información de un esclavo para controlar algún periférico de otro dispositivo y viceversa.

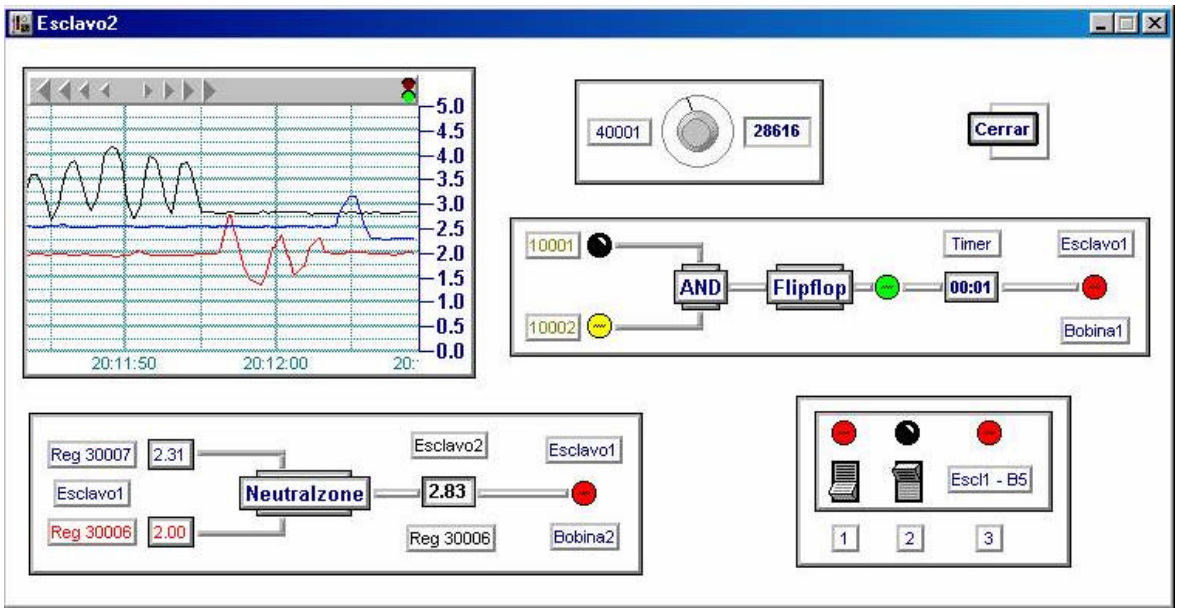

Figura 4.6. Interfase para la Red Serial Modbus

La figura 4.6 presenta la interfase completa en Lookout que se construyó para el monitoreo de la Red Serial Modbus TTL.

4.2.2.2.1 Objeto Flipflop

Flipflop cambia su señal lógica de salida de encendido a apagado, o de apagado a encendido cuando la señal “Input” es alta. La señal de salida no cambia cuando esta es baja. “Input” es un a expresión lógica. La figura 4.7 muestra como está llenado el objeto Flipflop en la interfase.

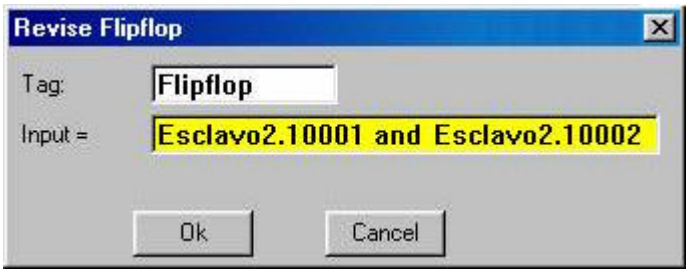

Figura 4.7. Objeto Flipflop

4.2.2.2.2 Objeto Pulse

Pulse es un timer que genera un pulso periódico de una duración específica. Cuando la transición de “On/off signal” es ON, la señal de salida cambia a alto por el tiempo de duración del pulso y regresa a cero por lo que resta del periodo.

La señal de salida inmediatamente se apaga cuando “On/off signal” se pone en bajo. “Timer period” es el intervalo de tiempo para el ciclo del pulso completo, y “Timer duration” es el ancho de cada pulso.

Estos parámetros pueden variar desde 0.0 segundos hasta varios años, con una resolución efectiva de 0.1 segundos sobre el rango completo. “Timer duration” debe ser siempre menor que “Timer period”.

Esto es descrito en el formato definido por el parámetro “Display format” y actualizado aproximadamente una vez por segundo. “On/off signal” es una expresión lógica mientras que “Timer period” y “Timer duration” son expresiones numéricas. La figura 4.8 muestra como está llenado el objeto Pulse en la interfase.

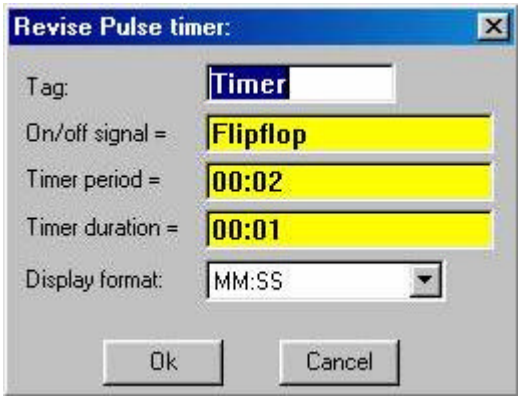

Figura 4.8. Objeto Pulse

4.2.2.2.3 Descripción

El propósito de esta interfase (la que se muestra en la figura 4.9) es encender y apagar continuamente algún elemento del Esclavo1 utilizando para esto señales provenientes del Esclavo2. Para ello se usa el Objeto Flipflop. Cuando las

entradas digitales 10001 y 10002 del Esclavo2 se activan al mismo tiempo, este objeto queda encendido o apagado de acuerdo a su condición previa.

Se enlaza este objeto etiquetado como “Flipflop” colocándolo en el campo “On/off signal” del objeto “Pulse”, de esta forma cuando el objeto “Flipflop” se encienda el objeto “Pulse” también lo hará. El último paso es conectar el objeto “Pulse” a alguna salida que se quiera controlar. En este caso se utiliza la Bobina1 del Esclavo1.

Esta salida a relé se encenderá por un segundo y se apagará por otro intervalo de igual duración, correspondiendo a los valores cargados en los campos “Timer period” y “Timer duration” del Objeto Pulse.

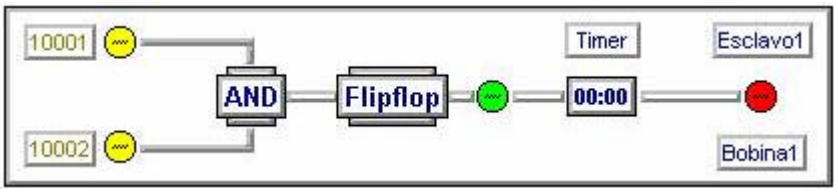

Figura 4.9. Manejo de los objetos Flipflop y Pulse

4.2.2.2.4 Objeto Neutralzone

Neutralzone es un controlador ON/OFF. Cuando el valor en “Signal” crece por encima de “Low limit” y “High limit”, el dato miembro “above” se enciende mientras que el dato miembro “below” se apaga. Cuando el valor en “Signal” decrece por debajo de “Low limit” y “High limit”, “above” se apaga y “below” se enciende. “Signal”, “High limit” y “Low limit” son expresiones numéricas. La figura 4.10 muestra como está llenado el objeto Neuralzone en la interfase.

4.2.2.2.5 Descripción

En esta interfase (la que se muestra en la figura 4.11) el campo “Signal” del objeto Neutralzone se rellena con el registro de entrada del Esclavo2 (30006) el cual se comparará con los límites alto y bajo de este objeto, que no son más que

los registros de entrada 30006 y 30007 del Esclavo1. El resultado activará o desactivará a la Bobina2 del Esclavo1.

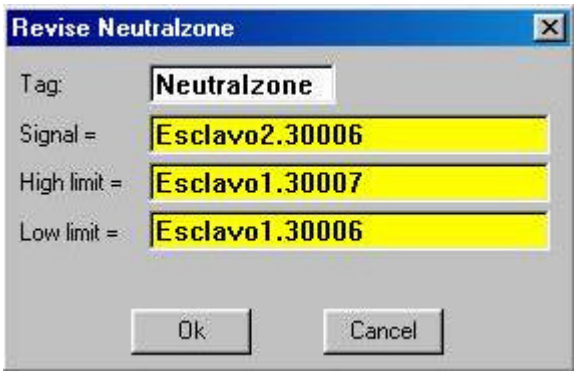

Figura 4.10. Objeto Neutralzone

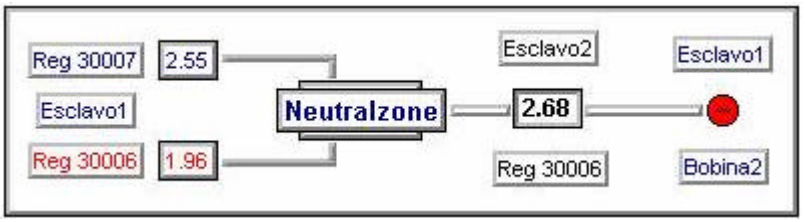

Figura 4.11. Manejo del Objeto Neutralzone

CAPÍTULO 5

PRUEBAS Y RESULTADOS

5.1 INTRODUCCIÓN

Cabe considerar al microcontrolador PIC16F877 como el cerebro del sistema implementado, el cual gestionará convenientemente tanto las tramas enviadas por un dispositivo maestro, como las tramas de respuesta necesarias que proporcionarán información a dicho dispositivo (maestro), sobre los diversos periféricos conectados al el (esclavo). Por tal motivo, se realizaron pruebas al software del sistema, con el objetivo de proveer al microcontrolador de garantías suficientes que satisfagan tanto trabajo óptimo como seguridad.

5.2 PRUEBAS AL DISEÑO DEL SOFTWARE DEL SISTEMA

5.2.1 Pruebas al Programa alojado en el Microcontrolador

Los algoritmos que manejan los datos entrantes por el puerto serie para cada modo de transmisión serial, demandaron muchísimas pruebas que hicieron replantar varias veces el programa como tal. Por lo tanto, fue necesario seguir un orden que permitiese conseguir pequeños resultados, que sumados lograran el fin requerido. Como proyección práctica a tal idea, se utilizó el siguiente método:

1. Desarrollar el código de una función específica

2. Probar su correcto funcionamiento por medio de una interfase en Lookout creada a la medida, o en la visualización de datos por medio del LCD (como forma de depurar el programa)

3. Resumir el código alojado en el microcontrolador basado en los resultados obtenidos, regresando al paso 2 hasta que el paso 3 no sea admisible.

Además, se debe considerar el manejo adecuado entre variables de propósito general, de interrupción y de subrutinas, para evitar sobrescribir variables de importancia con datos temporales. Otro punto a tomarse en cuenta, fue la reducción del código implementado para cada función Modbus, basado en las características físicas de la placa (máximo número de entradas, salidas y registros). Rigiéndose a lo explicado con anterioridad, se logró que el código final implementado posea 1200 palabras menos aproximadamente, al obtenido en un principio.

5.2.2 Pruebas a la Interfase Humano Máquina (HMI)

Al inicio del diseño de la HMI se empleó Lookout 4.5, ya que es la última de las versiones que desarrolló la National Instruments. Sin embargo, produjo algunos problemas de funcionamiento al sistema microcontrolado, entre ellos se puede mencionar, el encendido y apagado de bobinas, sin previa orden.

Las pruebas a la que fue sometida la interfase basada en la versión mencionada no proporcionaron resultados positivos, al contrario, nunca pudo establecerse el motivo real de este malfuncionamiento, llegando incluso a pensarse que el programa alojado en el microcontrolador estaba mal diseñado. La utilización de Lookout 3.8 demostró lo contrario, siendo la versión de Lookout utilizada para el diseño completo de la HMI.

5.3 RESULTADOS DE LA GESTIÓN DE TRAMAS

En la figura 5.1 aparece el cuadro de dialogo en el cual se observa el manejo estadístico del Protocolo Modbus. En este cuadro Lookout valida el conteo de tramas de respuesta transmitidas por uno o varios dispositivos (esclavos) en una transacción Modbus, desde la última vez que el botón de RESET se presionó.

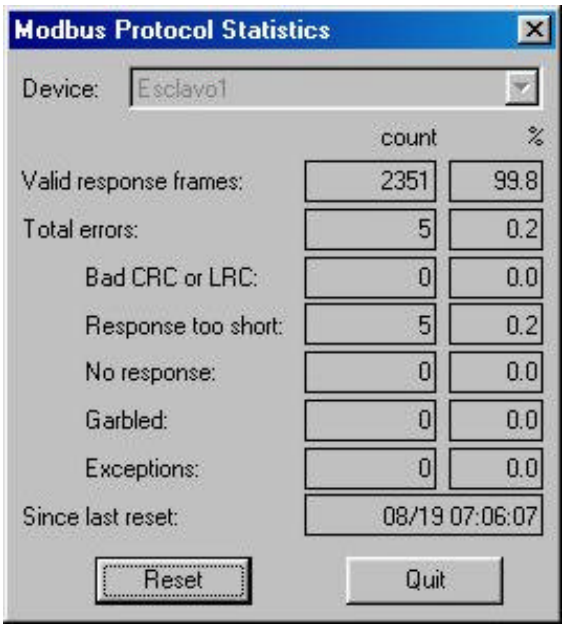

Figura 5.1 Cuadro de diálogo de Estadísticas del Protocolo Modbus

Los errores que puedan producirse se desplegarán en forma porcentual, tales como un mal CRC o LRC, una respuesta demasiado corta, una trama sin respuesta o incomprensible.

El porcentaje de tramas validas transmitidas por parte de la placa microcontrolada hacia el dispositivo maestro, durante varias pruebas a la que fue sometida, osciló entre un 98,0% a un 99,9%. Lo cual hace que el software de este sistema microprocesado sea bastante confiable.

Para acceder a esta valiosa información que Lookout brinda, escoger en Modo de Edición lo siguiente: Opciones, Modbus y Statistics.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- El propósito del proyecto fue obtener el máximo provecho del microcontrolador (entradas y salidas), tratando de conseguir en el proceso de diseño un sistema que se asemejara al funcionamiento de un PLC en sus aspectos más básicos, así como también la utilización de algunas de las características propias de los periféricos del microcontrolador, como son: PWM, timers y conversores análogo digitales.

- El número máximo de elementos del arreglo petición (arreglo encargado de almacenar la pregunta del maestro) no solo depende de la cantidad de registros holding que el esclavo podrá gestionar dentro del código del programa, sino también del número máximo de elementos que el compilador PICBasic Pro permite declarar dentro de un arreglo. Estamos hablando de 64 elementos si el arreglo en mención es de tipo byte.

- Son pocos los registros internos del microcontrolador (T1CON, TMR1H:TMR1L, T2CON y PR2) que pueden manipularse mediante la interfase hecha en Lookout. Un proyecto que requiera controlar un número más grande de registros internos encontrará en el código implementado para el PIC16F877 (hablando de la escritura de registros holding) las herramientas necesarias para lograrlo. De esta forma no se limita a controlar solamente los conversores análogo digitales, timers y PWM, sino todo lo que el PIC puede ofrecernos, como son por ejemplo los registros asociados a la Puerta Esclava Paralela e I2C, lográndolo por medio de simples cambios en el código del programa del microcontrolador e interfase

mencionada, y que solo se limitarían al espacio que el microcontrolador pueda almacenar dependiendo de la aplicación a realizarse.

- La visualización del contenido de los registros TMR1H:TMR1L a través de la interfase construida en Lookout, podría verse afectada si la frecuencia externa aplicada a T1CKI (reloj externo) o a T1CKI y T1OSI (cristal) es mayor a la frecuencia con la que Lookout actualiza la lectura de los registros TMR1H:TMR1L. Esta frecuencia de actualización depende exclusivamente del número de esclavos que se estén controlando al mismo tiempo y de los códigos de función utilizados para cada uno de ellos. Entre mayor sea el número de operaciones que realice el maestro, la actualización de la lectura de registros y en si del sistema en general será más lenta.

- El código implementado en el PIC16F877 es bastante flexible, tanto así que al reducir las características de los periféricos a controlar, se pudo demostrar que podía caber en un PIC16F870, y si se reduce aún más podría utilizarse un PIC16F84.

- En la memoria EEPROM del PIC16F877 no solamente se guardaron los valores de la Dirección del Esclavo, el Modo de Transmisión Serial, el registro asociado (SPBRG) a la Velocidad de Transmisión, la Habilitación del Driver de Red, T1CON, T2CON y PR2, sino también mensajes de despliegue para lectura de los registros internos y carátula. La memoria de datos de la EEPROM es de 256 bytes, se utilizaron 187 bytes, lo que quiere decir que se ocupó un total de 73.04% de esta memoria no volátil.

- Al compilar por medio de PICBasic Pro el programa realizado para el PIC16F877 (Modbus.bas), el resultado fue de 5934 palabras, pudiendo almacenar en este microcontrolador hasta 8192 palabras, por lo tanto el uso del mismo fue del 72.44%. Para el programa realizado en el PIC16F870 (Modbus - Red.bas), el resultado fue de 1648 palabras, pudiendo almacenar en este microcontrolador hasta 2048 palabras, por lo tanto el uso del mismo fue del 80.47%. No se utilizó memoria EEPROM para almacenar datos de interés.

- Se pudo comprobar que un mal funcionamiento de alguno de los dispositivos esclavos de nuestra Red Serial Modbus TTL, perjudicaría significativamente a toda la red, siendo la interfase hecha en Lookout la que de la pauta de la clase de errores que se estarían sucediendo en ese instante.

6.2 RECOMENDACIONES

- La capacidad de manejar una interfase RS-485 y la de poder grabar el microcontrolador directamente en la placa, serían dos opciones muy recomendadas para mejorar el diseño propuesto. Las consideraciones solamente quedan a criterio del diseñador.

- Se recomienda no hacer un uso extensivo del regulador a 5V integrado en la placa y cuya disponibilidad se encuentra en un borne etiquetado con el mismo nombre. El integrado LM7805C provee la alimentación de todo el circuito y su uso debe ser limitado a pequeñas aplicaciones externas y nunca utilizarse por ejemplo como fuente de alimentación a la Red Serial Modbus TTL propuesta en este trabajo. En tal caso lo mejor sería utilizar una fuente externa.

- Se recomienda que el código de programación utilizado en el microcontrolador para algunos de los códigos de función implementados sea revisado, si el número de entradas y salidas en un futuro proyecto fuera mucho mayor al planteado en este trabajo. Dicho código de programación respalda solamente las características físicas de esta placa, con el fin de reducir su extensión en programa.

- Si el usuario llegado el caso necesitara mayor información sobre el estado del microcontrolador o su entorno, sería recomendable crear una nueva interfase (ya que Lookout no lo permite) y código para el mismo, que a parte de las funciones implementadas en este proyecto, pueda gestionar funciones de diagnóstico. El único fin sería obtener un mayor control por parte del usuario.

- Cuando se quiere obtener el voltaje de alimentación para el integrado LM324 utilizando la interfase hecha en Lookout, es recomendable que el voltaje que entra al microcontrolador no supere el máximo absoluto que este puede

soportar, el daño no solo puede afectar al canal de conversión análogo digital sino a todo el microcontrolador. Como soporte secundario se recomienda utilizar un multímetro para facilitar tal procedimiento.

- Emplear un Modo de Transmisión Serial en todos los dispositivos esclavos conectados a una Red Serial Modbus, es lo que se recomienda para evitar posibles fallos de comunicación.

- Si se desea implementar una Red Serial Modbus TTL como la que se propuso en este proyecto de tesis, se recomienda que los microcontroladores a utilizar posean hardware USART. Las pruebas realizadas demostraron (utilizando el PIC16F84) que era casi imposible establecer una comunicación ya que los datos se corrompían debido a la longitud del bus.

- Si se decide implementar una Red Serial Modbus TTL, se recomienda que cada dispositivo esclavo se pruebe independiente de los otros, para realizarlo se deberán deshabilitar físicamente los drivers de red (alta impedancia) de los dispositivos esclavos que no estén bajo prueba.

- Si se construye una Red Serial Modbus basada en la utilización de microcontroladores como nodos de red, sería altamente recomendable (si estos se encuentran a buena distancia el uno del otro) diseñar el hardware y software necesarios que permitan programar cualquier dispositivo esclavo en forma remota, evitando trasladarse físicamente para realizar tal procedimiento.

- Un error de comunicación en una transacción Modbus maestro – esclavo, puede deberse a varias circunstancias. Suponiendo que el código implementado es correcto, las recomendaciones al respecto son las siguientes; cerciorarse que todos los integrados de la placa están bien sujetos a sus respectivos zócalos, que la dirección del esclavo que lo identifica y la velocidad de transmisión, corresponden al objeto Modbus utilizado en la interfase hecha en Lookout, que la placa no se encuentre dentro de la configuración de los parámetros de comunicación, que el flanco de la habilitación del Driver de Red (en caso de estar en una) corresponda al utilizado por el integrado que cumple dicha función, y que la integridad del bus sea la requerida.

ÍNDICE

ÍNDICE DE CAPÍTULOS

ÍNDICE DE FIGURAS Y TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 5

ÍNDICE DE TABLAS

CAPÍTULO 2

CAPÍTULO 3