UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

“Sistema de lectura de medidores de energía eléctrica basado en Protocolos de Internet”

PRESENTADO POR:
DAVID ARTURO CAMPOS PÉREZ
RONALD EDWIN DE LA CRUZ CHINCHILLA

PARA OPTAR AL TÍTULO DE:
INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, OCTUBRE DE 2006

UNIVERSIDAD DE EL SALVADOR

RECTORA :

DRA. MARÍA ISABEL RODRÍGUEZ

SECRETARIA GENERAL :

LICDA. ALICIA MARGARITA RIVAS DE RECINOS

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIO :

ING. OSCAR EDUARDO MARROQUÍN HERNÁNDEZ

ESCUELA DE INGENIERÍA ELECTRICA

DIRECTOR :

ING. LUIS ROBERTO CHÉVEZ PAZ

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

“Sistema de lectura de medidores de energía eléctrica basado en Protocolos de Internet”

Presentado por :

DAVID ARTURO CAMPOS PÉREZ
RONALD EDWIN DE LA CRUZ CHINCHILLA

Trabajo de Graduación aprobado por :

Docente Director :

ING. LUIS ESCOBAR BRIZUELA

San Salvador, Octubre de 2006

Trabajo de Graduación Aprobado por:

Docente Director :

ING. LUIS ESCOBAR BRIZUELA

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, 18 de octubre de 2006, en el Centro de Cómputo de esta Escuela, a las diecisiete horas con treinta minutos , en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

ESCUELA DE INGENIERIA ELECTRICA
FACULTAD DE INGENIERIA
Y ARQUITECTURA
Universidad de El Salvador

Firma:

1. **Ing. Luis Roberto Chévez Paz**
   Director

2. **Ing. Gerardo Marvin Hernández**
   Secretario

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

Firma:

1. **Ing. Marvin Mauricio Flores García**

2. **Ing. José Roberto Ramos**

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

"Sistema de lectura de medidores de energía eléctrica basado en Protocolos de Internet"

A cargo de los Bachilleres:

CAMPOS PEREZ, DAVID ARTURO
DE LA CRUZ CHINCHILLA, RONALD EDWIN

Habiendo obtenido el presente Trabajo una nota final, global de: 8.4

(Ocho punto Cuatro)

TABLA DE CONTENIDOS

ANEXOS

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCIÓN

En los últimos años ha aumentado la concientización por parte de las empresas e industrias en cuanto al ahorro en el consumo de energía eléctrica, y la difícil producción de energía de manera constante para satisfacer sus requerimientos. Cada día las empresas toman medidas de seguridad para el ahorro en cuanto al consumo de energía eléctrica también debido a su costo, ya que cada vez se hace más elevada su producción y latente escasez.

Para el sector industrial es básico y fundamental el estar al pendiente del consumo de energía eléctrica, así como de sus parámetros eléctricos, para mejorar la eficiencia en su producción. Ya no basta con poner a un personal que este al pendiente del consumo de energía y que este tomando mediciones de manera constante, sino el tener un equipo electrónico de medición las 24 horas monitoreando remotamente los parámetros eléctricos.

Con esta creciente necesidad, las empresas dedicadas a la fabricación de equipo electrónico destinado a la industria, han enfocado sus esfuerzos al diseño de equipos que ayuden al ahorro o consumo eficiente de energía eléctrica, a través del monitoreo de parámetros eléctricos existentes en sus líneas de suministro de energía y poder eliminar los procesos que producen una mala calidad de la potencia eléctrica.

Para realizar dicho monitoreo se necesita un medio[1] y protocolo de comunicación[2]; sin embargo en el área de las comunicaciones en entornos industriales, la estandarización de protocolos es un tema en permanente discusión, donde intervienen problemas técnicos y comerciales. Cada protocolo está optimizado para diferentes niveles de automatización y en consecuencia responden al interés de diferentes sistemas.
Debido a la no aceptación de un protocolo estándar único en las comunicaciones Industriales, los múltiples buses de campo[3] han perdido terreno ante la incursión de tecnologías de comunicación emergentes como Ethernet[4] en esta área. La aceptación mundial de Ethernet en los entornos administrativos y de oficina ha generado el deseo de expandir su aplicación a la planta. Ethernet se está

[1] Se refiere al medio físico para el intercambio de información. Ej. Cable UTP, línea serial, línea telefónica, ondas de radio, etc.
[2] El protocolo de comunicación define los aspectos técnicos involucrados en la comunicación, tales como tramas, códigos, lógica de operación, etc.
[3] Los buses de campo son una forma especial de LAN dedicada a aplicaciones de adquisición de datos y comando de elementos finales de control sobre la planta. Los buses de campo típicamente operan sobre cables de par trenzado de bajo costo
[4] Medio de transporte físico usando cable UTP y conectores RJ45. Regido por la norma IEEE 802.3 [2]

moviendo rápidamente hacia el mercado de los sistemas de control de procesos y la automatización, de esta forma reemplazando a los buses de campo en las industrias.

En las aplicaciones industriales, Ethernet es usado en conjunto con la pila de protocolos TCP/IP[5] universalmente aceptada. TCP/IP es el conjunto de protocolos usado en Internet[6], proveyendo un mecanismo de transporte de datos confiable entre máquinas y permitiendo interoperabilidad entre diversas plataformas. Usar TCP/IP sobre Ethernet a nivel de campo en la industria permite tener una verdadera integración con Intranet corporativa, y de esta forma se ejerce un estricto control sobre el equipo monitoreado.

En el presente trabajo de graduación se pretende utilizar un estándar de comunicación sobre Ethernet, Modbus/TCP[7] , para implementar el monitoreo de un *Medidor de Energía Trifásico*[8] capaz de ser accesado a través de la red LAN, usando los protocolos TCP/IP. El protocolo Modbus/TCP es muy difundido por ser abierto, lo cual le permite la fácil integración al software y dispositivos existentes en el mercado.

El medidor de energía actualmente se comunica vía el puerto RS-232 directamente a la computadora. Para no realizar cambios sobre su diseño actual (Hardware) se desarrollara una tarjeta que se conectara al puerto RS-232 del medidor y poseerá un conector RJ-45 para su correspondiente conexión a la red Ethernet.

El trabajo de graduación se enfoca sobre el diseño e implementación de dicha tarjeta, lo cual involucra un amplio conocimiento de protocolos ubicados en diferentes capas del modelo TCP/IP[9]; como Ethernet, IP, TCP, Modbus, etc.

El protocolo MODBUS/TCP es un protocolo de distribución totalmente gratuito, y su aceptación esta creciendo rápidamente en diferentes entornos, especialmente sobre redes Ethernet. La utilización de protocolos estándar fomenta la fácil interconexión entre nuestros dispositivos con los equipos y software del mundo, ya que todos “hablan el mismo idioma”.

---

[5] TCP *Transport Control Protocol* e IP *Internet Protocol*

[6] Internet es una red de redes vinculadas por gateways, que son computadoras que pueden traducir entre formatos incompatibles.

[7] MODBUS es un protocolo de mensajerilla utilizado para la comunicación entre un servidor y un cliente, estableciendo comunicación remota, utilizando el protocolo TCP/IP sobre una red Ethernet.

[8] Medidor creado en la Escuela de Ingeniería Eléctrica [1]

[9] TCP/IP es un protocolo que define 4 capas en arquitectura de implementación .

CAPÍTULO I

GENERALIDADES

1.0 Descripción del Tema

Con el presente Trabajo de Graduación, se desea auxiliar en los problemas presentados en el sector industrial a la hora de realizar interconexión de equipos de medición comerciales usados para el monitoreo del consumo de energía eléctrica y análisis de parámetros eléctricos. Se enfocara en la implementación de un dispositivo electrónico que permita el monitoreo remoto de un medidor de energía utilizando una red LAN[10].

El medidor de energía sobre el cual se trabajará fue desarrollado recientemente en un trabajo de graduación en esta escuela, denominado "Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754".
Tal medidor de energía actualmente posee comunicación serial vía puerto RS232[11], por lo cual, el equipo a desarrollar básicamente será una tarjeta con puerto RS232 para la comunicación con el medidor de energía y jack[12] RJ45 para su respectiva conexión a la red Ethernet. Dicha tarjeta debe utilizar un protocolo de comunicación estándar que cumpla con normas internacionales y de amplio uso por otros medidores comerciales para una fácil integración.

De esta forma la información recopilada por el medidor debe ser accesible en cualquier estación de trabajo que esté conectada a la red LAN (Ethernet).

Además, se implementara una interfase en un lenguaje de programación multiplataforma de alto nivel, para el estudio y recopilación de la información proporcionada por los equipos de medición de energía eléctrica integrados en la red de comunicación, de manera remota, desarrollando de esta forma un sistema que brinda una solución práctica en el sector industrial y comercial de nuestro país.

[10] LAN (Local Area Network) es una red de area local.
[11] Puerto de comunicación serial de 9 pines
[12] Puerto para el conector RJ45 (usado para cables UTP)

1.1 Objetivos

1.1.1 Objetivo General

Implementar una tarjeta que permita el monitoreo remonto del medidor de energía diseñado en la EIE, permitiéndole ser conectado a una red Ethernet local[13] como cualquier otro dispositivo de red.

1.1.2 Objetivos Específicos

Brindar al medidor de energía de un hardware (tarjeta Ethernet-RS232), que le permita pasar del medio serial a TCP/IP sobre Ethernet.

Desarrollar un entorno visual para el respectivo monitoreo del medidor de energía y sus parámetros principales.

Utilizar protocolos de comunicación de reconocimiento internacional y que permita una fácil integración con otros medidores comerciales en el mundo.

Desarrollar un sistema Hardware-Software económico, permitiéndole al medidor ser competitivo en el mercado.

Profundizar en protocolos estándar como Ethernet, TCP/IP, MODBUS/TCP, MODBUS SERIAL, etc. dejando una documentación completa que sirva de apoyo para futuras investigaciones.

Seleccionar un protocolo de comunicación estándar que sea útil a los futuros proyectos basados en la redes en escuela de Ingeniería Eléctrica.

1.2 Justificación

El presente trabajo de graduación surge, de los problemas presentados en aplicaciones industriales debido a la interconexión y accesibilidad de los medidores de potencia, ya que cada día se hace mas estricto el monitoreo del consumo de energía eléctrica, así como el análisis de sus parámetros eléctricos. La lectura de medidores de energía eléctrica a través de la red Internet es una necesidad para la industria eléctrica. En los próximos años se desarrollará gran actividad en este sector. La naturaleza del diseño involucra aspectos de

[13] Una Ethernet local puede ser un cable UTP perteneciente a la LAN *Local Area Networt* mas próxima.

comunicaciones, hardware de instrumentación y legislación del mercado eléctrico.

Uno de los principales problemas de este tipo de sistema de monitoreo es su costo, por lo que con este trabajo de graduación se busca permitirle al medidor de energía mantener su bajo costo e incrementar sus características para brindarle mayor competitividad.

El utilizar protocolos de comunicación propietarios ya no es una opción viable puesto que las empresas que desarrollan medidores están adoptando protocolos estándar, los cuales garantizan una fácil integración entre ellos y es una característica de gran interés para los clientes. El formato estándar (protocolo) de comunicación utilizado en el presente trabajo de graduación será adoptado en todos los proyectos de instrumentos basados en la red, que serán desarrollados en la Escuela de Ingeniería Eléctrica.

1.3 Alcances

Desarrollar la instrumentación necesaria para dotar al medidor de energía de un medio para conectarse a una red local Ethernet, y ser monitoreado remotamente desde cualquier computadora con acceso a la red LAN.

Implementar un protocolo de comunicación estándar utilizado ampliamente en medidores en el mundo, para una fácil integración a dispositivos que “hablen” el mismo protocolo.

Implementar las funciones básicas necesarias del protocolo en cuestión, para garantizar el monitoreo remoto de los parámetros básicos del medidor de energía.

Desarrollo de una aplicación grafica básica que sirva de interfaz al usuario para el monitoreo de los valores adquiridos por el medidor de energía.

1.4 Limitaciones

Se tienen limitaciones económicas, por lo que el desarrollo del software se debe realizar basándose en especificaciones técnicas y código de libre distribución para cada protocolo, que servirá de base para cada etapa de implementación.
El tiempo también limita la implementación de cada protocolo a sus requerimientos básicos necesarios, ya que de lo contrario involucraría un crecimiento significativo en la complejidad del trabajo de graduación.

CAPÍTULO II

TESIS PREVIA

Introducción

En la actualidad existen diversos productos de monitoreo de parámetros de la energía eléctrica, de varias marcas reconocidas mundialmente, como lo son: General Electric, Siemens, Schlumgerber, entre otras. Los parámetros de principal interés en la industria y monitoreados por éstos equipos son básicamente: Potencias, Voltajes, Corrientes, Factor de Potencia, Armónicos, Transitorios, entre otros. Cada uno de estos parámetros tiene su importancia, y un principal interés dependiendo del proceso de producción de cada empresa y equipo de producción utilizado.

Con el creciente desarrollo de la electrónica, la comunicación y el software por computadores, los medidores de estado sólido monitoreados remotamente a través de la red LAN, se ha vuelto una opción altamente rentable para las empresas, permitiéndoles un control en tiempo real de los parámetros proporcionados por la red de medidores de la empresa, y reduciendo los costos del personal a cargo de recolectar estos datos.

La ventaja principal del monitoreo de estos parámetros es básicamente fundamentar la toma de decisiones sobre el uso debido y planeación del consumo de la energía para cada sección de interés, así como la prevención de fallas en los equipos utilizados que se encuentren conectados a la misma línea de alimentación.

Los precios de estos equipos van desde los $800.00 USD hasta aproximadamente $20,000.00 USD, dependiendo de la marca, características y aplicaciones. La tendencia de dichas características es, no solamente el monitoreo de parámetros, sino la transmisión de los resultados del monitoreo a otros medios de concentración de información, a través de diversos protocolos de comunicación, tanto estándares como propietarios del fabricante.

Este avance tecnológico ha empezado a desarrollarse en la Escuela de Ingeniería Eléctrica, siendo uno de los primeros pasos el desarrollo de un

medidor de energía trifásico de estado sólido, con comunicación serial utilizando el puerto RS232. Dicho medidor captura los parámetros eléctricos básicos al igual que otros medidores en el mercado mundial y reduce su costo de producción.
Sin embargo para poder ser competitivo necesita un sistema de comunicación que permita su monitoreo remoto utilizando un medio y protocolos estándar de amplia aceptación en el mercado internacional para su fácil integración, volviéndolo competitivo.

En el presente capitulo se describe brevemente en que consiste tanto el hardware como el software del medidor sobre el cual se basa el presente trabajo de graduación. Para una mejor referencia ver el trabajo de graduación "Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754" [1].

2.0 Descripción General

El medidor de energía diseñado en la Escuela de Ingeniería Eléctrica, es capaz de operar en un sistema trifásico 204 V línea a línea y 5 A rms, para el canal de voltaje y canal de corriente respectivamente. Realizando muestreo de parámetros como voltaje, corriente, potencia y factor de potencia. Además es capaz de realizar mediciones sobre diferentes tipos de servicios, lo cual puede llevarse a cabo mediante la combinación de jumpers dentro del medidor, por defecto esta configurado para la configuración estrella 4 hilos.

Los tipos de conexiones soportados incluyen medición estrella 3 conductores, medición delta 3 conductores, medición estrella 4 conductores y medición delta 4 conductores. El medidor esta conformado por 6 canales de entrada analógicos, 3 de voltaje y 3 de corriente.

El hardware del medidor esta conformado por las siguientes etapas:

- Tarjeta del medidor de energía
- Tarjeta de temporización
- Tarjeta de Memoria

La comunicación entre el medidor y la computadora es serial (RS232) y el software fue desarrollado en LabVIEW. Se inicia dando una breve descripción de cada una de las etapas del hardware. Ver figura 2.1

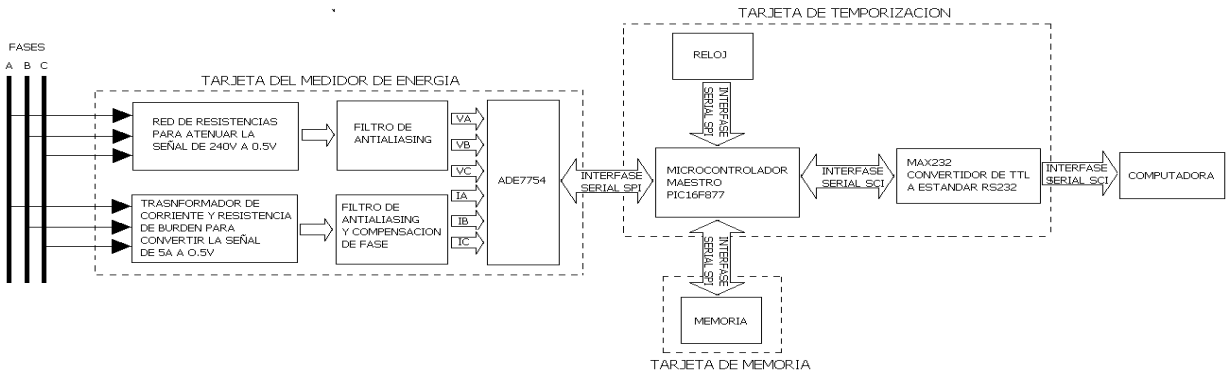

Figura 2.1 Diagrama de bloques del medidor trifásico multifunción.

2.0.1 Tarjeta del medidor de energía

El corazón del medidor lo conforma el ADE7754 que es un IC medidor de energía eléctrica trifásico de alta exactitud (ver figura 2.2) y provee comunicación serial hacia otros dispositivos, en este caso al microcontrolador que se encargara de interrogarlo para la adquisición de los parámetros eléctricos.

El IC ADE7754 es distribuido por Analog Device, siendo su arquitectura versátil y bajo costo clave para su elección. El IC también posee un servicio de interrupción que unido a la comunicación serial, permite conectarlo a un microcontrolador, permitiéndole la capacidad de ser un dispositivo autónomo con capacidad para almacenamiento y descarga de datos a través de un computador.

EL IC ADE7754 permite realizar ajustes por canal, mediante la escritura de los registros internos, como: Corrección de offset, Calibración de fase, Calibración de ganancia.

El IC ADE7754 proporciona la siguiente información:

- Energía activa
- Energía aparente
- Cálculo RMS simultáneo en las seis entradas análogas.
- Muestras de la forma de onda de los 6 canales analógicos con diferentes frecuencias de muestreo 3.3 KSPS, 6.5 KSPS, 13 KSPS Y 26 KSPS.
- Detección de picos de voltaje y corriente

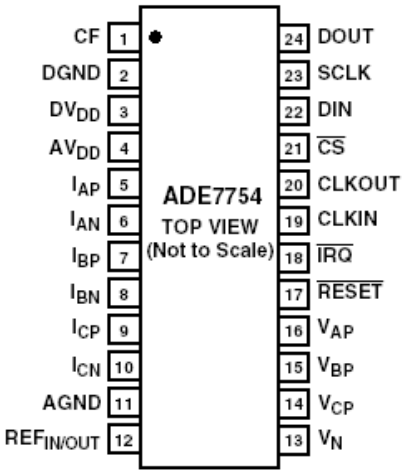

Figura 2.2 Diagrama de pines del IC ADE7754

Para que el IC ADE7754 pueda operar necesita 355 mV rms en sus entradas analógicas, para ello cada canal está provisto de una etapa de acondicionamiento de señal conformada por la red de atenuación y por filtros anti-alias en los 6 canales y filtros de compensación de fase para los canales de corriente ya que el uso de transformadores de corriente (TC) introduce un desfase que debe de ser corregido (ver figura 2.3)

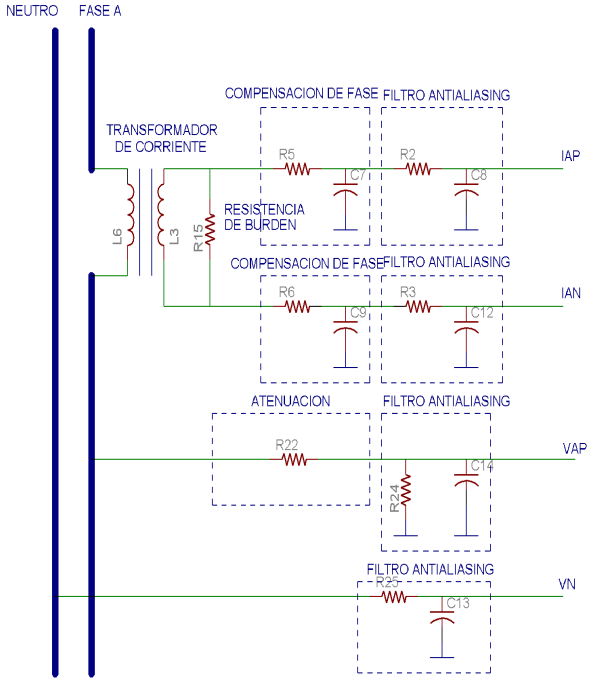

Figura 2.3 Etapa de la señal de canales de voltaje y corriente

La tarjeta del medidor trifásico esta conformada por la etapa de acondicionamiento de señal para los canales analógicos de voltaje y corriente del ADE7754 de las tres fases de alimentación; además de las líneas de conexión que corresponden al bus de comunicación SPI de microcontrolador maestro PIC16F877 y los indicadores LED del contador de frecuencia (CF) para la potencia activa y la salida /IRQ que indica la petición de un servicio de interrupción. Cabe destacar que el ADE7754 posee un cristal de operación de 10Mhz que es el valor recomendado por fabricante para una operación óptima del IC ADE7754. Ver figura 2.4

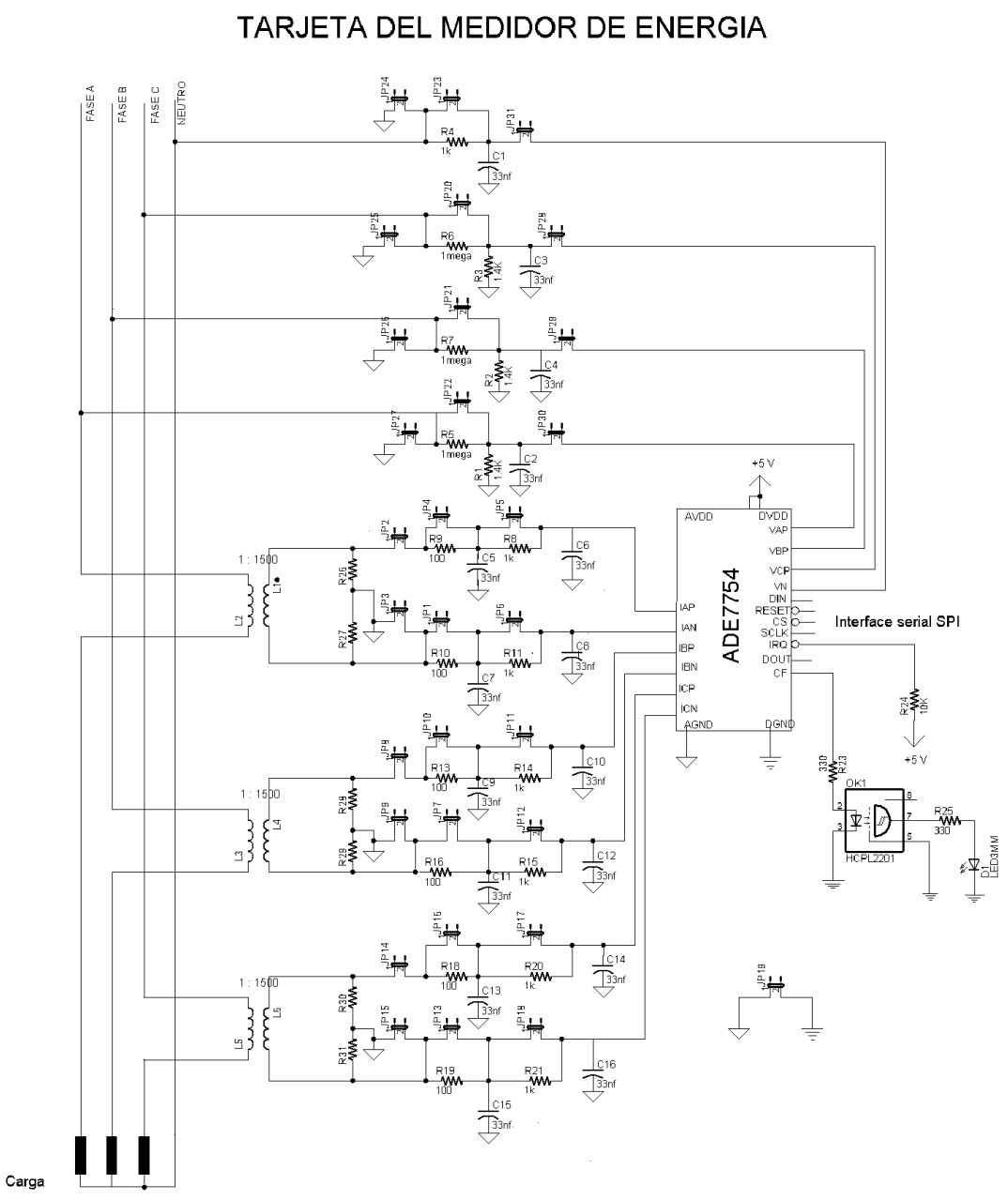

Figura 2.4 Tarjeta del IC ADE7754

2.0.2 Tarjeta de la etapa de memoria

La tarjeta esta provista de una memoria tipo RAM estática de una capacidad de 128kx8 del tipo DS1245Y-120 de la compañía DALLAS. La comunicación entre todas las etapas se realiza por la interfaz serial SPI para comunicación con el microcontrolador maestro. Por esta razón se tuvo la necesidad de utilizar un microcontrolador PIC16F877 para su implementación ya que esta memoria DS1245Y-120 posee un bus de direcciones de 17 líneas en paralelo para poder acceder a las 128K localidades de memoria, siendo el puerto E del microcontrolador idóneo para esta tarea. Este microcontrolador funciona en la modalidad de esclavo y estará siendo controlado por el PIC principal del medidor cuando éste así lo solicite. Ver figura 2.5

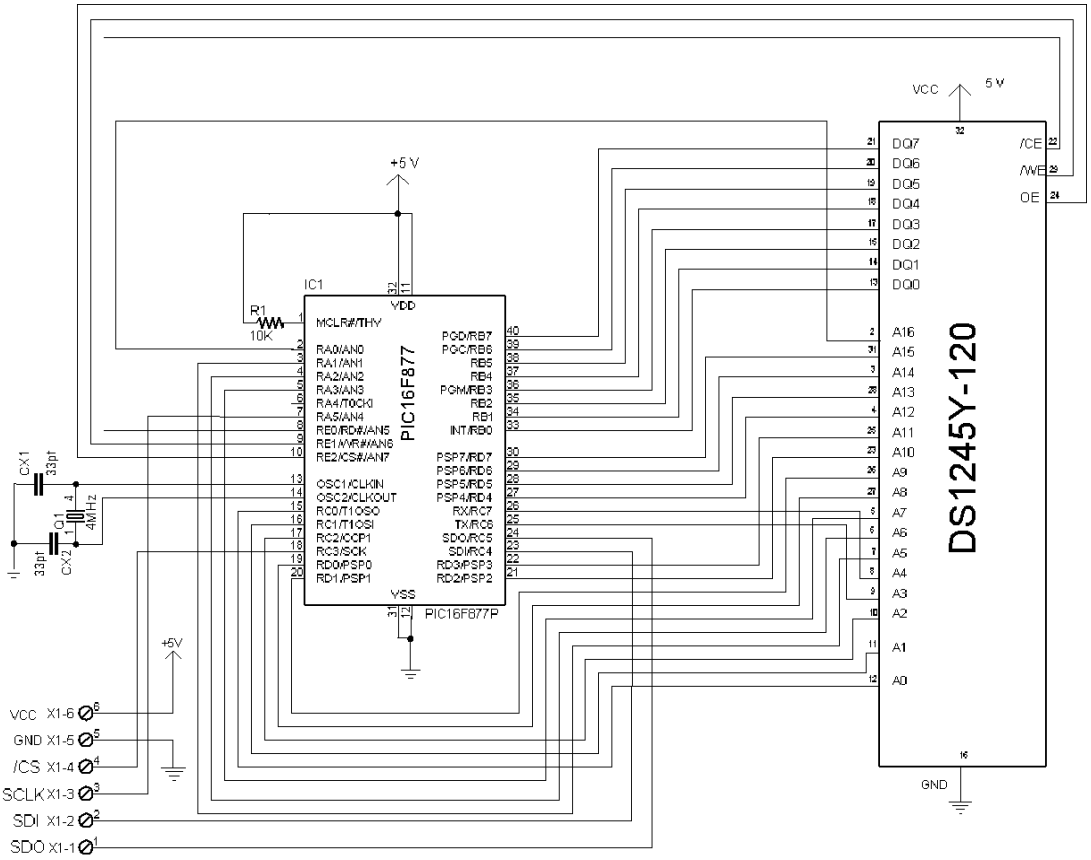

Figura 2.5 Diagrama de implementación de una memoria con interfase SPI.

2.0.3 Tarjeta de Temporización

Esta etapa esta compuesta por dos microcontroladores PIC16F877, uno de ellos maestro y el otro forma la implementación del reloj para mantener la comunicación con interfaz SPI. El PIC maestro coordina todos los eventos del medidor, estableciendo comunicación con la memoria, el ADE7754 y el reloj. Dicho reloj posee un formato de salida de 4 bytes que corresponden al tiempo en segundos que posteriormente se tratará desde la computadora para llevar bitácora de las actividades en segundos, minutos, horas, días y meses para un completo historial de las mediciones realizadas. Ver figura 2.6

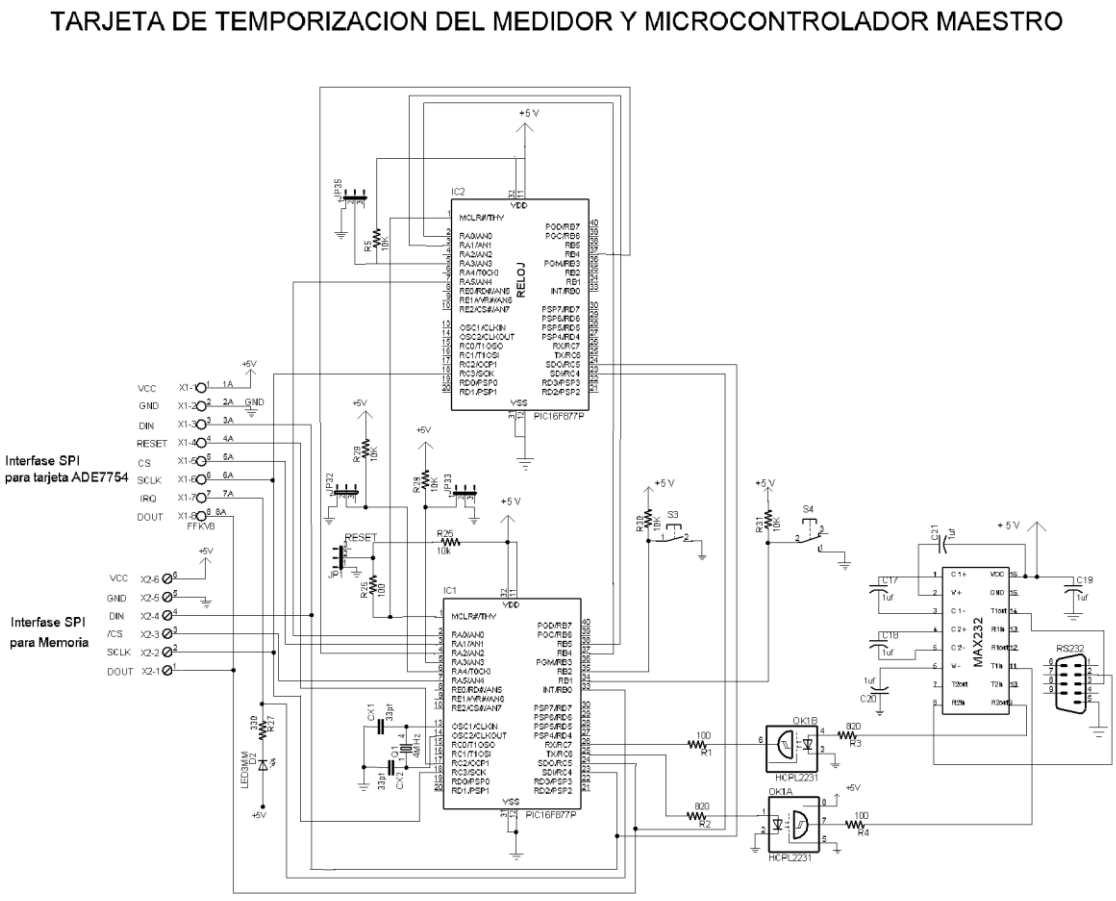

Figura 2.6 Tarjeta de Temporización donde se incluyen también el microcontrolador maestro y el convertidor TTL-RS232

2.1 Protocolo de Comunicación y Ambiente Gráfico

El protocolo de comunicación utilizado por el medidor para comunicación con la computadora es serial, a través del puerto RS232. Como se muestra en la figura 2.7

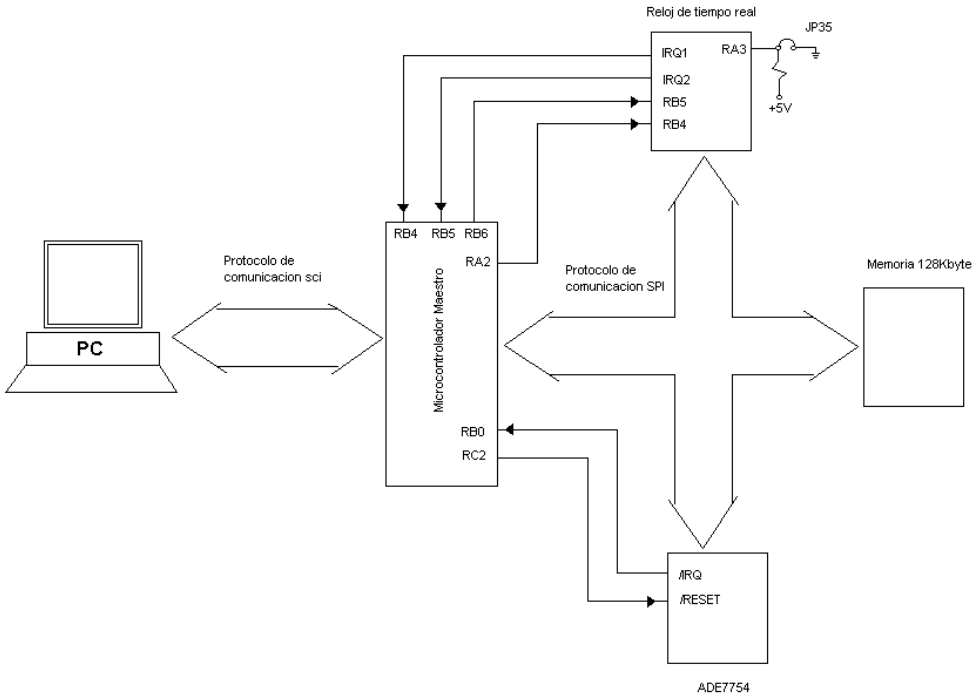

Figura 2.7 Diagrama de bloques del sistema

Si bien la comunicación utilizando el puerto RS232 cubre la necesidad de comunicación hacia la computadora para la adquisición de datos, calibración del equipo, etc. Esto involucra una constante visita al lugar en donde esta instalado el medidor de energía para la recolección de los datos, o la necesidad de tener una maquina dedicada exclusivamente al medidor en su cercanía.
Ambas alternativas resultan caras y poco atractivas al empresario, por lo cual el monitoreo remoto de los parámetros del medidor se ha vuelto hoy en día una característica indispensable en los medidores de energía eléctrica.

En el medidor de energía creado en la escuela de Ingeniería Eléctrica para llevar a cabo la comunicación serial SCI (puerto RS232), los diseñadores crearon un protocolo de comunicación propio, sin embargo, esto involucra un directo lazo con el software que ellos mismos desarrollaron y elimina la posibilidad de integración con otros medidores a través de un software que maneja protocolos estándar.

El software fue desarrollado en LABVIEW, un lenguaje de programación grafico, que presenta gran flexibilidad para el desarrollo de aplicaciones de adquisición de datos e instrumentación. La interfase grafica de usuario (GUI) es básicamente el panel frontal del instrumento virtual (VI), y bajo él se encuentra el diagrama de bloques que es la estructura del programa.

El VI se divide básicamente en dos partes:

- Programa principal
- Programa de calibración

Programa principal

En el programa principal se puede escoger la operación a realizar, entre algunas de ellas:

- Lectura actual del medidor de valores rms y de potencia
- Lectura de archivo de valores rms y de potencia entre dos fechas
- Lectura de archivo de valores rms y de potencia de una fecha especifica
- Descarga del medidor de valores rms y de tiempo
- Generación de archivos de reporte

Otros VI son llamados por el VI principal (ver figura 2.8), para lograr una interfaz grafica completa.

Figura 2.8 VI principal

VI de calibración

La función de este panel como su nombre lo dice es calibrar el instrumento, para ello establece comunicación con el ADE7754 y modifica sus registros. Todo este proceso se lleva a cabo a cabo usando el puerto RS232 y la comunicación SPI entre el PIC maestro y el ADE7754.

Es posible regular los registros de ganancia de la potencia activa, la ganancia de voltaje y corriente, el offset del voltaje y corriente, y además ajustar la constante del medidor. Se tiene la opción de registrar estos cambios en la memoria EEPROM del medidor, a fin de mantener los cambios de manera permanente. El panel principal se muestra en la figura 2.9

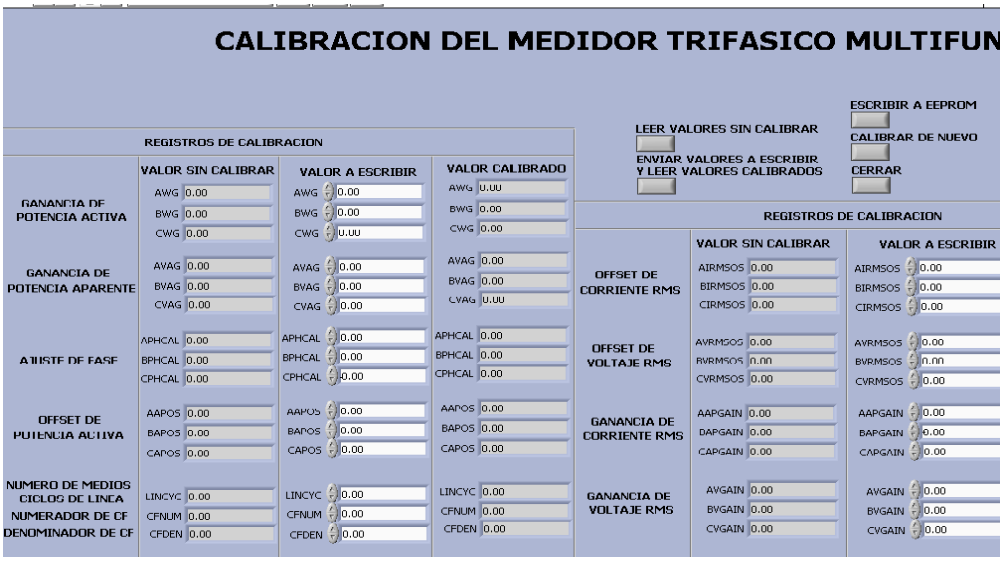

Figura 2.9 VI de calibración

Todos los aspectos relacionados al medidor de energía son especificados en el trabajo de graduación llamado “Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754” [1].

2.2 Cambios a realizar

Hasta este momento este capitulo ha sido dedicado a explicar el funcionamiento y arquitectura general del medidor de energía diseñado en la escuela de Ingeniería Eléctrica, todo esto con el objetivo de presentar el instrumento sobre el cual se basa el presente trabajo de graduación.

En esta sección daremos un resumen de la propuesta de cambios en el medidor de energía para llevar a cabo el monitoreo remoto del mismo, y a la vez cumplir con los demás objetivos, como los son:

1- Seleccionar un protocolo de comunicación estándar reconocido internacionalmente.
2- Seleccionar un protocolo de comunicación ampliamente usado por otros medidores de energía para una fácil integración y compatibilidad con los mismos.
3- Seleccionar un protocolo de comunicación que se acople a las necesidades emergentes en futuros proyectos en el área de redes, dentro de la escuela de Ingeniería Eléctrica.
4- Obtener un sistema que siga manteniendo su bajo costo.
5- Desarrollar una aplicación grafica de usuario para el monitoreo de los parámetros básicos del medidor.

Con esto en mente y con el campo de las comunicaciones en constante desarrollo no podemos aventurarnos a tomar una decisión sin antes investigar la forma en que otros medidores llevan a cabo el monitoreo remoto, los protocolos que utilizan y su respectivo software.
Debido a la creciente intrusión de redes Ethernet en múltiples empresas, el acceso a redes LAN[14] locales cada vez es mas común, volviéndose una alternativa potente para el monitoreo de instrumentos. En el mercado internacional muchos de los medidores más competitivos poseen capacidad para conexión a redes Ethernet, utilizando diferentes tipos de protocolos dependiendo de la marca de medidor.

Actualmente existe una gran variedad de protocolos destinados a diferentes sectores, por lo que su selección es de gran importancia, ya que cada uno esta optimizado para rendir de la mejor forma en cada campo.
El medidor en cuestión ya posee comunicación serial vía RS232, y para no alterar el sistema existente se pretende diseñar e implementar una tarjeta que sea capaz de comunicarse con el medidor por el puerto RS232 y posea salida para conector RJ45 hacia una red Ethernet. Dicha tarjeta debe poseer la instrumentación necesaria para manejar los protocolos inmersos en redes Ethernet, y el protocolo destinado a comunicar el medidor con el cliente.
Esta tarjeta ya es comercializada, sin embargo la utilización de tecnología extranjera nos amarra a una empresa específica. Por lo cual el desarrollo del sistema completo de monitoreo es una necesidad para mantener los bajos costos y un diseño propio de la Escuela de Ingeniería Eléctrica.

[14] LAN *Local Área Network*

Sistema de monitoreo a desarrollar

Para cubrir con los objetivos y llevar a cabo el monitoreo remoto del medidor de energía eléctrica se plantea la siguiente solución:

Se desarrollará una tarjeta que servirá de enlace entre el medidor de energía con puerto RS232 y la red Ethernet, utilizando protocolos estándar de reconocimiento internacional para una fácil integración a un software comercial que “hable” el mismo protocolo.

Con esta alternativa se evita modificar el Hardware existente del medidor y nos enfocamos únicamente en la comunicación de los parámetros brindados. Además se provee de un hardware y software independiente del medidor que lo vuelve más flexible, ya que puede ser incorporado en otros dispositivos que se comuniquen por el mismo puerto RS232 y deseen monitorearse remotamente usando Ethernet y MODBUS TCP.

Un panorama general del sistema completo de monitoreo se puede ver en la figura 2.10. Como se muestra en dicha figura se utilizara una red Ethernet para la comunicación, y dentro de este medio se utilizara el protocolo TCP/IP. Las razones se explicaran posteriormente.

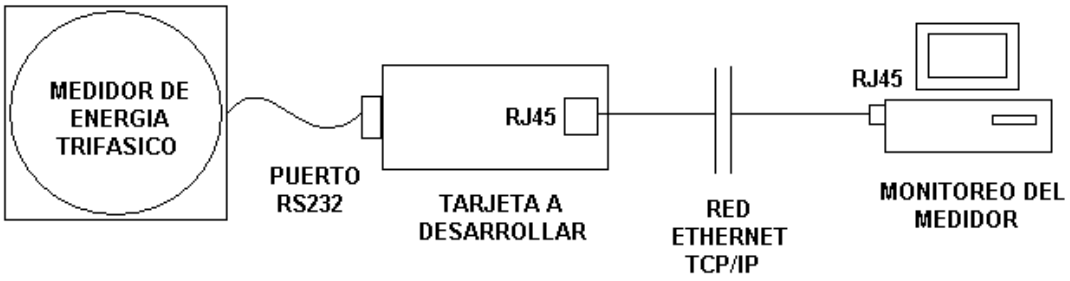

Figura 2.10 Sistema de monitoreo a desarrollar.

CAPÍTULO III

TEORÍA BÁSICA

Introducción

En el presente capítulo se abordarán los conceptos básicos para la implementación de la comunicación entre computadores y el instrumento de medición, se presentan los conceptos básicos de los distintos protocolos a utilizar, tales como el protocolo ETHERNET, ARP, ICMP, IP, TCP y del protocolo de la capa de aplicación MODBUS TCP y MODBUS serial.

Generalmente, la información que se desplaza por una red ethernet recibe el nombre de datos o paquete[15]. A medida que los datos atraviesan las capas, cada capa agrega información que posibilita una comunicación eficaz con su correspondiente capa en el otro computador.
Los modelos OSI[16] y TCP[17]/IP[18] se dividen en capas que explican cómo los datos se comunican de un computador a otro. Los modelos difieren en la cantidad y la función de las capas. No obstante, se puede usar cada modelo para ayudar a describir y brindar detalles sobre el flujo de información desde un origen a un destino.

Para que los paquetes de datos puedan viajar desde el origen hasta su destino a través de una red, es importante que todos los dispositivos de la red hablen el mismo lenguaje o protocolo[19].

[15] Un paquete es una unidad de información, lógicamente agrupada, que se desplaza entre los sistemas de computación
[16] Open Systems Interconnection
[17] Transmission Control Protocol
[18] Internet Protocol
[19] Un protocolo es un conjunto de reglas que hacen que la comunicación en una red sea más eficiente

3.0 Modelo OSI

El modelo de referencia de Interconexión de Sistemas Abiertos (OSI) fue el modelo de red descriptivo creado por ISO[20]. Proporcionó a los fabricantes un conjunto de estándares que aseguraron una mayor compatibilidad e interoperabilidad entre los distintos tipos de tecnología de red producidos por las empresas a nivel mundial.

El modelo de referencia OSI es un marco que se puede utilizar para comprender cómo viaja la información a través de una red. El modelo de referencia OSI explica de qué manera los paquetes de datos viajan a través de varias capas a otro dispositivo de una red, aun cuando el remitente y el destinatario poseen diferentes tipos de medios de red.

En el modelo de referencia OSI, hay siete capas numeradas las cuales se muestran en la figura 3.1, cada una de las cuales ilustra una función de red específica.

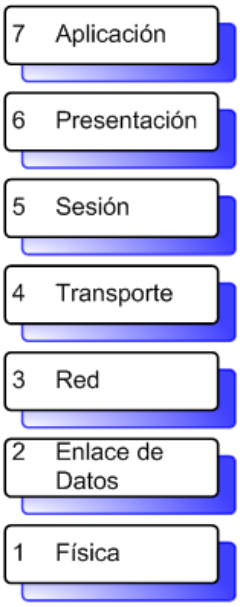

Figura 3.1 Capas del Modelo OSI

Capa Física: Transmisión binaria que involucra cables, conectores, voltajes, velocidades de transmisión de datos.

[20] International Organization for Standardization

Capa Enlace de Datos: Control directo de enlaces, acceso a los medios. Provee transferencia confiable de datos a través de los medios, conectividad y selección de ruta entre sistemas, direccionamiento lógico.

Capa de Red: Dirección de red y determinación de mejor ruta. Provee transferencia confiable de datos a través de los medios, conectividad y selección de ruta entre sistemas.

Capa de Transporte: conexiones de extremo a extremo. Se ocupa de aspectos de transporte entre hosts, confiabilidad del transporte de datos, detección de fallas y control de flujo de información de recuperación.

Capa de Sesión: comunicación entre hosts. Establece, administra y termina sesiones entre aplicaciones.

Capa de Presentación: presentación de datos. Garantizar que los datos sean legibles para el sistema receptor, formato de los datos, estructura de datos, negocia la sintaxis de transferencia de datos para la capa de aplicación.

Capa de Aplicación: procesos de red a aplicaciones. Suministra servicios de red a los procesos de aplicaciones.

3.1 Modelo TCP/IP

TCP/IP se desarrolló como un estándar abierto. Esto significaba que cualquier persona puede usar el TCP/IP. Esto contribuyó a acelerar el desarrollo de TCP/IP como un estándar.

El modelo TCP/IP tiene las siguientes cuatro capas que se muestran el la figura 3.2:

- Capa de Aplicación
- Capa de Transporte
- Capa de Internet
- Capa de Acceso a la red

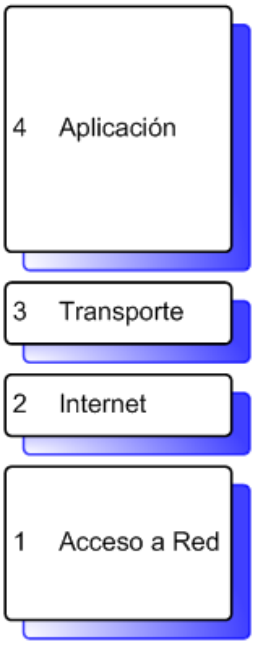

Figura 3.2 Capas del Modelo TCP/IP

Aunque algunas de las capas del modelo TCP/IP tienen el mismo nombre que las capas del modelo OSI, las capas de ambos modelos no se corresponden de manera exacta.

Capa de Aplicación: la capa de aplicación posee funciones diferentes en cada modelo. Los diseñadores de TCP/IP sintieron que la capa de aplicación debía incluir los detalles de las capas de sesión y presentación OSI. Crearon una capa de aplicación que maneja aspectos de representación, codificación y control de diálogo.
Algunos de los protocolos de capa de aplicación más comúnmente usados incluyen los siguientes:

- Protocolo de Transferencia de Archivos (FTP)
- Protocolo de Transferencia de Hipertexto (HTTP)
- Protocolo simple de transferencia de correo (SMTP)
- Sistema de denominación de dominios (DNS)
- Protocolo Trivial de Transferencia de Archivos (TFTP)

Pero para este trabajo se utilizará el Protocolo **MODBUS/TCP** de la capa de aplicación.

Capa de Transporte: la capa de transporte se encarga de los aspectos de calidad del servicio con respecto a la confiabilidad, el control de flujo y la corrección de errores. Uno de sus protocolos, el protocolo para el control de la

transmisión (TCP), ofrece maneras flexibles y de alta calidad para crear comunicaciones de red confiables, sin problemas de flujo y con un nivel de error bajo.
TCP es un protocolo orientado a conexión. Mantiene un diálogo entre el origen y el destino mientras empaqueta la información de la capa de aplicación en unidades denominadas segmentos. Orientado a conexión no significa que existe un circuito entre los computadores que se comunican. Significa que segmentos de la capa 4 viajan de un lado a otro entre dos hosts para comprobar que la conexión exista lógicamente para un determinado período.

Capa de Internet: el propósito de la capa Internet es dividir los segmentos TCP en paquetes y enviarlos desde cualquier red. Los paquetes llegan a la red de destino independientemente de la ruta que utilizaron para llegar allí. El protocolo específico que rige esta capa se denomina Protocolo Internet (IP). En esta capa se produce la determinación de la mejor ruta y la conmutación de paquetes.
La relación entre IP y TCP es importante. Se puede pensar en el IP como el que indica el camino a los paquetes, en tanto que el TCP brinda un transporte seguro.

Capa de Acceso a red: El nombre de la capa de acceso de red es muy amplio y se presta a confusión. También se conoce como la capa de host a red. Esta capa guarda relación con todos los componentes, tanto físicos como lógicos, necesarios para lograr un enlace físico. Incluye todos los detalles de la capa física y de enlace de datos del modelo OSI.

3.2 Comparación entre el modelo OSI y TCP/IP

Figura 3.3 Comparación entre el modelo OSI y TCP/IP

Las similitudes incluyen:

- Ambos se dividen en capas.
- Ambos tienen capas de aplicación, aunque incluyen servicios muy distintos.
- Ambos tienen capas de transporte y de red similares.
- Ambos suponen que se conmutan paquetes. Esto significa que los paquetes individuales pueden usar rutas diferentes para llegar al mismo destino. Esto se contrasta con las redes conmutadas por circuito, en las que todos los paquetes toman la misma ruta.

Las diferencias incluyen:

- TCP/IP combina las funciones de la capa de presentación y de sesión en la capa de aplicación.
- TCP/IP combina la capa de enlace de datos y la capa física del modelo OSI en la capa de acceso de red.
- TCP/IP parece ser más simple porque tiene menos capas.

Los protocolos TCP/IP son los estándares en torno a los cuales se desarrolló Internet, de modo que la credibilidad del modelo TCP/IP se debe en gran parte a sus protocolos. En comparación, por lo general las redes no se desarrollan a partir del protocolo OSI, aunque el modelo OSI se usa como guía.

A lo largo de todo el documento se hará referencia al TCP/IP. Se hará énfasis en lo siguiente:

- MODBUS como un protocolo de Capa 4 TCP/IP
- TCP como un protocolo de Capa 3 TCP/IP
- IP como un protocolo de Capa 2 TCP/IP
- Ethernet como una tecnología de Capa 1 TCP/IP

Hay una diferencia entre un modelo y un protocolo. Se utilizará el modelo TCP/IP para describir los distintos protocolos TCP/IP.

3.3 Acceso a red (Capa 1)

3.3.1 IEEE[21] Std 802.3 (Ethernet)

Ethernet es ahora la tecnología LAN[22] dominante en el mundo. Ethernet no es una tecnología sino una familia de tecnologías LAN que se pueden entender mejor utilizando el modelo de referencia OSI. Todas las LAN deben afrontar el

[21] Institute of Electrical and Electronics Engineers
[22] Local Area Network

tema básico de cómo denominar a las estaciones individuales (nodos) y Ethernet no es la excepción. Las especificaciones de Ethernet admiten diferentes medios, anchos de banda y demás variaciones de la capa 1 y 2. Sin embargo, el formato de trama básico y el esquema de direccionamiento son igual para todas las variedades de Ethernet.

3.3.2 Reglas del IEEE para la denominación de Ethernet

Ethernet no es una tecnología, sino una familia de tecnologías que incluye Legacy, Fast Ethernet y Gigabit Ethernet. Las velocidades de Ethernet pueden ser de 10, 100, 1000 ó 10000 Mbps. El formato básico de la trama y las subcapas del IEEE de las Capas OSI 1 y 2 siguen siendo los mismos para todas las formas de Ethernet.

Cuando es necesario expandir Ethernet para agregar un nuevo medio o capacidad, el IEEE publica un nuevo suplemento del estándar 802.3 aunque el IEEE no puede forzar a los fabricantes a cumplir con todas las particularidades de ningún estándar.

Ethernet opera en dos áreas del modelo OSI, la mitad inferior de la capa de enlace de datos, conocida como subcapa MAC[23] y la capa física, como se muestra en la figura 3.4

3.3.3 Entramado de la capa de acceso a red

Las corrientes de bits codificadas (datos) en medios físicos representan un logro tecnológico extraordinario, pero por sí solas no bastan para que las comunicaciones puedan llevarse a cabo. El entramado ayuda a obtener información esencial que, de otro modo, no se podría obtener solamente con las corrientes de bits codificadas.
El entramado es el proceso de encapsulamiento de la capa 1. Una trama es la unidad de datos del protocolo de la capa 1.

[23] Media Access Control

Figura 3.4 Ubicación de Ethernet 802.3 dentro del Modelo OSI

Hay varios tipos distintos de tramas que se describen en diversos estándares. En la figura 3.5 se muestra el formato de una trama genérica, una trama genérica tiene secciones denominadas campos, y cada campo está formado por bytes. Los nombres de los campos son los siguientes:

- Campo de inicio de trama
- Campo de dirección
- Campos de longitud/tipo
- Campo de datos
- Campo de secuencia de verificación de trama

Nombres de Campos				
A	B	C	D	E
Campo de Inicio de Trama	Campo de Dirección	Campo de Tipo/Longitud	Campo de Datos	Campo de FCS

Figura 3.5 Formato de la trama genérica

Cuando los computadores se conectan a un medio físico, debe existir alguna forma de informar a los otros computadores cuando están próximos a enviar una trama. Las diversas tecnologías tienen distintas formas para hacerlo, pero todas las tramas, de cualquier tecnología, tienen una secuencia de señalización de inicio de bytes.

La mayoría de las tramas tienen algunos campos especializados. En algunas tecnologías, el campo "longitud" especifica la longitud exacta de una trama en bytes. Algunas tienen un campo "tipo", que especifica el protocolo de Capa 2 que realiza la petición de envío.

La razón del envío de tramas es hacer que los datos de las capas superiores, especialmente los datos de aplicación del usuario, lleguen desde el origen hasta el destino. El paquete de datos incluye el mensaje a ser enviado, o los datos de aplicación del usuario. Puede resultar necesario agregar bytes de relleno de modo que las tramas tengan una longitud mínima para los fines de temporización. Los bytes de control de enlace lógico (LLC[24]) también se incluyen en el campo de datos de las tramas del estándar IEEE. La subcapa LLC toma los datos de protocolo de la red, un paquete IP, y agrega información de control para ayudar a entregar ese paquete IP al nodo de destino. La capa 1 se comunica con las capas de nivel superior a través de LLC.

Todas las tramas y los bits, bytes y campos ubicados dentro de ellas, están susceptibles a errores de distintos orígenes. El campo de Secuencia de verificación de trama (FCS[25]) contiene un número calculado por el nodo de origen en función de los datos de la trama. Entonces, esta FCS se agrega al final de la trama que se envía. Cuando el computador destino recibe la trama, se vuelve a calcular el número FCS y se compara con el número FCS que se incluye en la trama. Si los dos números son distintos, se da por sentado que se ha producido un error, se descarta la trama y se le puede pedir al origen que vuelva a realizar la transmisión. Debido a que la fuente no puede detectar que la trama ha sido descartada, se deben iniciar retransmisiones por un protocolo de capa superior orientado a conexión que provea control de flujo de datos. Usualmente se dan retransmisiones debido a que los protocolos, como TCP/IP, requieren que las estaciones envíen tramas de reconocimiento, ACK[26], dentro de un tiempo preestablecido.

Hay tres formas principales para calcular el número de Secuencia de verificación de trama:

Verificación por redundancia cíclica (CRC[27]): Realiza cálculos en los datos.

[24] Logical Link Control
[25] Frame Check Sequence
[26] acknowledge
[27] Cyclic Redundancy Check

Paridad bidimensional: Coloca a cada uno de los bytes en un arreglo bidimensional y realiza chequeos verticales y horizontales de redundancia sobre el mismo, creando así un byte extra, que resulta en un número par o impar de unos binarios.

Checksum (suma de verificación) de Internet: Agrega los valores de todos los bits de datos para obtener una suma
El nodo que transmite los datos debe llamar la atención de otros dispositivos para iniciar una trama y para finalizar la trama. El campo de longitud implica el final y se considera que la trama termina después de la FCS. A veces hay una secuencia formal de bytes que se denomina delimitador de fin de trama

3.3.4 Formato de la trama MAC IEEE 802.3

En la figura 3.6 se muestran los nueve campos de la trama: el preámbulo, el delimitador de inicio de trama (Start Frame Delimiter SFD), las direcciones de las tramas destino y fuente, un campo de longitud o tipo para indicar la longitud o tipo de protocolo del siguiente campo que contiene los datos del cliente MAC, y un campo que contiene relleno (PAD) si es requerido, el campo de verificación de secuencia de trama y un campo de extensión si es requerido. De estos nueve campos, todos son de longitudes fijas, excepto para los campos de dato, relleno, y extensión, los cuales pueden contener un número entero de bytes entre los valores mínimos y máximos que están determinados por la implementación específica de el CSMA/CD MAC.

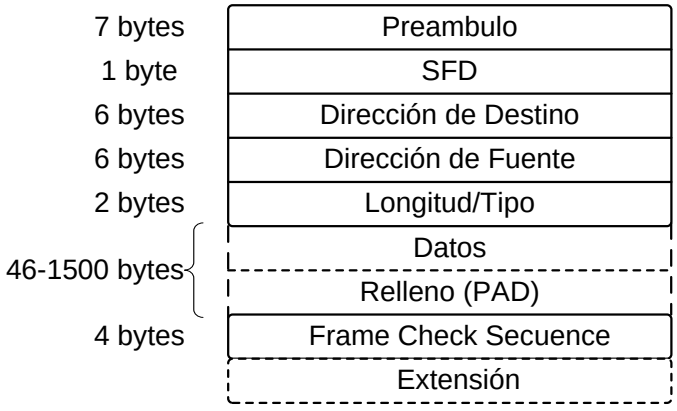

Figura 3.6 Formato de la trama MAC

3.3.5 Campos de la trama MAC IEEE 802.3

Los nueve campos de la trama MAC son los siguientes y su descripción se muestra en la Tabla 3.1

Preámbulo: El preámbulo es un campo de 7 bytes que es usado para permitir a la PLS[28] alcanzar su sincronización de estado estable con oportuna trama recibida.

Delimitador de Inicio de trama (SFD): El SDF un campo de un byte y es la secuencia 10101011. Indica el inicio de la trama.

Dirección destino: El campo de dirección destino, especifica la estación o estaciones para la cual la trama se ha destinado, es decir contiene la dirección MAC destino. Puede ser una dirección unicast o multicast (incluyendo broadcast).

Dirección Origen: El campo de dirección de origen contiene la dirección MAC de origen. La dirección origen generalmente es la dirección unicast del nodo de transmisión de Ethernet.

Longitud/Tipo: El campo Longitud/Tipo admite dos usos diferentes. Si el valor es menor a 1536 decimal, 0x600 (hexadecimal), entonces el valor indica la longitud. La interpretación de la longitud se utiliza cuando la Capa LLC proporciona la identificación del protocolo. El valor del tipo especifica el protocolo de capa superior que recibe los datos una vez que se ha completado el procesamiento de Ethernet. La longitud indica la cantidad de bytes de datos que sigue este campo.

Datos y Relleno: Los Campos de datos y de relleno, de ser necesario, pueden tener cualquier longitud, mientras que la trama no exceda el tamaño máximo permitido de trama. La unidad máxima de transmisión (MTU) para Ethernet es de 1500 bytes, de modo que los datos no deben superar dicho tamaño. El contenido de este campo no está especificado. Se inserta un relleno no especificado inmediatamente después de los datos del usuario cuando no hay suficientes datos de usuario para que la trama cumpla con la longitud mínima especificada. Ethernet requiere que cada trama tenga entre 64 y 1518 bytes de longitud.

FCS: Una FCS contiene un valor de verificación CRC de 4 bytes, creado por el dispositivo emisor y recalculado por el dispositivo receptor para verificar la existencia de tramas dañadas. Ya que la corrupción de un solo bit en cualquier punto desde el inicio de la dirección destino hasta el extremo del campo de FCS hará que la checksum (suma de verificación) sea diferente, la cobertura de la

[28] Physical Layer Signaling

FCS se auto-incluye. No es posible distinguir la corrupción de la FCS en sí y la corrupción de cualquier campo previo que se utilizó en el cálculo.

Una trama recibida que tiene una Secuencia de verificación de trama incorrecta, también conocido como error de CRC o de checksum, difiere de la transmisión original en al menos un bit. En una trama con error de FCS, es probable que la información del encabezado sea correcta, pero la checksum que calcula la estación receptora no concuerda con la checksum que adjunta la estación transmisora al extremo de la trama. Por lo tanto, se descarta la trama.

Bytes	Descripción
7	Preámbulo
1	Delimitación de inicio de trama (SFD)
6	Dirección MAC de destino
6	Dirección MAC de origen
2	Campo de Longitud/Tipo (longitud si es menos de 0600 hexadecimal, de lo contrario tipo de protocolo)
46 a 1500	Datos (si es menos de 46 bytes, se debe agregar un relleno al final)
4	Secuencia de verificación de trama (suma de comprobación CRC)

Tabla 3.1 Campos de tramas Ethernet

Una gran cantidad de errores FCS provenientes de una sola estación indican, por lo general, una NIC[29] defectuosa y/o falla o corrupción en los controladores del software, o un cable defectuoso que conecta esa estación a la red. Si los errores FCS están asociados con muchas estaciones, por lo general, pueden rastrearse a la presencia de un cableado defectuoso, una versión defectuosa del controlador de la NIC, un puerto de hub defectuoso o a ruido inducido en el sistema de cables.

Una trama con un valor válido en el campo "longitud" pero que no concuerda con el número real de bytes contabilizados en el campo de datos de la trama recibida recibe el nombre de error de rango. Este error también aparece cuando el valor del campo de longitud es menor que el tamaño mínimo legal sin relleno para el campo de datos. Un error, similar, Fuera de rango, se informa cuando el valor del campo "longitud" indica que el tamaño de los datos es demasiado grande para ser legal.

[29] Network. Interface Card

3.4 Internet (Capa 2)

3.4.1 Protocolo de Internet (IPv4)

El Protocolo Internet está diseñado para su uso en sistemas interconectados de redes de comunicación de computadores por intercambio de paquetes. El protocolo Internet proporciona los medios necesarios para la transmisión de bloques de datos llamados datagramas desde el origen al destino, donde origen y destino son hosts identificados por direcciones de longitud fija. El protocolo Internet también se encarga, si es necesario, de la fragmentación y el reensamblaje de grandes datagramas para su transmisión a través de redes de trama pequeña.
El Protocolo Internet está específicamente limitado a proporcionar las funciones necesarias para enviar un paquete de bits (un datagrama Internet) desde un origen a un destino a través de un sistema de redes interconectadas. No existen mecanismos para aumentar la fiabilidad de datos entre los extremos, control de flujo, secuenciamiento u otros servicios que se encuentran normalmente en otros protocolos host a host.

El protocolo Internet implementa dos funciones básicas: Direccionamiento y Fragmentación.
Los módulos Internet usan las direcciones que se encuentran en la cabecera Internet para transmitir los datagramas Internet hacia sus destinos. La selección de un camino para la transmisión se llama “enrutamiento”.
Los módulos Internet usan campos en la cabecera Internet para fragmentar y reensamblar los datagramas Internet cuando sea necesario para su transmisión a través de redes de "trama pequeña".

Estos módulos comparten reglas comunes para interpretar los campos de dirección y para fragmentar y ensamblar datagramas Internet.

3.4.2 Modelo de operación

La Fig. 3.7 ilustra el lugar del protocolo Internet en la jerarquía de protocolos.
El protocolo Internet interactúa por un lado con los protocolos host a host de alto nivel y por otro con el protocolo de la red local.
El modelo de operación para transmitir un datagrama de una aplicación a otra se ilustra en el siguiente escenario: Suponemos que esta transmisión involucra a un gateway intermedio.

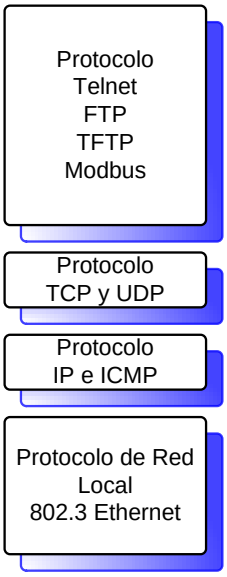

Figura 3.7 Ubicación del Protocolo Internet en la jerarquía de protocolos

La aplicación remitente prepara sus datos y llama a su módulo Internet local para enviar esos datos como un datagrama y pasa la dirección de destino y otros parámetros como argumentos de la llamada.

El módulo Internet prepara una cabecera de datagrama y adjunta los datos a él. El módulo Internet determina una dirección de la red de área local para esta dirección Internet, que en este caso es la dirección de un gateway.
Envía este datagrama y la dirección de red local a la interfaz de red local.

La interfaz de red local crea una cabecera de red local, le adjunta el datagrama y entonces envía el resultado a través de la red local.
El datagrama llega a un host gateway encapsulado en la cabecera de red local, la interfaz de red local desprende esta cabecera y dirige el datagrama hacia el módulo Internet. El módulo Internet determina a partir de la dirección Internet que el datagrama debe ser reenviado a otro host en una segunda red. El módulo Internet determina una dirección de red local para el host de destino. Llama a la interfaz de red local de esa red para enviar el datagrama.

Esta interfaz de red local crea una cabecera de red local y le adjunta el datagrama enviando el resultado al host de destino.
En este host de destino la interfaz de red local le quita al datagrama la cabecera de red local y se lo pasa al módulo Internet.

El módulo Internet determina que el datagrama va dirigido a una aplicación en este host. Pasa los datos a la aplicación en respuesta a una llamada al sistema, pasando la dirección de origen y otros parámetros como resultado de la llamada.

3.4.3 Descripción de funciones

La función o propósito del Protocolo Internet es mover datagramas a través de un conjunto de redes interconectadas. Esto se consigue pasando los datagramas desde un módulo Internet a otro hasta que se alcanza el destino. Los módulos Internet residen en hosts y gateways en el sistema Internet. Los datagramas son encaminados desde un módulo Internet a otro a través de redes individuales basándose en la interpretación de una dirección Internet. Por eso, un importante mecanismo del protocolo Internet es la dirección IP.

En el enrutamiento de mensajes desde un módulo Internet a otro, los datagramas pueden necesitar atravesar una red cuyo tamaño máximo de paquete es menor que el tamaño del datagrama. Para salvar esta dificultad se proporciona un mecanismo de fragmentación en el protocolo Internet.

Direccionamiento
Se establece una distinción entre nombres, direcciones y rutas[30]. Un nombre indica qué buscamos. Una dirección indica dónde está. Una ruta indica cómo llegar allí. El protocolo Internet maneja principalmente direcciones. Es tarea de los protocolos de mayor nivel (es decir, protocolos host a host o entre aplicaciones) hacer corresponder nombres con direcciones. El módulo Internet hace corresponder direcciones de Internet con direcciones de red local. Es tarea de los procedimientos de menor nivel (es decir, redes locales o gateways) realizar la correspondencia entre direcciones de red local y rutas.
Las direcciones son de una longitud fija de 4 bytes (32 bits).

Fragmentación
La fragmentación de un datagrama Internet es necesaria cuando éste se origina en una red local que permite un tamaño de paquete grande y debe atravesar una red local que limita los paquetes a un tamaño inferior para llegar a su destino.

Un datagrama Internet puede ser marcado como "no fragmentar". Todo datagrama Internet así marcado no será fragmentado entre distintas redes bajo ninguna circunstancia. Si un datagrama Internet marcado como "no fragmentar" no puede ser entregado en su destino sin fragmentarlo, entonces debe ser descartado.
El procedimiento de fragmentación y reensamblaje en Internet tiene que ser capaz de dividir un datagrama en un número casi arbitrario de piezas que puedan ser luego reensambladas. El receptor de los fragmentos utiliza el campo de identificación para asegurarse de que no se mezclan fragmentos de distintos datagramas. El campo posición ("offset") le indica al receptor la posición de un

[30] Shoch, J., "Inter-Network Naming, Addressing, and Routing", COMPCON, IEEE Computer Society, Otoño 1978

fragmento en el datagrama original. La posición y longitud del fragmento determinan la porción de datagrama original comprendida en este fragmento.

3.4.4 Formato de la cabecera de Internet

A continuación vemos un resumen del contenido de la cabecera Internet.

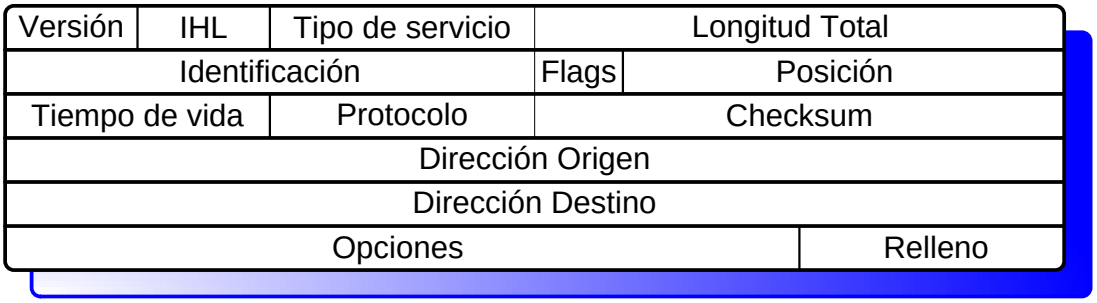

Figura 3.8 Formato de cabecera IP

Versión (4 bits): El campo Versión describe el formato de la cabecera Internet. En este documento se describe la versión 4.

IHL[31] (4 bits): Longitud de la cabecera Internet, es la longitud de la cabecera en palabras de 32 bits, y por tanto apunta al comienzo de los datos. Nótese que el valor mínimo para una cabecera correcta es 5.

Tipo de Servicio (8 bits): El Tipo de Servicio proporciona una indicación de los parámetros abstractos de la calidad de servicio deseada. Estos parámetros se usarán para guiar la selección de los parámetros de servicio reales al transmitir un datagrama a través de una red en particular.

Bit	Valor	Función
0 – 2		Prioridad
3	0	Demora normal
	1	Baja demora
4	0	Rendimiento normal
	1	Alto rendimiento
5	0	Fiabilidad normal
	1	Alta fiabilidad
6 - 7		Reservado para uso futuro

Tabla 3.2 Tipos de servicios de la cabecera IP

[31] Internet Header Length

Algunas redes ofrecen prioridad de servicio, la cual trata de algún modo el tráfico de alta prioridad como más importante que el resto del tráfico (generalmente aceptando sólo tráfico por encima de cierta prioridad en momentos de sobrecarga). La elección más común es un compromiso a tres niveles entre baja demora, alta fiabilidad, y alto rendimiento.

PRECEDENCIA	**D**	**T**	**R**	**0**	**0**

Tabla 3.3 Formato del campo de Tipo de servicio

Precedencia	Función
111	Control de red
110	Control entre redes
101	Criterio/ECP
100	Muy urgente (Flash Override)
011	Urgente (Flash)
010	Inmediato
001	Prioridad
000	Rutina

Tabla 3.4 Funciones que maneja la precedencia

Longitud Total (16 bits): La longitud total es la longitud del datagrama, medida en bytes, incluyendo la cabecera y los datos. Este campo permite que la longitud máxima de un datagrama sea de 65,535 bytes. Los datagramas de tal longitud no son prácticos para la mayoría de hosts y redes. Todos los hosts deben estar preparados para aceptar datagramas de hasta 576 bytes (tanto si llegan completos como en fragmentos). Se recomienda que los hosts envíen datagramas mayores de 576 bytes sólo si tienen la seguridad de que el destinatario está preparado para aceptarlos.

Identificación (16 bits): Es un valor de identificación asignado por el remitente como ayuda en el ensamblaje de fragmentos de un datagrama.

Flags o indicadores (3 bits): Son diversos indicadores de control.

Bit	Valor	Función
0	0	Reservado, debe ser cero
1	0	Puede fragmentarse
	1	No fragmentar
2	0	Último fragmento
	1	Más fragmentos

Tabla 3.5 Indicadores de la cabecera IP

Posición del Fragmento (13 bits): Este campo indica a que parte del datagrama pertenece este fragmento. La posición del fragmento se mide en unidades de 8 bytes (64 bits). El primer fragmento tiene posición 0.

Tiempo de Vida (8 bits):
Es una indicación de un límite superior en el periodo de vida de un datagrama Internet, es decir indica el tiempo que el datagrama tiene permitido permanecer en el sistema Internet. Es fijado por el remitente del datagrama y reducido en los puntos a lo largo de la ruta donde es procesado. Si el tiempo de vida se reduce a cero antes de que el datagrama llegue a su destino, el datagrama Internet es destruido. Puede pensarse en el tiempo de vida como en un plazo de autodestrucción. Este campo es modificado durante el procesamiento de la cabecera Internet. Se debe pensar en el TTL sólo como un límite superior del tiempo durante el cual un datagrama puede existir. La intención es hacer que los datagramas imposibles de entregar sean descartados, y limitar el máximo periodo de vida de un datagrama.

Protocolo (8 bits): Este campo indica el protocolo del siguiente nivel usado en la parte de datos del datagrama Internet. Los valores de varios protocolos son especificados en "Números Asignados"[32]

Suma de Control de Cabecera (16 bits): Proporciona una verificación de que la información utilizada al procesar el datagrama Internet ha sido transmitida correctamente. Los datos pueden contener errores. Si la suma de control de cabecera falla, el datagrama Internet es descartado inmediatamente por la entidad que detecta el error.

El protocolo Internet no proporciona ningún mecanismo de comunicación fiable. No existen acuses de recibo ni entre extremos ni entre saltos. No hay control de errores para los datos, sólo una suma de control de cabecera. No hay retransmisiones. No existe control de flujo.

El algoritmo de la suma de control es: El campo suma de control es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera. A la hora de calcular la suma de control, el valor inicial de este campo es cero.

Opciones (variable): Proporcionan funciones de control necesarias o útiles en algunas situaciones pero innecesarias para las comunicaciones más comunes. Las opciones incluyen recursos para marcas de tiempo, seguridad y enrutamiento especial.

---

[32] Postel, J., "Assigned Numbers," RFC 790, USC/Information Sciences Institute, Septiembre 1981

Las opciones pueden o no aparecer en los datagramas. Deben ser implementadas por todos los módulos IP (host y “gateways”). Lo que es opcional es su transmisión en cualquier datagrama en particular, no su implementación.

3.5 Transporte (Capa 3)

3.5.1 Protocolo de Control de Transmisión (TCP)

El "protocolo de control de transmisión" (Transmission Control Protocol, TCP) está pensado para ser utilizado como un protocolo host a host muy fiable entre miembros de redes de comunicación de computadoras por intercambio de paquetes y en un sistema interconectado de tales redes.

Describiremos las funciones que debe realizar el protocolo de control de transmisión, el programa que lo implementa, y su interfaz con los programas o usuarios que requieran de sus servicios.

TCP se encaja en una arquitectura de protocolos en capas justo por encima de IP (Protocolo de Internet), protocolo básico que proporciona un medio para TCP de enviar y recibir segmentos de longitud variable de información envuelta en "sobres" de datagramas de Internet. El datagrama de Internet proporciona un medio de direccionar TCP´s de origen y de destino situados en redes diferentes. El protocolo de Internet también trata con la fragmentación y el reensamble de segmentos de TCP.

3.5.2 Operación

Como se ha hecho notar más arriba, el propósito principal de TCP consiste en proporcionar un servicio de conexión o circuito lógico fiable y seguro entre pares de procesos. Para proporcionar este servicio encima de un entorno de Internet menos fiable, el sistema de comunicación requiere de mecanismos relacionados con las siguientes áreas: Transferencia básica de datos, Fiabilidad, Control de flujo, Multiplexamiento, Conexiones, Prioridad y seguridad

La operación básica de TCP en cada uno de estas áreas se describe en los siguientes párrafos:

Transferencia básica de datos:
TCP es capaz de transferir un flujo continuo de bits en cada sentido entre sus usuarios empaquetando un cierto número de bytes en segmentos para su transmisión a través del sistema de Internet.
En general, los módulos de TCP deciden cuándo bloquear y enviar datos según su propia conveniencia.
Algunas veces los usuarios necesitan estar seguros de que todos los datos que habían entregado al módulo de TCP han sido transmitidos.

Fiabilidad:
El módulo de TCP debe poder recuperar los datos que se corrompan, pierdan, dupliquen o se entreguen desordenados por el sistema de comunicación del entorno de Internet. Esto se consigue asignando un número de secuencia a cada byte transmitido, y exigiendo un acuse de recibo (ACK, 'acknowledgment') del módulo de TCP receptor. Si no se recibe un ACK dentro de un cierto plazo de expiración prefijado, los datos se retransmiten. En el receptor, se utilizan los números de secuencia para ordenar correctamente los segmentos que puedan haber llegado desordenados y para eliminar los duplicados. La corrupción de datos se trata añadiendo un campo de suma de control ('checksum') a cada segmento transmitido, comprobándose en el receptor y descartando los segmentos dañados.

Flujo de control:
TCP proporciona al receptor un medio para controlar la cantidad de datos enviados por el emisor. Esto se consigue devolviendo una "ventana" con cada ACK, indicando el rango de números de secuencia aceptables más allá del último segmento recibido con éxito. La ventana indica el número de bytes que se permite que el emisor transmita antes de que reciba el siguiente permiso.

Multiplexamiento:
Para permitir que muchos procesos dentro de un único host utilicen simultáneamente las posibilidades de comunicación de TCP, el módulo de TCP proporciona una serie de direcciones o puertos dentro de cada host. Concatenadas con las direcciones de red y de host de la capa de comunicación Internet conforman lo que se denomina una dirección de conector ('socket'). Un par de direcciones de conector identifica de forma única la conexión. Es decir, un conector puede utilizarse simultáneamente en múltiples conexiones.

Conexiones:
La fiabilidad y los mecanismos de control de flujo descritos más arriba exigen que los módulos de TCP inicialicen y mantengan una información de estado para cada flujo de datos. La combinación de esta información, incluyendo las direcciones de los conectores, los números de secuencia y los tamaños de las ventanas, se denomina una conexión. Cada conexión queda especificada de forma única por un par de conectores que corresponden con sus dos extremos.

Cuando dos procesos desean comunicarse, sus módulos de TCP deben establecer primero una conexión (inicializar la información de estado en cada lado). Cuando la comunicación se ha completado, la conexión se termina o cierra con la intención de liberar recursos para otros usos.

Prioridad y seguridad:
Los usuarios de TCP pueden indicar el nivel de seguridad y prioridad de su comunicación. Se emplean valores por defecto cuando estas características no se necesiten.

3.5.3 Modelo de Operación

El entorno de inter-redes consiste en una serie de hosts' conectados a varias redes que a su vez están interconectadas vía gateways. Aquí se supone que las redes pueden ser tanto redes locales (de tipo de ETHERNET) o grandes redes (ARPANET), pero en cualquier caso basadas en tecnología de intercambio de paquetes.

Los procesos transmiten datos llamando al módulo de TCP y pasando búferes de datos como argumentos de la llamada. El módulo de TCP empaqueta en segmentos los datos provenientes de estos búferes y efectúa una llamada al módulo de Internet para que transmita cada segmento al módulo de TCP de destino. El TCP receptor coloca los datos de un segmento en el búfer de recepción del usuario y lo notifica al usuario receptor. Los módulos de TCP incluyen información de control en los segmentos que puede ser utilizada para asegurar una transmisión fiable y ordenada de datos.
El modelo de comunicación del protocolo de Internet es de tal forma que hay un módulo del protocolo Internet asociado con cada módulo de TCP y que, a su vez, proporciona una interfaz con la red local. Este módulo de Internet empaqueta los segmentos de TCP dentro de los datagramas de Internet y encamina estos datagramas hacia un módulo de Internet de destino por un gateway intermedio. Para poder transmitir el datagrama a través de la red local, se encapsula dentro de un paquete de red local.
El módulo de Internet de destino desenvuelve el segmento del datagrama (después de haber reensamblado el datagrama, si así era necesario) y lo pasa al módulo de TCP de destino.

En la figura 3.7 se ilustra el lugar de TCP en la jerarquía de protocolos. Se espera que TCP sea capaz de soportar protocolos de nivel superior eficientemente.

Un flujo de datos enviado sobre una conexión de TCP se entrega de forma fiable y ordenada al destino.

La transmisión es fiable gracias al uso de números de secuencia y de acuses de recibo. Básicamente, se le asigna un número de secuencia a cada byte de datos. El número de secuencia del primer byte de datos en un segmento se transmite con ese segmento y se le denomina el número de secuencia del segmento. Los segmentos también llevan un número de acuse de recibo que es el número de secuencia del siguiente byte de datos esperado en la transmisión en el sentido inverso.
Cuando el módulo de TCP transmite un segmento conteniendo datos, pone una copia en una cola de retransmisión e inicia un contador de tiempo; si llega el acuse de recibo para esos datos, el segmento se borra de la cola. Si no se recibe el acuse de recibo dentro de un plazo de expiración, el segmento se retransmite.
La llegada del acuse de recibo no garantiza que los datos ya hayan sido entregados al usuario final, sino únicamente que el TCP receptor ha asumido la responsabilidad de hacerlo.
Para controlar el flujo de datos entre los módulos de TCP, se utiliza un mecanismo de flujo de control. El TCP receptor devuelve una "ventana" al TCP emisor. Esta ventana especifica el número de bytes, a contar a partir del número del acuse de recibo, que el TCP receptor está en ese momento preparado para recibir.

Para identificar los distintos flujos de datos que un módulo TCP puede manejar simultáneamente, TCP proporciona un identificador de puerto. Como los identificadores de puertos son seleccionados de forma independiente por cada TCP, puede que no sean únicos. Para disponer de direcciones únicas dentro de cada TCP, se concatena la dirección de Internet que identifica al módulo de TCP con el identificador de puerto para así conformar una dirección de conector ('socket') que será única a largo de todo el conjunto de redes interconectadas.

Una conexión queda completamente especificada por el par de conectores de sus extremos. Una conexión se especifica en la llamada de apertura OPEN con los argumentos de un puerto local y una dirección de conector remoto.

Los procedimientos para establecer conexiones utilizan el indicador de control de sincronización (SYN[33]) e involucran un intercambio de tres mensajes.

Una conexión se inicia ante la coincidencia de un segmento entrante que contiene un SYN. La concordancia de conectores locales y remotos determina cuándo una conexión ha sido iniciada. La conexión queda "establecida" cuando los números de secuencia quedan sincronizados en ambos sentidos.

La finalización de una conexión también involucra el intercambio de segmentos, en este caso transportando un indicador de control FIN.

[33] Synchronize

3.5.4 Formato de la cabecera TCP

Los segmentos de TCP se envían como datagramas de Internet. La cabecera del protocolo de Internet transporta varios campos de información, entre los que se incluyen las direcciones de los host de origen y de destino[34]. Una cabecera de TCP sigue a la cabecera de Internet, aportando información específica del protocolo de TCP. Esta división permite la existencia de otros protocolos de la capa de host distintos de TCP.

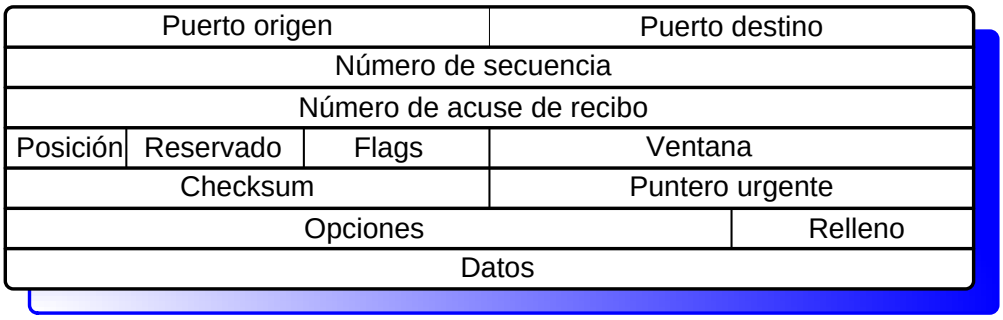

Figura 3.9 Formato de Cabecera TCP

Puerto de origen (16 bits): El número del puerto de origen.

Puerto de destino (16 bits): El número del puerto de destino.

Número de secuencia (32 bits): El número de secuencia del primer byte de datos de este segmento (excepto cuando el indicador SYN esté puesto a uno). Si SYN está puesto a uno es el número de secuencia original (ISN[35]) y, entonces, el primer byte de datos es ISN+1.

Número de acuse de recibo (32 bits): Si el bit de control ACK está puesto a uno, este campo contiene el valor del siguiente número de secuencia que el emisor del segmento espera recibir. Una vez que una conexión queda establecida, este número se envía siempre.

Posición de los datos (4 bits): El número de palabras de 32 bits que ocupa la cabecera de TCP. Este número indica dónde comienzan los datos. La cabecera

---

[34] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, Septiembre de 1981. (N.T. Versión en castellano por P.J. Ponce de León: "Protocolo Internet", Mayo de 1999)

[35] Initial Sequence Number

de TCP (incluso una que lleve opciones) es siempre un número entero de palabras de 32 bits.

Reservado (6 bits): Reservado para uso futuro. Debe valer 0.

Bits de control (6 bits, de izquierda a derecha):

URG: Hace significativo el campo "Puntero urgente"

ACK: Hace significativo el campo "Número de acuse de recibo"

PSH: Función de "Entregar datos inmediatamente" ('push')

RST: Reiniciar ('Reset') la conexión

SYN: Sincronizar ('Synchronize') los números de secuencia

FIN: Últimos datos del emisor

Ventana (16 bits): El número de bytes de datos, a contar a partir del número indicado en el campo de "Número de acuse de recibo", que el emisor de este segmento está dispuesto a aceptar.

Suma de control (16 bits): El campo "Suma de control" es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera y del texto. Si un segmento contiene un número impar de bytes de cabecera y texto, el último byte se rellena con ceros a la derecha para formar una palabra de 16 bits con el propósito de calcular la suma de control. En el cálculo de la suma de control, el propio campo suma de control se considera formado por ceros.
La suma de control también incluye una pseudocabecera de 96 bits prefijada imaginariamente a la cabecera TCP. Esta pseudocabecera contiene la dirección de origen, la dirección de destino, el protocolo, y la longitud del segmento de TCP. Esto proporciona una protección ante segmentos mal encaminados. Esta información es transportada por el protocolo de Internet y es transferida a través de la interfaz TCP/Red en los argumentos o en los resultados de las llamadas de TCP a IP.

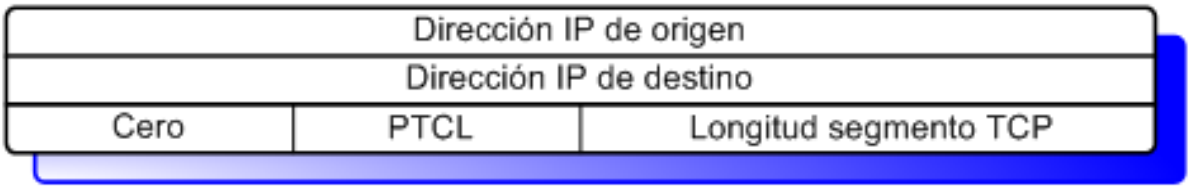

Figura 3.10 Formato de Pseudocabecera TCP

La "longitud TCP" consiste en la suma de la longitud de la cabecera de TCP más la de los datos en bytes (esto no es una cantidad transmitida explícitamente, sino que ha de calcularse), y no incluye los 12 bytes de la pseudo cabecera.

Puntero urgente (16 bits): Este campo indica el valor actual del puntero urgente como un desplazamiento positivo desde el número de secuencia de este segmento. El puntero urgente apunta al número de secuencia del byte al que seguirán los datos urgentes. Este campo es interpretado únicamente si el bit de control URG está establecido a uno.

Opciones (variable): Los campos de opciones pueden ocupar un cierto espacio al final de la cabecera de TCP, pero siempre de una longitud múltiplo de 8 bits. En el cálculo de la suma de control, se incluyen todas las opciones.

Una opción puede empezar en cualquier posición múltiplo de ocho. Existen dos posibilidades para el formato de una opción:
Caso 1: Un byte único con el tipo de opción.
Caso 2: Un byte con el tipo de opción, un byte con la longitud de la opción, y los bytes con los datos propiamente dichos de la opción.

La longitud de la opción tiene en cuenta tanto el byte con el tipo de opción como el propio byte de longitud así como los bytes con los datos de la opción.

Nótese que la lista de opciones puede ser más corta que lo que el campo "Posición de los datos" podría implicar. El contenido de la cabecera más allá de la opción "Fin de la lista de opciones" debe ser un relleno de cabecera (es decir, ceros).

Un módulo de TCP debe implementar todas las opciones.
Las opciones definidas en la actualidad incluyen (donde el tipo se indica en octal):

Tipo	Longitud	Significado
0	——	Fin de la lista de operaciones
1	——	Sin operación
2	4	Tamaño máximo de segmento

Tabla 3.6 Significado de Tipo

3.6 Modbus/TCP (Capa 4)

3.6.1 ¿Por qué Modbus/TCP?

Antes de hacernos esta pregunta se realizó una investigación sobre los protocolos[36] más utilizados en el sector industrial, con el objetivo de conocer el área de aplicación de cada protocolo. Encontramos diferentes protocolos, por ejemplo Fieldbus Foundation, Profibus y HART, que están diseñados para instrumentación de control de procesos. En cambio otros como DeviceNet y SDC están optimizados para los mercados de los dispositivos discretos (on-off) de detectores, actuadores e interruptores, donde el tiempo de respuesta y repetibilidad son factores críticos. Sin embargo el que más se adecuo al campo de aplicación fue MODBUS/TCP[37].

En las aplicaciones industriales, Ethernet es usado en conjunto con la pila de protocolos TCP/IP universalmente aceptada. TCP/IP es el conjunto de protocolos usado en Internet, suministrando un mecanismo de transporte de datos confiable entre máquinas y permitiendo interoperabilidad entre diversas plataformas. Usar TCP/IP sobre Ethernet a nivel de campo en la industria permite tener una verdadera integración con la Intranet corporativa, y de esta forma se ejerce un estricto control sobre la producción.

Estos factores hacen que un protocolo basado en Ethernet TCP/IP sea el más adecuado, pero para su selección se usarán los siguientes criterios:

1- Más difundido en el mercado de medidores de energía.
2- Amplia información técnica para implementación, de preferencia ORG.
3- Facilidad en la implementación.
4- Protocolo adecuado para posteriores proyectos.

1- Más difundido en el mercado de medidores de energía.

Este criterio puede ser solucionado buscando otros medidores de estado sólido en el mercado internacional, y que también implementen comunicación a través de Ethernet. El resultado de la búsqueda se muestra en la Tabla 3.7 donde se

[36] Un protocolo de comunicación define el conjunto de reglas a seguir para llevar a cabo la comunicación, siendo éstas de reconocimiento internacional.
[37] *MODBUS/TCP* protocolo del nivel de Aplicación usado sobre una red Ethernet, utilizando los protocolos TCP/IP. Ver [3][4].

muestra que el protocolo más usado en medidores de energía de estado sólido con comunicación Ethernet es MODBUS/TCP, además de ser muy utilizado en otros medios como MODBUS RTU y ASCII[38].

2- Amplia información técnica para implementación, de preferencia ORG[39]

En efecto MODBUS es una ORG y por lo tanto la información esta al acceso de todo el que desee obtenerla, de forma gratuita. La información puede obtenerse de la dirección www.modbus.org, proporcionando información técnica para la implementación del protocolo.

MEDIDOR	EMPRESA	PROTOCOLO USADO
QUANTUM Q100	Schlumberger Electricity Inc.	Modbus/TCP, DNP3, mini DLMS
DMMS350 DMWH300 SHARK 100T DMMS300+	Electro Industries	Modbus/TCP Modbus RS485 Modbus y DNP3 Modbus RTU y DNP3
Pro-MON	Managin Automation	Modbus RTU
eM200	Energy Tracking	Modbus TCP
Power Logia	Scheider Electric	Modbus TCP
H8920-3	Veris Industries	Modbus TCP
EPM 5200	General Electric	Modbus y DNP 3

Tabla 3.7. Resultado de búsqueda de protocolos más usados en medidores internacionales

3- Facilidad en la implementación.

Para la implementación del protocolo se manejan dos documentos:
MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE [4]
MODBUS APPLICATION PROTOCOL SPECIFICATION [3]

[38] MODBUS RTU y ASCII son modos de transmisión sobre línea serial RS232 o RS485.
[39] ORG es una organización sin fines de lucro.

Ambos documentos son breves, claros y explican el protocolo a profundidad, siendo de gran ayuda para su implementación. Presentan desde las tramas a nivel de bits hasta el flujo grama y sugerencias para la creación de las clases a implementar.
Si bien no son estrictamente una guía paso a paso, pero proveen ayuda que en otros protocolos puede ser inaccesible o muy confusa.

4- Protocolo adecuado para posteriores proyectos.

Modbus es un protocolo que abarca una amplia variedad de medios sobre los cuales puede ser implementado:

- TCP/IP sobre Ethernet.
- Transmisión serial (EIA/TIA-232-E, EIA-422, EIA/TIA-485-A; fibra, radio, etc.)
- Modbus Plus (High speed Token passing network)

Por lo cual su lógica, una vez implementada en un medio, puede servir de base para saltar a otros medios. Y sin tener que empezar de cero, como seria con un protocolo que únicamente trabajo sobre Ethernet. Además, MODBUS es muy utilizado en la industria en diversas aplicaciones, teniendo su fuerte en aplicaciones de control automático.

Con este estudio previo, se optó por utilizar un estándar de instrumentación sobre Ethernet, en este caso *Modbus/TCP*, para realizar la implementación de monitoreo remoto capaz de ser accedida a través de lnternet, usando los protocolos TCP/IP. El protocolo Modbus/TCP es muy difundido por ser abierto, lo cual le permite la comunicación con gran diversidad de elementos industriales; es por eso que es de gran importancia trabajar sobre él, y además debido a que en nuestro medio no se encuentran desarrollos concernientes a este tema.

3.6.2 Especificaciones Modbus/TCP

3.6.2.1 Introducción

Modbus/TCP[40] es un protocolo de comunicación diseñado para permitir a equipo industrial tal como Controladores Lógicos Programables (PLCs), computadores, motores, sensores, medidores y otros tipos de dispositivos físicos de entrada/salida comunicarse sobre una red. Modbus/TCP fue introducido por Schneider Automation como una variante de la familia MODBUS ampliamente

[40] Especificado en los documentos [2][3]

usada, siendo protocolos de comunicación simples y abiertos, destinados para la supervisión y el control de equipo de automatización. Específicamente, el protocolo cubre el uso de mensajes MODBUS en un entorno Intranet o Internet usando los protocolos TCP/IP.

La especificación Modbus/TCP define un estándar inter-operable en el campo de la automatización industrial, el cual es mas sencillo de implementar para cualquier dispositivo que soporta sockets TCP/IP[41].

En el presente trabajo se implementa este protocolo en lenguaje de programación ensamblador, y luego se descarga en el microcontrolador.

3.6.2.2 Concepto

MODBUS/TCP es un protocolo de mensajería que forma parte de la capa de aplicación, posicionado en la capa 4 del modelo TCP/IP (7 del modelo OSI[42]), que provee comunicación cliente/servidor entre dispositivos conectados sobre diferentes tipos de buses de redes. El protocolo MODBUS/TCP define una estructura de mensajería que los dispositivos reconocerán y usaran, en general sobre cualquier tipo de red, pero para nuestro caso sobre una red Ethernet. Dicho protocolo describe el proceso que un dispositivo usa para realizar peticiones hacia otros dispositivos, cómo este responde a las peticiones de otros dispositivos, y como los errores serán detectados y reportados. Básicamente se establece un formato común para la estructura y contenido del campo de mensaje.

Durante la comunicación sobre una red MODBUS, el protocolo determina la forma en que cada dispositivo reconocerá sus direcciones, reconocerá un mensaje direccionado a éste, determinara el tipo de acción a ser tomada, y extraerá cualquier dato u otra información contenida en el mensaje. Si una replica es requerida, el controlador construirá el mensaje replica y lo enviara usando el protocolo MODBUS.

Cuando MODBUS viaja sobre otras redes, los mensajes conteniendo el protocolo MODBUS son embebidos dentro de la trama o estructura de paquete que es usado en la nueva red. Lo cual permite “hablar un mismo idioma” sobre medios que utilizan protocolos diferentes.

3.6.2.3 Modelo cliente-servidor

[41] Los Socket son el nivel básico para realizar una comunicación sobre Ethernet a nivel de software, estableciendo comunicación entre servidor/cliente.
[42] OSI es el *Open Systems Interconnection Reference Model*

El servicio de mensajería MODBUS provee comunicación Cliente/Servidor entre dispositivos conectados sobre una red Ethernet. El modelo cliente/servidor es basado sobre cuatro tipos de mensajes:

- Petición MODBUS
- Confirmación MODBUS
- Indicación MODBUS
- Respuesta MODBUS

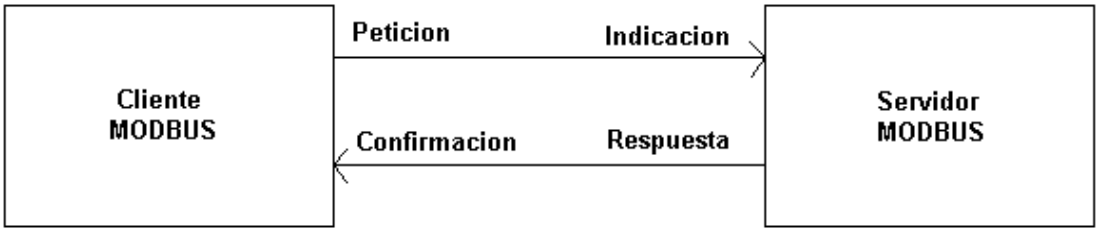

Figura 3.11 Mensajes en modelo cliente/servidor MODBUS/TCP

Petición MODBUS:
Como se muestra en la figura 3.11, es el mensaje enviado sobre la red por el cliente para inicializar una transacción.

Indicación MODBUS:
Es básicamente el mensaje de petición hecho por el cliente pero ahora recibido en el lado del servidor.

Respuesta MODBUS:
Es el mensaje de respuesta enviado por el servidor.

Confirmación MODBUS:
Es el mensaje de respuesta recibido en el lado del cliente.

Este servicio de mensajería MODBUS es utilizado en intercambios de información en tiempo real.

3.6.2.4 Modbus sobre TCP/IP Unidad de Dato de Aplicación (ADU).

Aquí se muestra la forma en que las peticiones y respuestas MODBUS se encapsula cuando son llevadas a través de redes MODBUS TCP/IP.

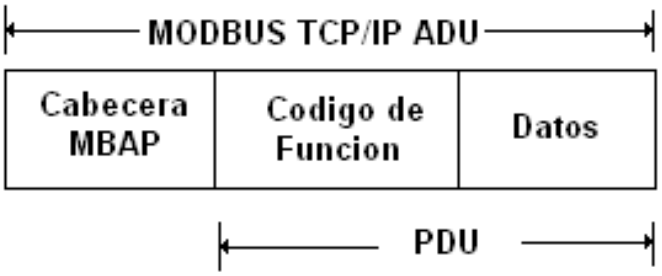

Figura 3.12 Petición/Respuesta del protocolo MODBUS sobre TCP/IP

Como se muestra en la figura 3.12, la trama se divide en tres secciones principales:

Cabecera MBAP.
Sobre TCP/IP una cabecera es utilizada para identificar la Unidad de Dato de Aplicación (ADU). La cual es llamada Cabecera MBAP (Cabecera del protocolo de aplicación MODBUS). La cabecera MODBUS contiene cuatro campos que se describen el la tabla 3.7
La cabecera tiene 7 bites de longitud:

Identificador de transacción
Este es usado en pares de transacciones, el servidor MODBUS copia en la respuesta el identificador de transacción de la petición.

Identificador de Protocolo
El protocolo MODBUS es identificado por el valor cero.

Campo	Longitud	Descripción	Cliente	Servidor
Identificador de Transacción	2 Bites	Identificación de una transacción petición/respuesta en Modbus	Inicializada por el cliente	Recopiado por el servidor desde la petición recibida.
Identificador de protocolo	2 Bites	0 = Protocolo MODBUS	Inicializado por el cliente	Recopiado por el servidor desde la petición recibida.
Longitud	2 Bites	Numero de bites siguientes	Inicializado por el cliente (petición)	Inicializado por el servidor (respuesta)

Identificador de unidad	1 Bite	Identificación de un esclavo remoto conectado sobre una línea serial u otro bus	Inicializado por el cliente	Recopiado por el servidor desde la petición recibida.

Tabla 3.8 Campos de la cabecera MODBUS

Longitud
El campo de longitud es un bite de conteo de los siguientes campos, incluyendo el identificador de unidad y campo de datos.

Identificador de unidad
Este campo es usado para propósitos de ruteo entre sistemas. Este es típicamente usado para comunicar a un esclavo MODBUS o a un MODBUS+ en línea serial a través de un gateway entre una red Ethernet TCP-IP y una línea serial MODBUS. Este bite es establecido por el cliente MODBUS en la petición y debe ser retornado con el mismo valor en la respuesta por el servidor.

CATEGORÍAS DE LOS CÓDIGOS DE FUNCIÓN EN MODBUS.

Existen tres tipos de código de función MODBUS, estos son:

1 - Códigos de función públicos

Son códigos de función bien definidos
Garantizan ser únicos
Validados por la comunidad MODBUS-IDA
Documentación publica

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**	**6**	**14**	6.14
			Write File record	**21**	**6**	**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Slave ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19

Tabla 3.9 Definición de los códigos de función públicos

La tabla 3.9 muestra los códigos de función públicos existentes y sus aplicaciones, entre los cuales en el presente trabajo de graduación se implementaron los códigos de función 4 (Lectura de registros de entrada) y 16 (Escritura de múltiples registros).

2 - Códigos de función definidos por el usuario

Existen dos rangos de códigos de función definidos por el usuario, del 65 al 72 y desde el 100 al 110 decimal.

El usuario puede seleccionar e implementar un código de función que no es soportado por la especificación.

No ser garantiza que el uso del código de función seleccionada sea único.

3 - Códigos de función reservados

Códigos de función actualmente usados por algunas compañías para sus productos y que no están disponibles para uso publico.

La figura 3.13 muestra los rangos para cada código de función.

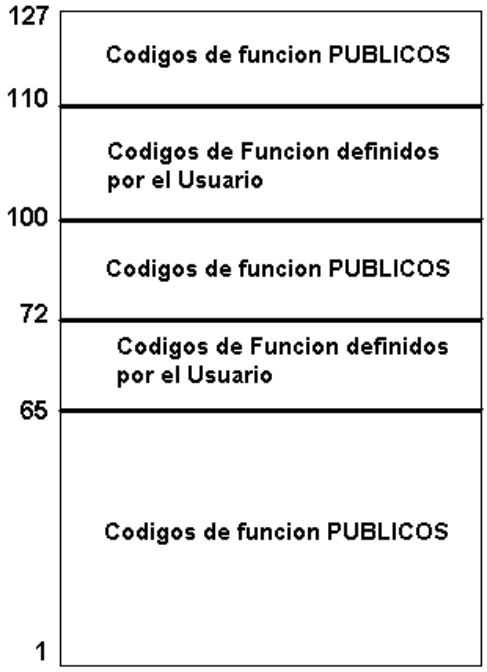

Figura 3.13 Categorías de los Códigos de Función.

3.6.2.5 Modelo de la arquitectura de componentes Modbus

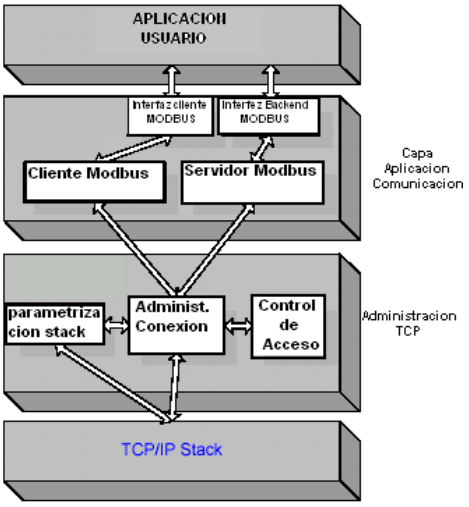

Figura 3.14 Arquitectura conceptual del servicio de mensajera MODBUS

CAPA DE APLICACIÓN DE COMUNICACIÓN

Un dispositivo MODBUS puede proveer una interfaz cliente/ o servidor. La interfaz backend MODBUS puede ser proveída para permitir indirectamente el acceso al objeto de aplicación de usuario. Cuatro áreas pueden componer esta interfaz: entradas discretas, salidas discretas (coils), registros de entrada y registros de salida.

Tablas primarias	Tipos de objetos	De tipo	Comentarios
Entradas discretas	Único Bit	Solo lectura	Este tipo de dato puede ser proveído por un sistema Entrada/Salida.
Salidas discretas (coils)	Único Bit	Lectura-Escritura	Este tipo de dato puede ser alterable por un programa aplicación.
Entrada de registros	Palabra de 16 bits	Solo lectura	Este tipo de dato puede ser proveído por un sistema entrada/salida
Salida de registros	Palabra de 16 bits	Lectura-Escritura	Este tipo de dato puede ser alterable por un programa de aplicación.

Tabla 3.10 Organización de la memoria

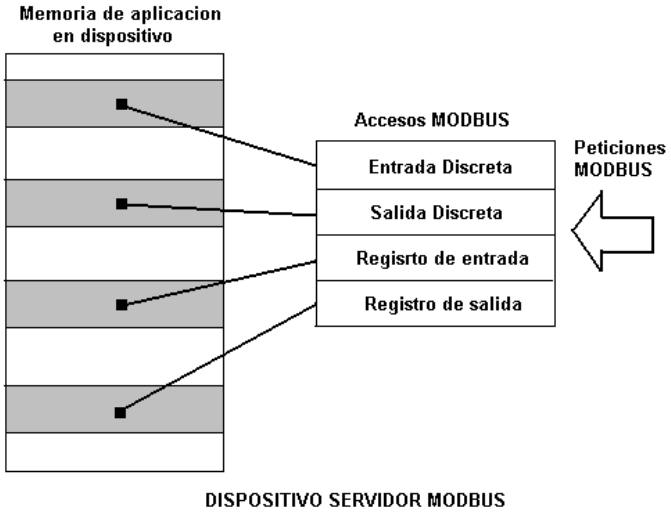

Figura 3.15 Modelo de datos MODBUS con bloques separados

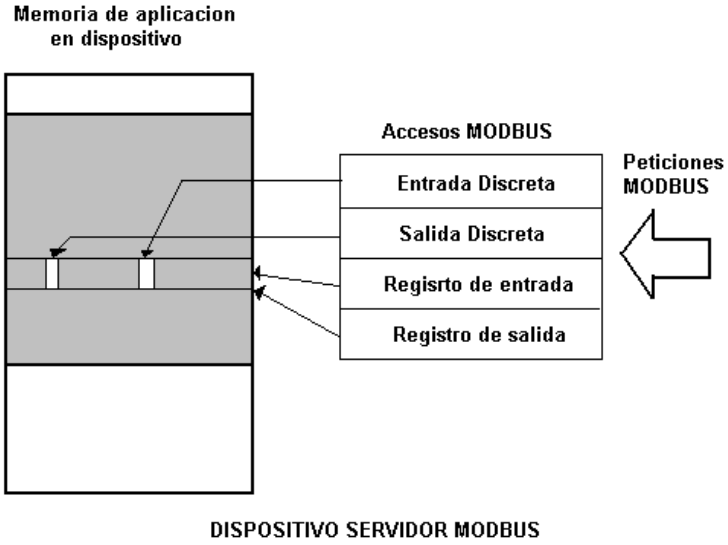

Figura 3.16 Modelo de datos MODBUS con únicamente 1 bloque.

Cliente MODBUS

El cliente MODBUS le permite a la aplicación usuario explícitamente controlar el intercambio de información con un dispositivo remoto. El cliente MODBUS construye una petición MODBUS de los parámetros contenidos en una demanda enviada por la aplicación usuario a la interfaz cliente MODBUS.

El cliente MODBUS usa una transacción MODBUS cuya administración incluya espera y procesado de una confirmación MODBUS.

Interfaz cliente MODBUS
La interfaz cliente MODBUS provee una interfaz que le permite a la aplicación usuario construir la petición para varios servicios MODBUS incluyendo acceso a objetos aplicación MODBUS.

Servidor MODBUS
Sobre la recepción de una petición MODBUS este modulo activa una acción local para leer, escribir o llevar a cabo otras acciones. El procesamiento de estas acciones es hecho totalmente transparente a la aplicación de programador. La principal función del servidor MODBUS es esperar por una petición MODBUS sobre el puerto TCP 502, para tratar esta petición y entonces construir una respuesta MODBUS dependiendo del contexto del dispositivo.

Interfaz Backend MODBUS
La interfaz MODBUS backend es una interfaz desde el servidor MODBUS a la aplicación usuario en el que los objeto aplicación están definidos.

CAPA DE ADMINISTRACIÓN TCP
Una de las principales funciones del servicio de mensajería es manejar el establecimiento de la comunicación y terminarlo, manejar el flujo de datos sobre las conexiones TCP.

Administración de conexión.
Una comunicación entre un modulo cliente y un servidor MODBUS requiere el uso de un modulo de administración de conexión TCP. Este es cargado para manejar globalmente los mensajes de las conexiones TCP.

Dos posibilidades son propuestas para la administración de conexiones. En una la aplicación usuario maneja por si misma las conexiones TCP y en la otra la administración de las conexiones es totalmente hecha por el modulo de administración y por lo tanto es transparente a la aplicación usuario. La última solución implica menos flexibilidad.

El puerto TCP 502 es reservado para comunicación MODBUS. Es importante notar que aun si otro puerto TCP en el servidor es configurado para servicios MODBUS en ciertas aplicaciones, el puerto TCP 502 debe mantenerse disponible en adición a cualquier aplicación con puerto especifico.

CAPA TCP/IP STACK

El stack TCP/IP puede ser configurado en orden de adaptar el control de flujo de datos, el administrador de direcciones y el administrador de conexión a diferentes circunstancias específicas a un producto o a un sistema.

3.6.2.6 Administrador de Conexiones TCP

Modulo de administración de conexión.

Descripción general.

Una comunicación MODBUS requiere el establecimiento de una conexión TCP entre un cliente y un servidor.
El establecimiento de la conexión puede ser activada explícitamente por el modulo de aplicación usuario o automáticamente por modulo de administración de conexión TCP.
En el primer caso una interfaz de un programa aplicación es proveído en el modulo de aplicación de usuario para manejar completamente la conexión. Esta solución provee flexibilidad para el programa aplicación pero requiere una buena experiencia sobre el mecanismo TCP.
En el segundo caso el administrador de conexión TCP es completamente oculto a la aplicación usuario que únicamente envía y recibe mensajes MODBUS. El modulo administrador de conexión TCP es el encargado del establecimiento de una nueva conexión TCP cuando ésta es requerida.

Reglas de implementación:

Sin requerimientos explícitos del usuario, es recomendable implementar el administrador automático de conexión TCP.
Es recomendable mantener la conexión TCP abierta con un dispositivo remoto y no abrir y cerrar ésta para cada transacción MODBUS/TCP.
Es recomendable para un cliente MODBUS abrir un mínimo de conexiones TCP con un servidor remoto MODBUS (con la misma dirección IP). Una conexión por aplicación puede ser una buena elección.
Muchas transacciones MODBUS pueden ser activadas simultáneamente sobre la misma conexión TCP.
Nota: Si esto es hecho entonces el identificador de transacción MODBUS debe ser usado para únicamente identificar los pares, petición y respuesta.
En el caso de una comunicación bidireccional entre dos entidades remotas MODBUS (cada una de ellas es cliente y servidor), es necesario abrir separadamente conexiones para el flujo de datos del cliente y para el flujo de datos del servidor.
Una trama TCP debe transportar únicamente un ADU MODBUS. De nuevo se recomienda enviar múltiples peticiones MODBUS o respuestas sobre la misma TCP PDU.

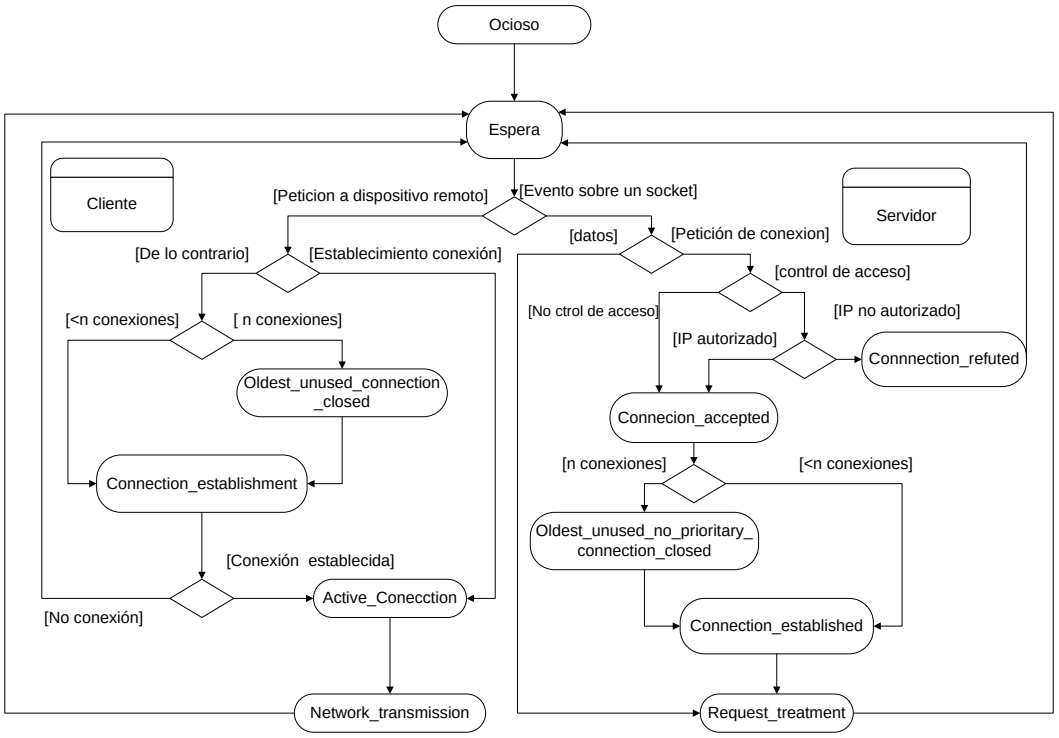

Figura 3.17 Diagrama de actividades del Administrador de conexión TCP

Administrador de conexión explicito TCP

El modulo de aplicación de usurario es el encargado de administrar todas las conexiones TCP: activas y enlaces pasivos, finalizaciones de conexiones, etc. Esta administración es hecha para todas la comunicaciones MODBUS entre un cliente y un servidor.

Administrador de conexión automático TCP

El administrador de conexión es totalmente transparente para el modulo de aplicación usuario. El modulo de administración de conexión puede aceptar un numero suficiente de clientes y conexiones a servidores.

Una conexión con un equipo remoto es establecida con el primer paquete recibido desde un cliente remoto o desde la aplicación de usuario local. Esta conexión será cerrada si una señal de terminación llega desde la red o se decide localmente en el dispositivo. Sobre la recepción de una petición de conexión, las opciones de control de acceso pueden ser usadas para prohibir el acceso a clientes no autorizados.

3.6.2.7 Descripción del Administrador de Conexión

Establecimiento de conexión:
El servicio de mensajería MODBUS debe proveer un socket escuchando por el puerto 502, que permite aceptar nuevas conexiones e intercambiar datos con otros dispositivos.
Cuando el servicio de mensajera necesita intercambiar datos con un servidor remoto, este debe abrir una nueva conexión cliente con un puerto remoto 502 con el objetivo de intercambiar datos con este dispositivo distante. El puerto local puede ser mayor que 1024 y diferente para cada conexión cliente.

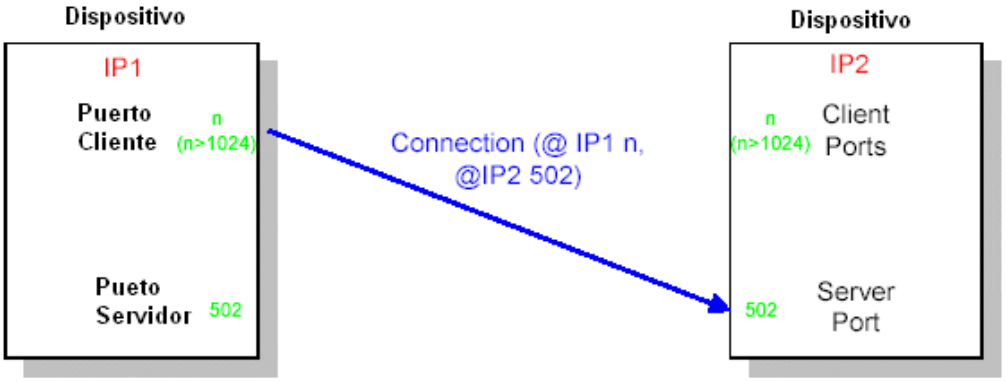

Figura 3.18 Establecimiento de conexión TCP MODBUS.

Si el número de clientes y conexiones de servidores es mayor que el número de conexiones autorizadas, la más antigua conexión sin uso es cerrada. El mecanismo de control de acceso puede ser activado para chequear si la dirección IP del cliente remoto es autorizada. Si no es así la nueva conexión es rechazada.

Transferencia de datos MODBUS
Una petición MODBUS será enviada por la conexión TCP actualmente abierta. La dirección IP del dispositivo remoto es usada para encontrar la conexión TCP. En caso de múltiples conexiones TCP abiertas con el mismo remoto, una conexión tiene que ser seleccionada para enviar el mensaje MODBUS, diferentes criterios de selección pueden ser usados tales como el más antiguo, el primero. La conexión debe mantenerse abierta durante toda la comunicación MODBUS. Como se mencionara mas adelante, un cliente puede inicializar muchas transacciones MODBUS con un servidor sin esperar el final de la previa.

Cerrado de conexión
Cuando la comunicación MODBUS ha finalizado entre un cliente y un servidor, el cliente tiene que inicializar un cierre de conexión de la conexión usada para esta comunicación.

3.6.2.8 Uso del Stack TCP/IP

El stack TCP/IP provee una interfaz para manejar conexiones, para enviar y recibir datos, y también para hacer algunas parameterizaciones con el fin de adaptar el comportamiento del stack al dispositivo o restricciones del sistema.

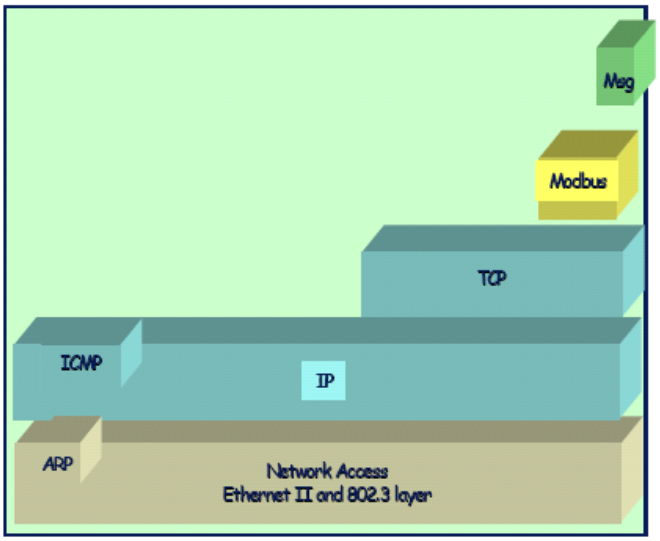

Figura 3.19 Protocolos usados para implementar la comunicación

Uso de la interfaz Socket BSD

NOTA: Algunos stack TCP/IP proponen otros tipos de interfaz para desarrollar el problema. Un cliente MODBUS o servidor puede usar otra interfaz específica.

Un socket es un punto final de comunicación. Este es el bloque básico de construcción para la comunicación. Una comunicación MODBUS es ejecutada enviando y recibiendo datos a través de sockets. La librería TCP/IP provee únicamente flujos a través de socket usando TCP y proveyendo una conexión basada en un servicio de comunicación.
Los socket son creados utilizando la función **socket**. Un número de socket es retornado, que es usado por el creador para acceder el socket. Los sockets son creados sin direcciones (dirección IP y número de puerto). Hasta que un puerto es asociado a un socket, este no puede ser usado para recibir datos.
La función **bind ( )** es usada para asociar un numero de puerto a un socket. La función bind ( ) crea un enlace entre el socket y el numero de puerto especificado.

Cuando se desea inicializar una conexión, el cliente debe llamar a la función **connect ( )** especificando el numero de socket, la dirección IP remota y el numero de puerto que escucha remotamente.
Con el objetivo de completar una conexión, el servidor debe usar la función **accept ( )** especificando el numero del socket que fue especificado en la llamada anterior a **listen ( )** (establecimiento de conexión pasiva). Un nuevo socket es creado con las mismas propiedades que el inicial. Este nuevo socket es conectado al socket cliente, y su número es retornado al servidor. El socket inicial esta por lo tanto libre para otros clientes que deseen conectarse con el servidor.
Después del establecimiento de la conexión TCP los datos pueden ser transferidos. Las funciones **send ( )** y **recv ( )** son designadas específicamente para ser usadas con sockets que ya han establecido conexión.
La función **setsockopt ( )** le permite al creador del socket asociar opciones con un socket. Estas opciones modifican el comportamiento del socket.
La función **select ( )** le permite al programador probar eventos sobre todos los sockets.
La función **shutdown ( )** le permite a un usuario de socket deshabilitar send ( ) y/o receive ( ) sobre un socket.
Una vez un socket no es necesario, el descriptor del socket puede ser descartado usando la función **close ( )**.

La figura 3.20 describe una comunicación completa MODBUS que describe los intercambios hechos entre un cliente y un servidor. El cliente establece la conexión y envía 3 peticiones MODBUS a el servidor sin esperara la respuesta de el primero. Después de recibir todas las respuestas el cliente cierra la correspondiente conexión.

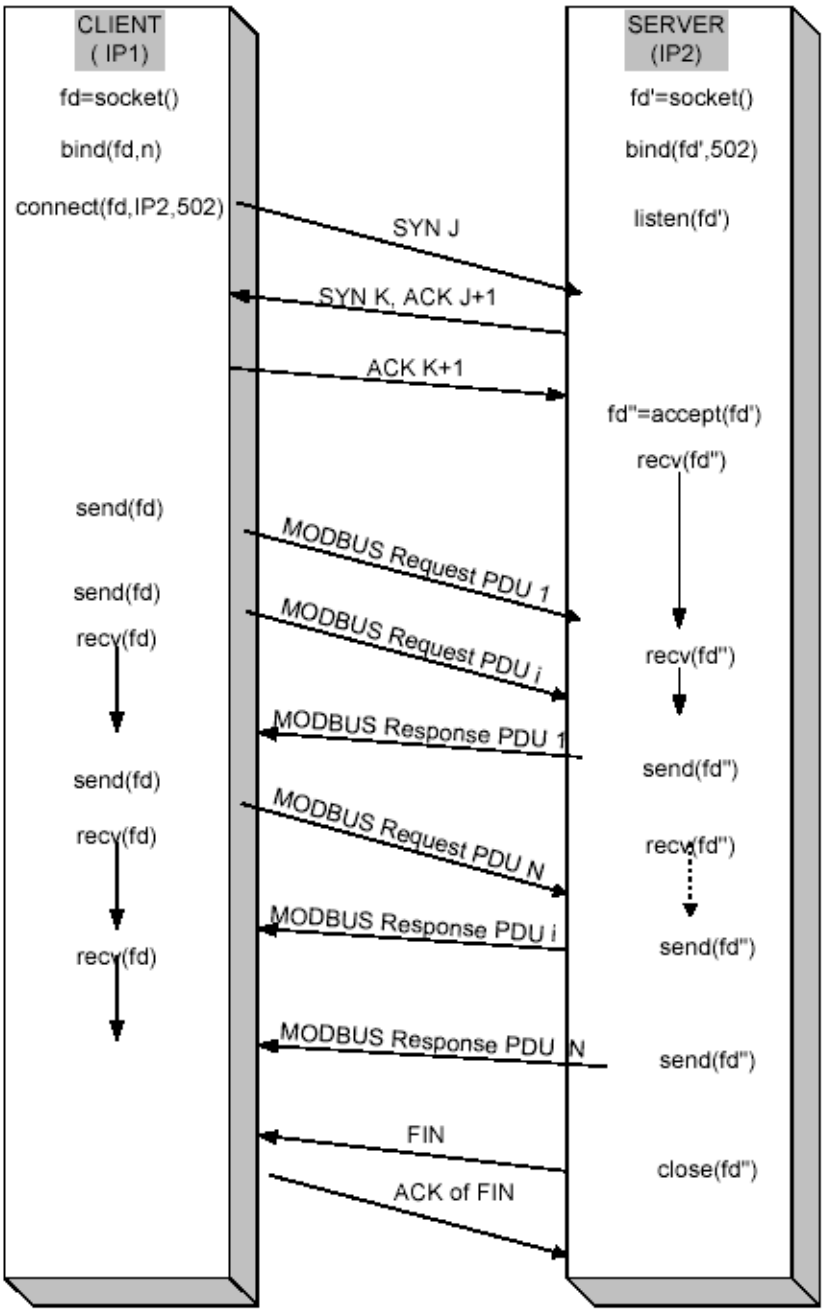

Figura 3.20 Intercambio MODBUS

A pesar de que se describe el funcionamiento del stack TCP/IP BSD, no se ha utilizado ninguna librería para el control TCP en el desarrollo de la tarjeta ETHERNET-RS232. Sin embargo se ha utilizado para el desarrollo de los protocolos TCP/IP en ensamblador sobre el PIC16F877A.

3.6.2.9 Capa de Aplicación de Comunicación

3.6.2.9.1 Cliente Modbus

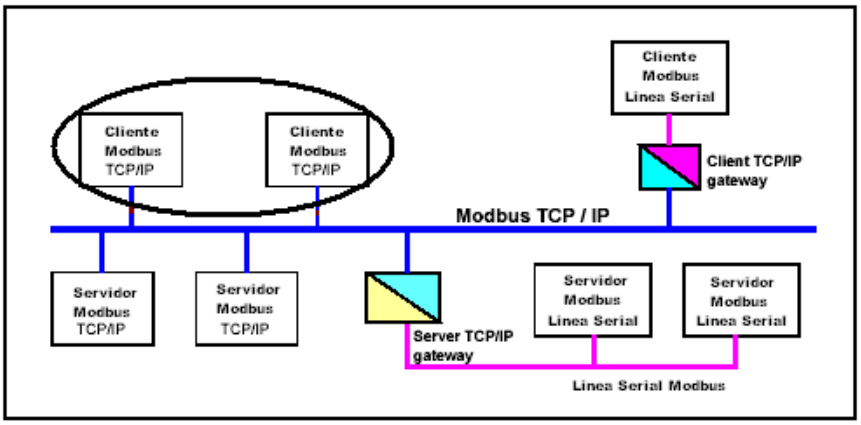

Figura 3.21 Unidad cliente MODBUS

Diseño del cliente MODBUS

La definición del protocolo MODBUS/TCP permite un simple diseño de un cliente. El siguiente diagrama de actividades describe los tratamiento principales que son procesados por un cliente para enviar una petición MODBUS y tratar una respuesta MODBUS.

Un cliente MODBUS puede recibir tres eventos:

Una nueva demanda desde la aplicación usuario para enviar una petición, en este caso una petición MODBUS será encodificada y enviada sobre la red usando el servicio componente administrador TCP. La capa inferior (modulo administrador TCP) puede retornar un error debido a un error en la conexión TCP, o algún otro error.

Una respuesta desde el administrador TCP, en este caso el cliente el cliente tiene que analizar el contenido de la respuesta, y enviar una confirmación a la aplicación de usuario.

La expiación de un Time-out debido a una falta de respuesta. Un nuevo intento puede ser enviado sobre la red o una confirmación negativa puede ser enviada a la Aplicación Usuario.

NOTA: Estos re-envíos son inicializados por el cliente MODBUS, algunos otros re-envíos pueden también ser hechos por la capa TCP en caso de ausencia de acknowledgement TCP.

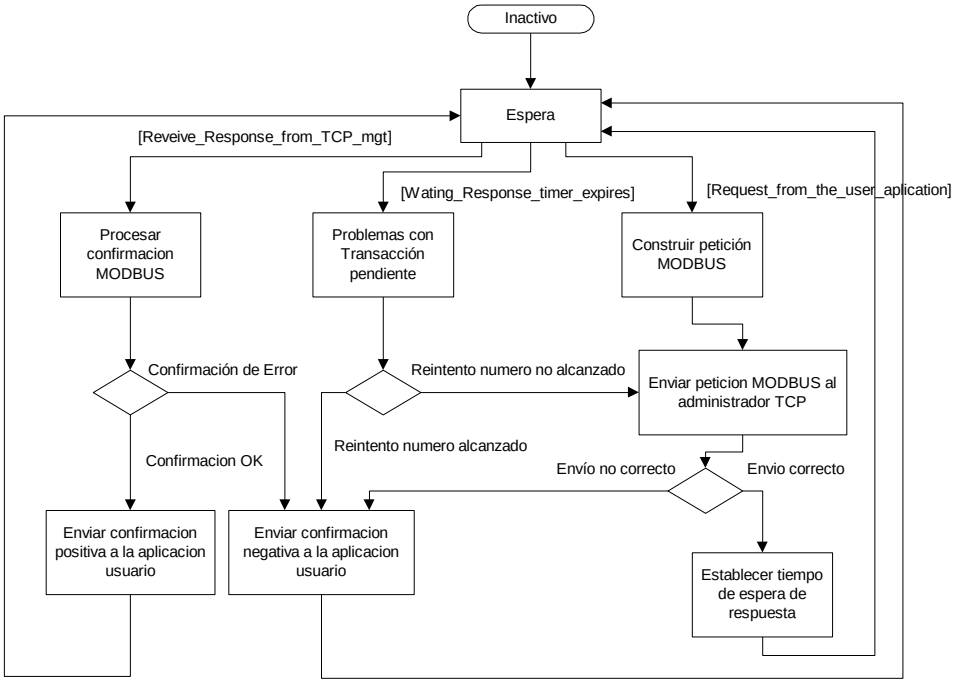

Figura 3.22 Diagrama de actividad del cliente MODBUS

Construcción de una petición MODBUS.

Seguido de la recepción de una demanda desde la aplicación usuario, el cliente tiene que construir una petición MODBUS y enviarla al administrador TCP.

Construir la petición MODBUS puede ser dividido en varias sub-tareas:

La creación de la instancia de una transacción MODBUS que le permita al cliente memorizar toda la información requerida para luego asociar la respuesta a la petición y enviar la confirmación a la aplicación usuario.

El codificado de la petición MODBUS (PDU+cabecera MPAB). La aplicación usuario que inicializa la demanda tiene que proveer toda la información requerida que permita al cliente codificar la petición. El PDU MODBUS es codificado de acuerdo a el "MODBUS Application Protocol Specification [1]". (Código de función MODBUS, parámetros asociados y datos de aplicación). Todos los campos de la cabecera MBAP son llenados. Entonces el ADU de la petición MODBUS es construido prefijando el PDU con la cabecera MBAP.

Los siguientes diagramas de actividades describen más profundamente la fase de construcción de la petición MODBUS.

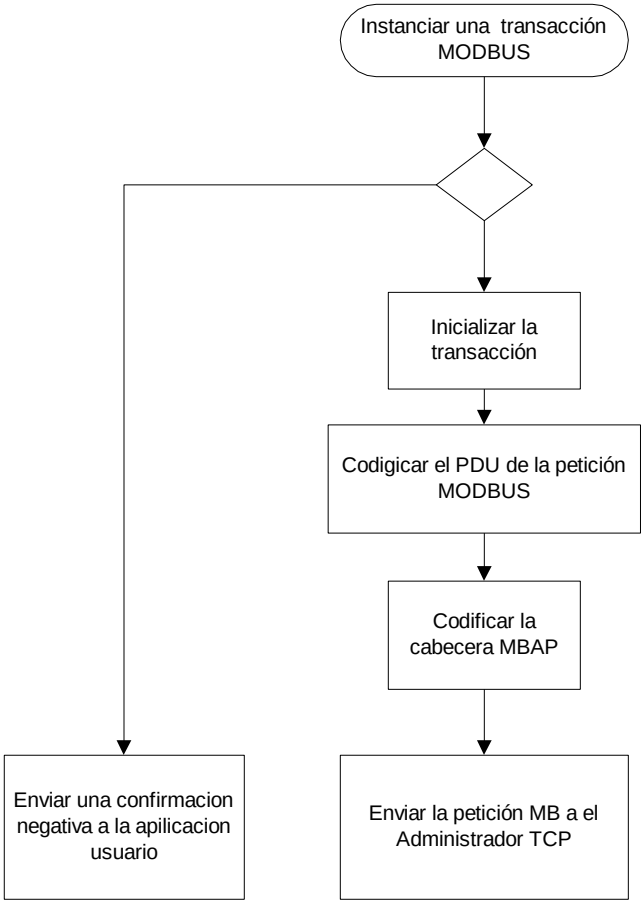

Figura 3.23 Diagrama de actividad para la construcción de una Petición

El siguiente ejemplo describe la codificación del ADU en la petición MODBUS para leer el registro #5 en un servidor remoto:

	DESCRIPCION	TAMAÑO	EJEMPLO
Cabecera MBAP	Identificador de transacción Hi	1	0x15
	Identificador de transacción Lo	1	0x01
	Identificador de protocolo	2	0x0000
	Longitud	2	0x0006
	Identificador de unidad	1	0xFF
Petición MODBUS	Código de función	1	0x03
	Dirección de inicio	2	0x0004
	Cantidad de registros	2	0x0001

Tabla 3.11 Codificación del ADU en la petición MODBUS

Identificador de transacción:
El identificador de transacción es usado para asociar la respuesta futura con la petición. Por lo cual, a un tiempo, sobre una conexión TCP, este identificador debe ser único. Existen diferentes maneras de usar el identificador de transacción:
Por ejemplo, este puede ser usado como un simple “numero de secuencia TCP” con un contador que es incrementado por cada petición.
Este puede también ser juiciosamente usado como un índice elegante o puntero para identificar un contexto de transacción con el objeto de memorizar el servidor remoto actual la petición MODBUS pendiente.

Normalmente, un cliente MODBUS sobre una línea serial debe enviar una petición a la vez. Esto significa que el cliente debe esperara por la respuesta a la primera petición antes de enviar una segunda petición. Sobre TCP/MODBUS, muchas peticiones pueden ser enviadas sin esperar por una confirmación al mismo servidor. El gateway MODBUS/TCP a MODBUS línea serial es el encargado de asegurar la compatibilidad entre estos dos comportamientos.

El número de peticiones aceptadas por un servidor depende sobre su capacidad en término de número de recursos y tamaño de la ventana TCP. En el mismo sentido el número de transacciones inicializadas simultáneamente por un cliente depende también de su capacidad de recursos. Este parámetro de implementación es llamado “NumberMaxOfClientTransaction” y debe ser descrito como una de las características del cliente. Dependiendo del tipo de dispositivo este parámetro puede tomar un valor de 1 a 16.

Identificador de unidad
Este campo es usado para propósitos de ruteo cuando se direcciona un dispositivo sobre una subred MODBUS o línea serial MODBUS+. En este caso, el “identificador de unidad” porta la dirección del esclavo MODBUS del dispositivo remoto:
Si el servidor MODBUS es conectado a una subred MODBUS+ o línea serial MODBUS y es diseccionado a través de un bridge o gateway, el identificador de unidad MODBUS es necesario para identificar el dispositivo esclavo conectado sobre la subred atrás del bridge o gateway. La dirección de IP destino identifica al bridge adecuado, quien a su vez usa el identificador de unidad MODBUS para enviar la petición a el dispositivo esclavo correcto.
La dirección del dispositivo esclavo MODBUS sobre línea serial son asignados de 1 a 247 (decimal). Dirección 0 es usada como dirección broadcast.

Sobre TCP/IP, el servidor MODBUS es diseccionado usando su dirección IP; por lo cual, el identificador de unidad MODBUS no es usado. El valor 0xFF debe ser usado.
Cuando se direcciona un servidor MODBUS conectado directamente a una red TCP/IP. Es recomendable no usar una dirección de esclavo MODBUS significativa en el campo “identificador de unidad”. En el evento de una

relocalización de la dirección IP dentro de un sistema automático y si una dirección IP previamente asignada a un servidor MODBUS es entonces asignado a un gateway, usando una dirección de esclavo significativa puede causar problemas debido a un mal ruteo por el gateway.
Usando una dirección de esclavo no significativa, el gateway simplemente descartará el PDU MODBUS sin problemas. 0xFF es recomendado para el "identificador de unidad" con un valor no significante.

NOTA: El valor 0 es también aceptado para comunicación directa a un dispositivo MODBUS/TCP.

Confirmación de procesos MODBUS
Cuando una trama de respuesta es recibida sobre una conexión TCP, el portador del identificador de transacción en la cabecera MBAP es usada para asociar la respuesta con la petición original previamente enviada sobre la conexión TCP.
Si el identificador de transacción no se refiere a ninguna transacción pendiente MODBUS, la respuesta debe ser descartada.
Si el identificador de transacción se refiere a una transacción pendiente MODBUS, la respuesta debe ser analizada con el objeto de enviar una confirmación MODBUS a la aplicación usuario. (confirmación positiva o negativa)
Analizar la respuesta consiste en verificar la cabecera MBAP y el PDU de la respuesta MODBUS:

Cabecera MBAP
Después de la verificación del identificador de protocolo que debe ser 0x0000, la longitud proporciona el tamaño de la respuesta MODBUS.
Si la respuesta viene desde un dispositivo servidor MODBUS directamente conectado a la red TCP/IP, la identificación de conexión TCP es suficiente para identificar sin ambigüedad el servidor remoto. Por lo tanto, el portador del identificador de unidad en la cabecera MBAP no es significante (valor 0XFF) y debe ser descargado.
Si el servidor remoto es conectado a una subred de línea serial y la respuesta viene desde un bridge, un router o un gateway, entonces el identificador de unidad (!=0xFF) identifica el servidor remoto MODBUS que ha originalmente enviado la respuesta.

PDU de respuesta MODBUS
El código de función debe ser verificado y el formato de respuesta MODBUS analizado de acuerdo al protocolo de aplicación MODBUS:
Si el código de función es el mismo que el usado en la petición, y si el formato de respuesta es el correcto, entonces la respuesta MODBUS es proporcionada a la aplicación usuario como una **confirmación positiva**.
Si el código de función es un código de excepción MODBUS (Código de función +80H), la respuesta de excepción MODBUS es dada a la aplicación de usuario como una **confirmación positiva**.

Si el código función es diferente de el usado en la petición (=código función no esperado), o si el formato de la respuesta es incorrecta, entonces un error es señalizado a la aplicación usuario usando una **confirmación negativa**.
El diagrama de actividad de la figura 3.24 describe, más profundamente el proceso de la fase de confirmación.

Administrador del time-out
Deliberadamente no existen especificaciones del tiempo de respuesta requerido para una transacción sobre MODBUS/TCP.
Esto es debido a que MODBUS/TCP se espera que sea usado en una amplia variedad posible de situaciones de comunicación, desde escaneos I/O sujeto a tiempo en milisegundos hasta largas distancias de enlaces de radio con retardos de muchos segundos.
Desde una perspectiva de cliente, el time-out debe tomar en cuenta el retardo esperado en el transporte a través de la red, para determinar un tiempo de respuesta “razonable”. Tal retardo de transporte debe ser en milisegundos para una red ruteada Ethernet o cientos de milisegundos para una conexión de red WAN.

Hay que tener en mente que todo tiempo “time-out” usado en un cliente para inicializar un re-envió en la aplicación debe ser mayor que el tiempo de respuesta “rasonable” máximo esperado. Si esto no es seguido, será un causa potencial para congestión potencial sobe el dispositivito objetivo o sobre la red, que puede por lo cual causar muchos errores. Esto es una situación que siempre debe ser evitada.

Por lo cual en la práctica, el time-out del cliente usado en aplicaciones de alto desempeño son siempre dependientes de la topología de la red y el desempeño esperado del cliente. Aplicaciones que no son críticas en tiempo pueden a menudo permitir valores de time-out por defecto en TCP, que reportaran falla de comunicación después de varios segundos sobre varias plataformas.

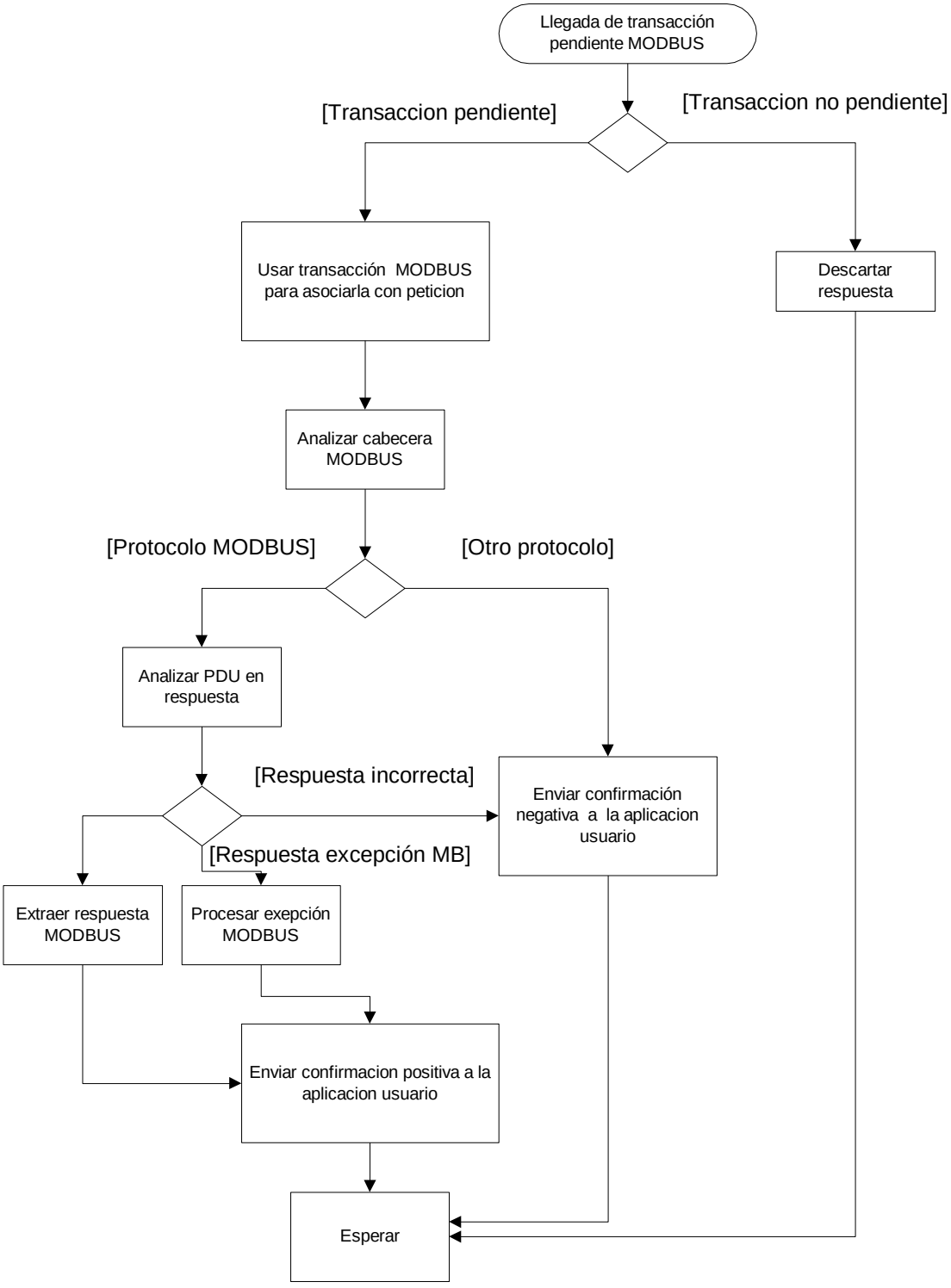

Figura 3.24 Análisis del PDU

3.6.2.9.2 Servidor Modbus

El rol de un servidor MODBUS es proveer acceso al objeto aplicación y servicios al cliente remoto MODBUS.
Diferentes tipos de acceso pueden ser proveídos dependiendo de la aplicación de usuario:
Acceso simple para obtener y establecer atributos de objetos aplicación.
Acceso avanzado con el objeto disparar servicios específicos de aplicación.

El servidor MODBUS tiene:

Trazar objetos aplicación hacia objetos MODBUS que se puedan leer y escribir, con el propósito de obtener o establecer atributos de los objetos aplicación.
Proveer un camino para servicios “trigger” sobre objetos aplicación.

En tiempo de corrido el servidor MODBUS tiene que analizar una petición MODBUS, para procesar la acción requerida, y enviar de regreso una respuesta MODBUS.

Diseño del servidor MODBUS
El diseño del servidor MODBUS depende de dos factores:
El tipo de acceso a el objeto aplicación (acceso simple a atributos o acceso avanzado a los servicios)
El tipo de interacción entre el servidor MODBUS y la aplicación usuario (síncrona o asíncrona).
El siguiente diagrama de actividad describe el tratamiento principal que es procesado por el servidor para obtener una petición MODBUS desde el administrador TCP, entonces analizar la petición, para procesar la acción requerida, y enviar de regreso una respuesta MODBUS.

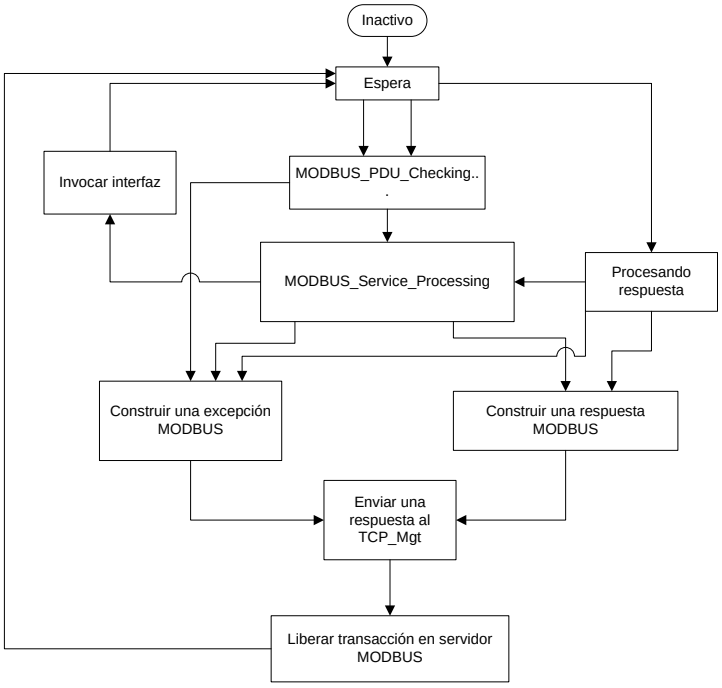

Figura 3.25 Lógica en servidor

Como se muestra en el diagrama de actividad previo:
Algunos servicios pueden ser procesados inmediatamente por el propio servidor MODBUS, sin iteración directa con la aplicación de usuario.
Algunos servicios pueden requerir la iteración explicita con la aplicación usuario a ser procesada.
Algunos otros servicios avanzados requieren invocar una interfase específica llamada servicio MODBUS Back End. Por ejemplo, un servicio de aplicación usuario puede ser activado usando una secuencia de varias transacciones de petición/respuesta de acuerdo al nivel de protocolo de la aplicación usuario. El servicio Back End es responsable por el correcto procesamiento de todas las transacciones individuales MODBUS con el objeto de ejecutar el servicio global de aplicación de usuario.
El servidor MODBUS puede aceptar servir simultáneamente varias peticiones MODBUS. El número máximo de peticiones simultáneas MODBUS que el servidor puede aceptar es una de las principales características del servidor MODBUS. Este numero del diseño del servidor y su capacidad de procesamiento y memoria. Este paramento de implementación es llamado "NumberMaxOfServerTransaction" y debe ser descrita como una de las características del servidor MODBUS.

El comportamiento y el desempeño del servidor MODBUS son significativamente afectados por el parámetro "NumberMaxOfTransaction". Particularmente, es importante notar que el manejo del número de transacciones concurrentes

MODBUS puede afectar el tiempo de respuesta de una petición MODBUS por el servidor.

Chequeo del PDU MODBUS.
El siguiente diagrama describe la actividad de chequeo del PDU MODBUS:

La función de chequeo del PDU MODBUS consiste primero en pasar la cabecera MBAP. El campo identificador de protocolo procede a ser chequeado:
Si este es diferente del tipo de protocolo MODBUS, la indicación es simplemente descartada.
Si este es correcto (= tipo de protocolo MODBUS; valor 0x00), una transacción MODBUS es instanciada.
El numero máximo de transacciones MODBUS que el servidor puede instanciar es definido por el parámetro “NumberMaxOfTransaction”.
En caso de no tener más transacciones disponibles, el servidor construye una respuesta de excepción MODBUS (código de excepción 6: Servidor trabajando)

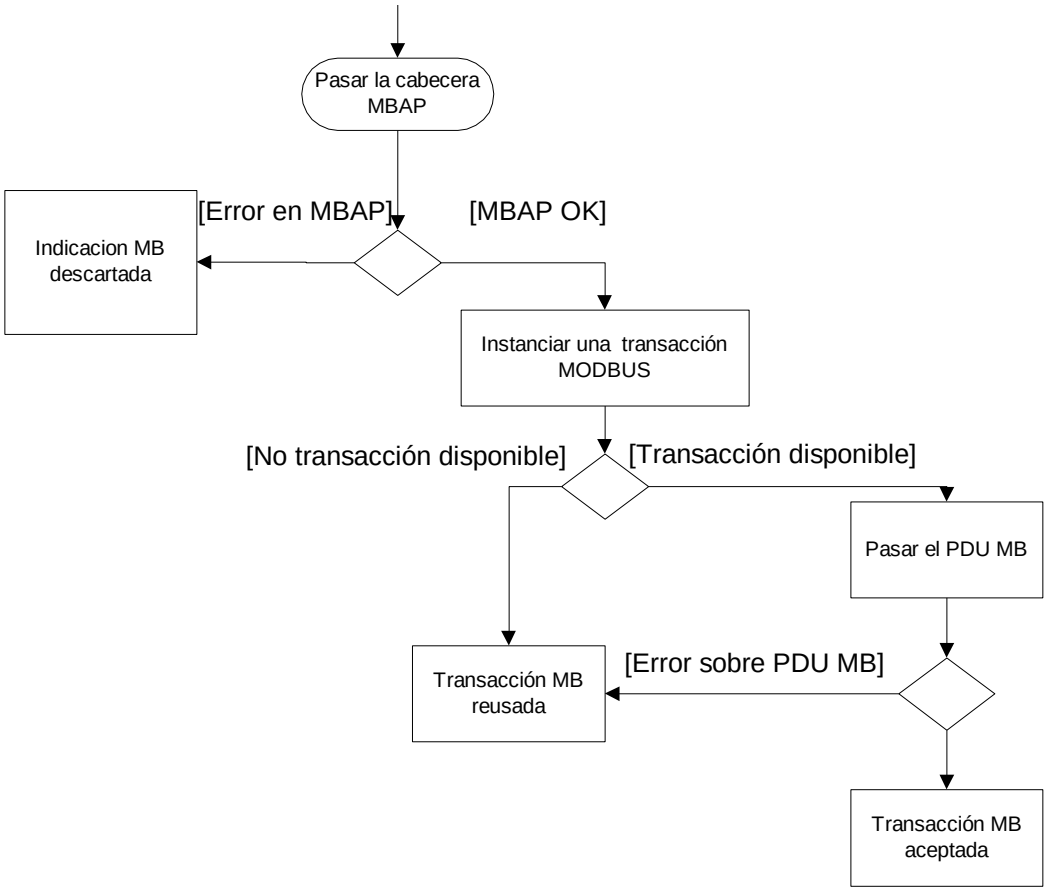

Figura 3.26 Diagrama de actividad para el chequeo del PDU MODBUS

Si una transacción MODBUS esta disponible, esta es inicializada con el objeto de almacenar la siguiente información:
El identificador de conexión TCP usado para enviar la indicación (dada por el administrador TCP)
El ID de transacción MODBUS
El identificador de unidad MODBUS (obtenido en la cabecera MBAP)

Entonces el PDU MODBUS es pasado. El código de función es controlado primero:
En caso de invalidación, una respuesta de excepción MODBUS es construida (Código de excepción 1: Función Invalida)
Si el código de función es aceptado, el servidor inicializa la actividad “Servicio de procesado MODBUS”.

Servicio de procesado MODBUS
El procesamiento del servicio MODBUS requerido puede ser hecho en diferentes caminos dependiendo de la arquitectura del software y hardware como se describe en los siguientes ejemplos:
Dentro de un dispositivo compacto o una arquitectura “mono-thread” donde el servidor MODBUS puede acceder directamente a los datos de la aplicación usuario, el servicio requerido puede ser procesado “localmente” por el servidor sin invocar el servicio Backend. El procesamiento es hecho de acuerdo a la especificación de protocolo aplicación MODBUS. En caso de un error, una respuesta de excepción MODBUS es construida.

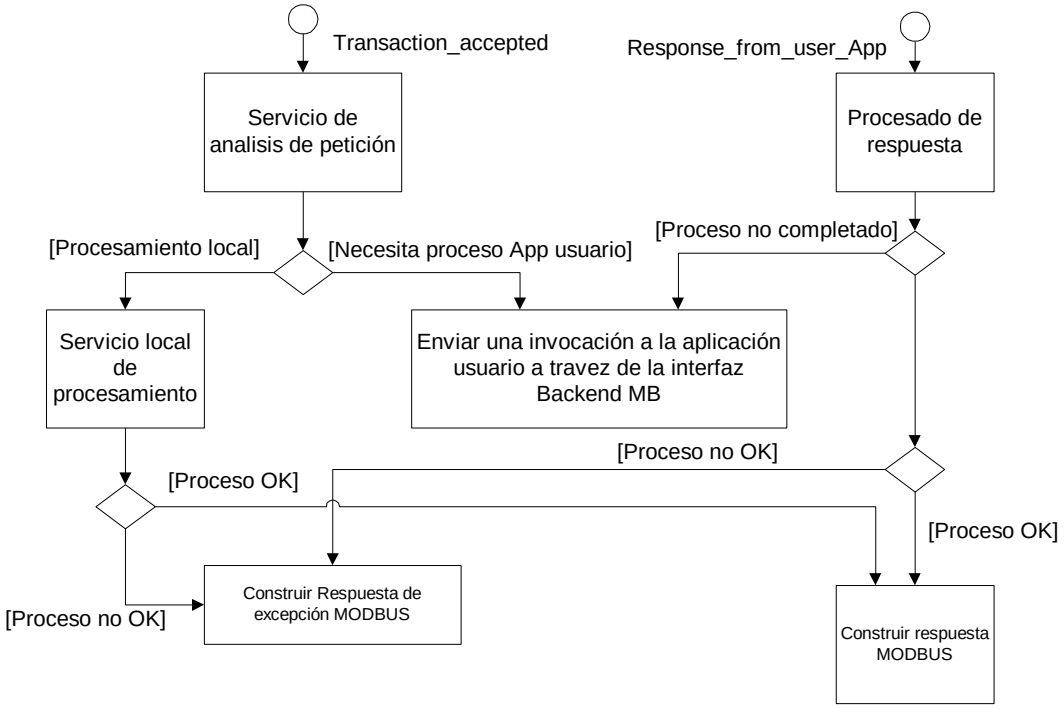

Figura 3.27 Diagrama de actividad de los servicios de procesamiento MODBUS

Dentro de un dispositivo modular multiprocesador o con una arquitectura multi-thread donde las “capas de comunicación” y las “capas de aplicación usuario” son dos entidades separadas, algunos servicios triviales pueden ser procesados completamente por la entidad de Comunicación mientras que algunos otros pueden requerir una cooperación con la entidad aplicación de usuario usando el servicio de Back End.

Para interactuar con la Aplicación Usuario, el servicio Back End de MODBUS debe implementar todos los mecanismos apropiados con el objeto de manejar las transacciones de la Aplicación Usuario y manejar correctamente las invocaciones a la aplicación Usuario y respuestas asociadas.

Interfaz de la Aplicación de Usuario (Interfaz Backend)
Muchas estrategias pueden ser implementadas en el servicio Backend de MODBUS para lograr su trabajo su trabajo sin embargo estas no son equivalente en términos del rendimiento de procesamiento del usuario de red, ancho de banda usado por la interfaz, tiempo de respuesta, o aún carga de trabajo de diseño.
El servicio Backend MODBUS usara la interfaz apropiada a la Aplicación Usuario:
Cualquiera, ya sea una interfaz física basada sobre un enlace serial, o un esquema de un “dual-port RAM”, o una simple línea I/O, o una interfaz lógica basada sobre un servicio de mensajería proveída por un sistema de operación.
La interfaz a la Aplicación de Usuario puede ser sincronía o asíncrona.

El servicio Backend MODBUS también usa el patrón de diseño apropiado para obtener y establecer (“get/set”) atributos de los objetos o para accionar servicios (“trigger services”). En algunos casos, un simple “patrón gateway” será adecuado. En algunos otros casos el diseño tendrá que implementar un “patrón proxy” con su correspondiente estrategia de cacheo, desde una simple tabla de intercambio hasta mas sofisticados mecanismos de replicación.
El servicio MODBUS Backend tiene la responsabilidad de implementar la trascripción MODBUS con el objeto de interactuar con la Aplicación Usuario. Por lo cual, este puede tener que implementar mecanismos para el empaquetamiento fragmentación/reconstrucción, garantía de consistencia de datos, y sincronización es lo requerido.

Construcción de respuesta MODBUS
Una vez la petición ha sido procesada, el servidor MODBUS tiene que construir la respuesta usando el adecuado servidor de transacciones MODBUS y tiene que enviar ésta al componente administrador TCP.
Dependiendo de los resultados del procesamiento, dos tipos de respuesta pueden ser construidas:
Una respuesta positiva MODBUS:
El código de función de la respuesta = el código función de la petición.

Una respuesta de excepción MODBUS:
El objetivo es proveer al cliente información relevante concerniente a la detección de errores durante el procesamiento.
El código de función de la respuesta = condigo de función de la petición + 0x80;
El código de función es proveído o indica la razón del error.

Código de excepción	Nombre MODBUS	Comentarios
01	Código de Función ilegal	El código función es desconocido por el servidor
02	Dirección de datos ilegal	Dependiendo de la petición
03	Valor de dato ilegal	Dependiendo de la petición
04	Fallo servidor	El servidor fallo durante la excepción
05	Acknowledge	El servidor acepta la invocación de servicios pero el servicio requiere un tiempo relativamente largo tiempo ejecutarse. El servidor por lo tanto únicamente retorna una acknowledgement de la recepción de la invocación del servicio.
06	Servidor trabajando	El servidor es incapaz de aceptar la el PDU de la petición MODBUS. La aplicación cliente tiene la responsabilidad de decidir si y cuando reenviar la petición.
0A	Problema con Gateway	La dirección del gateway no esta disponible
0B	Problema con Gateway	El dispositivo objetivo fallo en la respuesta. El gateway genera esta excepción.

Tabla 3.12 Excepciones MODBUS

El PDU de la respuesta MODBUS debe ser prefijado con la cabecera MBAP que es construida usando memoria de datos en el contexto de la transacción.

- Identificador de unidad
El identificador de unidad es copiado como éste fue dado dentro de la petición MODBUS recibida y memorizada en el contexto de la transacción.

- Longitud
El servidor calcula el tamaño del PDU MODBUS más el byte del identificador de unidad. Este valor es establecido en el campo “longitud”.

- Identificador de protocolo
El campo del identificador de protocolo es establecido a 0x0000 (protocolo MODBUS), como este fue dado dentro de la petición recibida MODBUS.

- Identificado de transacción
Este campo es establecido al valor del “Identificador de transacción” que fue asociado con la petición original y memorizada en el contexto de la transacción.

Entonces la respuesta MODBUS debe ser retornada al cliente MODBUS correcto usando la conexión TCP memorizada en el contexto de la transacción. Cuando la respuesta es enviada, el contexto de la transacción debe ser liberado.

3.7 MODBUS sobre línea serial

El estándar MODBUS define un protocolo de mensajería en la capa de aplicación, posicionado en la capa 4 del modelo TCP/IP (capa 7 del modelo OSI) que provee una comunicación cliente-servidor entre dispositivos conectados en diferentes tipos de buses en la red. Estandariza también un protocolo específico en línea serial para intercambiar peticiones entre un maestro y uno o varios esclavos.

3.7.1 Descripción de Protocolo

En esta sección describiremos el protocolo MODBUS sobre línea serial. El protocolo MODBUS serial es un protocolo de Maestro-Esclavo. Este protocolo toma su lugar en la capa 1 y 7 del modelo OSI.

Un sistema de tipo maestro-esclavo tiene un nodo (el nodo maestro) que emite comandos explícitos a uno de los nodos del “esclavo” y procesa la respuesta. Un nodo esclavo típicamente no transmite datos sin una petición del nodo maestro, y no se comunica con otros esclavos.

A nivel físico, los sistemas MODBUS sobre línea serial pueden usar diferentes interfases físicas (RS485 EIA/ TIA -485 Standard, RS232 EIA/ TIA -232 Standard).

Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)

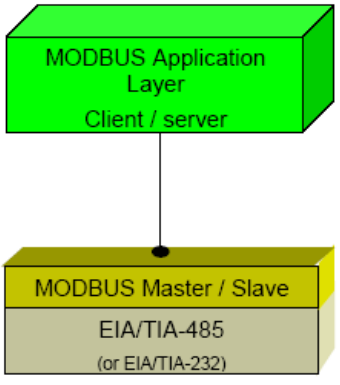

Figura 3.28 Protocolo Modbus y el Modelo OSI

El protocolo de mensajería MODBUS de la capa de aplicación, posicionado el la capa 4 del modelo TCP/IP (capa 7 del OSI), provee una comunicación cliente-servidor entre dispositivos conectados a buses en la red. En MODBUS serial el rol del “cliente” es proporcionado por el “Maestro” del bus serial y nodo “esclavo” actúa como “servidor”.

3.7.2 Capa de enlace de datos MODBUS

3.7.2.1 Principio del Protocolo MODBUS maestro/esclavo

El protocolo en línea serial MODBUS es un protocolo Maestro-Esclavos. Solo un maestro (al mismo tiempo) es conectado al bus, y uno o varios (247 número máximo) nodos esclavos son también conectados al mismo bus serial. Una comunicación MODBUS es siempre iniciada por el maestro. Los nodos esclavos nunca transmitirán datos sin recibir una petición del nodo maestro. Los nodos esclavos nunca se comunicarán uno con otro. El nodo maestro inicia solo una transacción al mismo tiempo.

El nodo maestro emite una petición MODBUS a los nodos esclavos de dos formas:

- En modo unicast, el maestro direcciona un esclavo individual. Después de recibir y procesar la petición, el esclavo retorna un mensaje (una “replica”) al maestro.

  En este modo, una transacción MODBUS consiste de dos mensajes: una petición desde el maestro, y una replica desde el esclavo.

  Cada esclavo debe tener una única dirección (desde 1 hasta 247) y que puede ser direccionado independientemente de otros nodos.

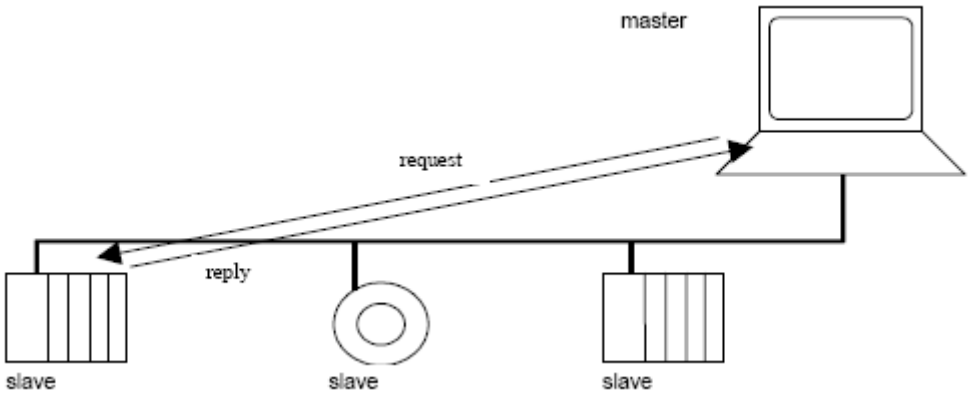

Figura 3.29 Petición MODBUS modo unicast

- En modo broadcast, el maestro puede enviar una petición a todos los esclavos. (esta modalidad no fue implementada)

  Ninguna respuesta es retornada a la petición broadcast enviada por el maestro. Las peticiones broadcast están necesariamente escribiendo comandos. Todos los dispositivos deben aceptar los broadcast para función de escritura. La dirección 0 es reservada para identificar un cambio broadcast.

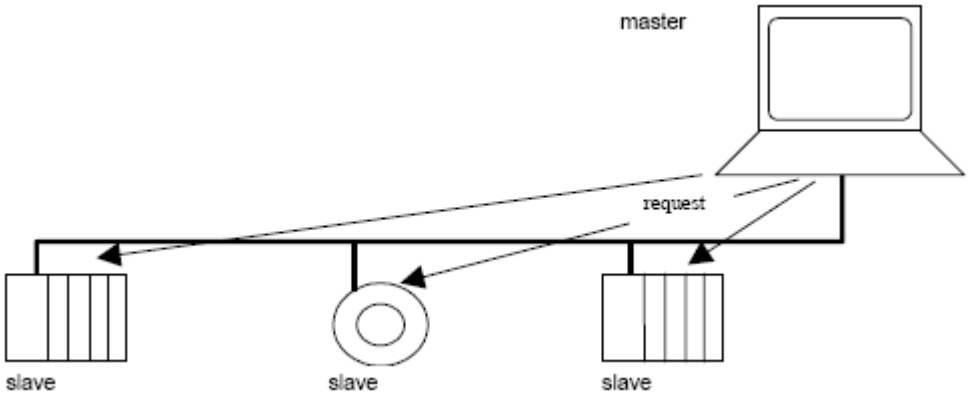

Figura 3.30 Petición MODBUS modo broadcast

.

3.7.2.2 Reglas de direccionamiento MODBUS

El espacio de direccionamiento MODBUS comprende 256 direcciones diferentes.

0	De 1 a 247	De 248 a 255
Dirección Broadcast	**Direcciones individuales de los esclavos**	**Reservado**

Tabla 3.13 Campos de direcciones MODBUS serial

La dirección 0 es reservada como la dirección broadcast. Todos los nodos esclavos deben reconocer la dirección broadcast.

El nodo Maestro MODBUS no tiene una dirección específica, solo los nodos esclavos deben tener una dirección. Esta dirección debe ser única el bus serial MODBUS.

3.7.2.3 Descripción de la trama MODBUS

Las especificaciones del Protocolo de Aplicación MODBUS (MODBUS Aplication Protocol Specification V1.1a [1]), define una simple Unidad de Dato de Protocolo (Protocol Data Unit PDU) independiente de las capas de comunicación subyacentes.

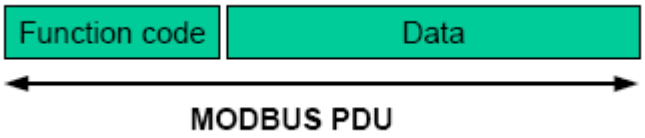

Figura 3.31 Unidad de Dato de Protocolo MODBUS

El mapeo del protocolo MODBUS en un bus específico de la red introduce algunos campos adicionales en la Unidad de Dato de Protocolo PDU. El cliente que inicia una transacción MODBUS construye el PDU MODBUS y agrega campos en orden para construir la comunicación apropiada del PDU.

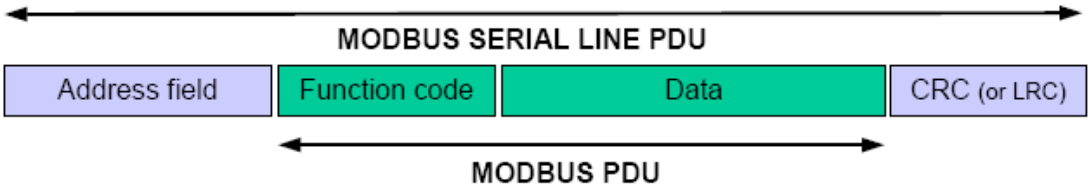

Figura 3.32 Trama MODBUS sobre línea serial

En MODBUS línea serial, el campo de direcciones solo contiene la dirección del esclavo.

Como se describió previamente las direcciones de los nodos esclavos que son válidos están en el rango de 0 – 247 en decimal. El esclavo individual de un dispositivo están asignadas direcciones en el rango de 1 – 247. Un maestro direcciona a un esclavo poniendo la dirección del esclavo en el campo de dirección del mensaje. Cuando el esclavo retorna su respuesta coloca su dirección en el campo de dirección de la respuesta para que el master conozca cual esclavo esta respondiendo.

El código de función indica al servidor que tipo de acción va a realizar. El código de función puede ser seguido por un campo de dato que contienen parámetros de peticiones y repuestas.

El campo de chequeo de error es el resultado de un cálculo de “Redundancy Checking” que es realizado en los contenidos del mensaje. Dos tipos de métodos de cálculos son usados dependiendo del modo de transmisión que esta siendo usado (modo RTU o ASCII).

3.7.2.4 Diagramas de estado Maestro / Esclavos

La capa de enlace de datos MODBUS comprende dos sub capas separadas:

- El protocolo Maestro / esclavo
- El modo de transmisión (el modo RTU vrs ASCII)

Las siguientes secciones describen los diagramas de estado de un maestro y de un esclavo que son independientes del modo de transmisión usada.

Los modos de transmisión RTU y ASCII son especificados más adelante, usando dos diagramas de estado. La recepción y envío de una trama son descritos.

3.7.2.4.1 Diagrama de estado del maestro

En la figura 3.33 se explica el comportamiento del maestro

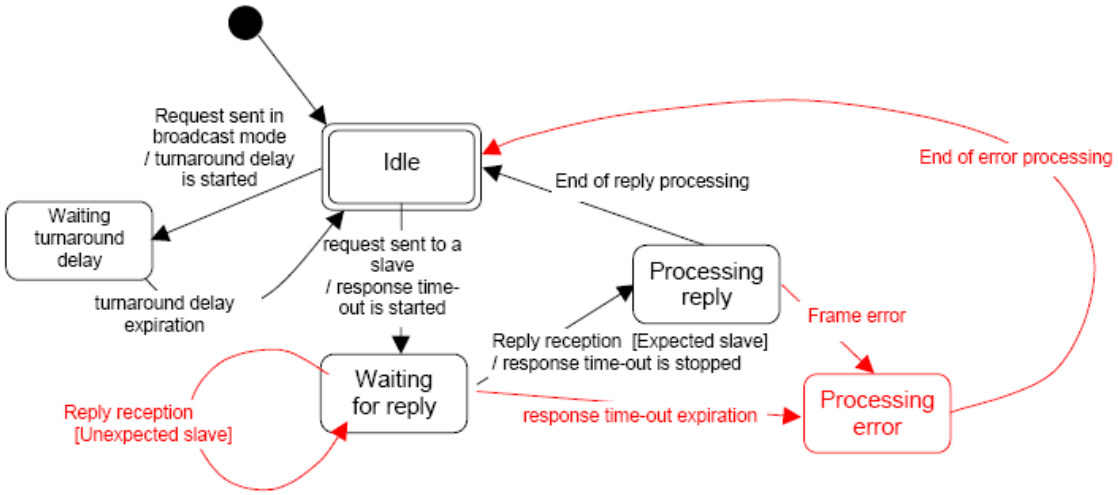

Figura 3.33 Diagrama de estado del maestro

Algunas explicaciones del diagrama de estado anterior:

- El estado “Idle” (ocioso) = ninguna petición pendiente. Este es el estado inicial después de encendido. Una petición solamente puede ser enviada en estado “ocioso”. Después de enviar una petición, el maestro deja el estado ocioso y no puede enviar una segunda petición al mismo tiempo.
- Cuando una petición unicast es enviada a un esclavo, el maestro pasa al estado “Esperando por replica” y un “Tiempo de repuesta” es iniciado. Se previene que el maestro permanezca indefinidamente en el estado “Esperando por replica”. El valor de tiempo de respuesta es una aplicación dependiente.
- Cuando una replica es recibida, el Maestro verifica la replica antes de iniciar el procesamiento de datos. La verificación de la replica podría resultar en un error, por ejemplo una replica de un esclavo inesperado, o un error en la recepción de la trama. En el caso de una replica recibida de un esclavo inesperado, el tiempo de repuesta se mantiene mientas corre. En el caso de un error detectado en la trama, una retransmisión podría ser realizada.
- Si la replica no es recibida, el tiempo de repuesta expira, y un error es generado. Entonces el maestro vuelve al estado “ocioso”, habilitando una retransmisión de la petición. El número de retransmisiones depende de la configuración del maestro.

- Cuando una petición broadcast es enviada en el bus serial, no se retorna ninguna respuesta de parte de los esclavos. Sin embargo un retardo es respetado por el Maestro para permitir que algún esclavo procese la petición actual antes de enviar una nueva. Este retardo es llamado “retardo de repunte” (Turnaround delay). Por lo tanto el maestro pasa al estado “Esperando un retardo de repunte” antes de volver al estado “ocioso” y antes de estar disponible para enviar otra petición.
- En unicast el tiempo de repuesta debe ser bastante grande para que cualquier esclavo procese la petición y retorne la respuesta, en broadcast el “retardo de repunte” debe ser lo bastante grande para que cualquier esclavo procese solamente la petición y este disponible para recibir una nueva. Por lo tanto el “retardo de repunte” debería ser mas corto que el tiempo de respuesta. Típicamente el tiempo de respuesta es de 1s a varios segundos a 9600 bps, y el retardo de repunte es de 100ms a 200ms

3.7.2.4.2 Diagrama de estado del esclavo

En la figura 3.34 se explica el comportamiento del esclavo:

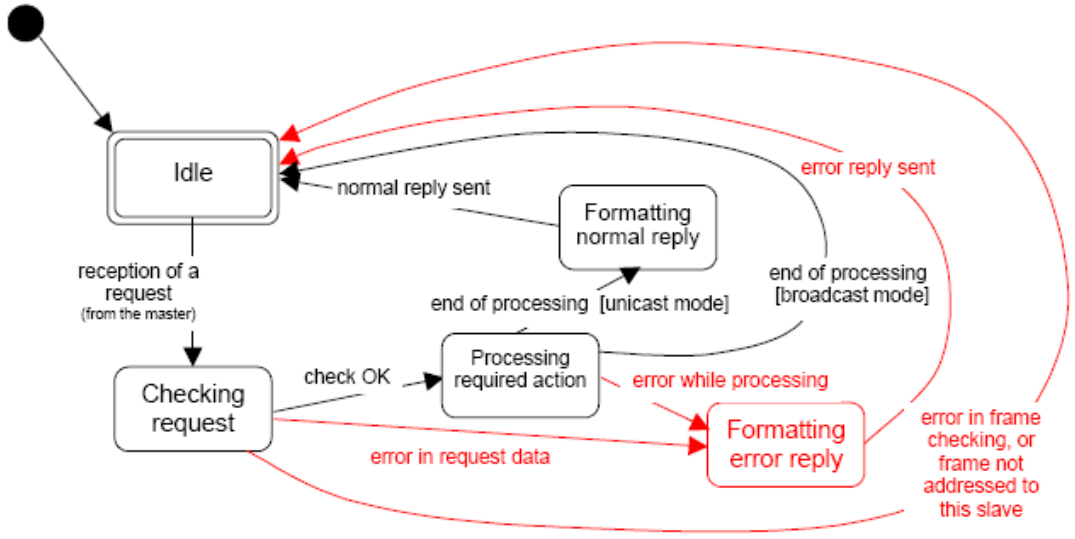

Figura 3.34 Diagrama de estado del esclavo

Algunas explicaciones del diagrama de estado anterior:

- El estado “Idle” (ocioso) = ninguna petición pendiente. Este es el estado inicial después de encendido.
- Cuando una petición es recibida, el esclavo verifica el paquete antes de realizar la acción requerida en el paquete. Diferentes errores podrían ocurrir: errores de formato en la petición, acción inválida,… En Caso de error, una replica debe ser enviada al maestro.
- Una vez la acción requerida ha sido completada, un mensaje unicast requiere que una replica deber ser hecha y ser enviada al maestro.
- Si un esclavo detecta un error en la trama recibida, no se regresa una respuesta al maestro.

3.7.2.5 Los dos modos de transmisión serial

Dos modos diferentes de transmisión serial son definidos: El modo RTU y el modo ASCII.
Define el bit contenido de campos de mensajes transmitidos serialmente en la línea. Determina como la información es empaquetada en los campos de mensaje y como es decodificada.

El modo de transmisión (y parámetros del puerto serial) deben ser los mismos para todos los dispositivos en una línea serial MODBUS.
Aunque el modo ASCII es requerido en algunas aplicaciones específicas, la interoperabilidad entre dispositivos MODBUS puede ser alcanzada solo si cada dispositivo tiene el mismo modo de transmisión: Todos los dispositivos deben implementar el modo RTU. El modo de transmisión ASCII es opcional.
Los dispositivos deben de ser configurados para que el usuario decida el modo de transmisión, RTU o ASCII. El modo por defecto debe ser el modo RTU

3.7.2.6 Modo de Transmisión RTU

Cuando los dispositivos se comunican en una línea serial MODBUS usando el modo RTU (Remote Terminal Unit), cada 8-bit (byte) en un mensaje contiene dos caracteres hexadecimales (dos de 4-bit). La principal ventaja de este modo es que su gran densidad de caracteres, que permite una mejor tasa de transferencia de los datos que el modo ASCII para la misma tasa de baudios. Cada mensaje debe ser transmitido en un flujo continuo de caracteres.

El formato para cada byte (11 bits) en modo RTU es:

Sistema de codificación: 8-bits binarios
Bits por Byte: 1 bit de inicio
8 bits de datos, el bit menos significativo se envía primero
1 bit de complemento de paridad
1 de alto

La paridad par es requerida, otros modos (paridad impar, no paridad) podrían ser usados también. Para asegurar una máxima compatibilidad con otros productos, es recomendado que soporte también el modo de no paridad. El modo de paridad por defecto debe ser paridad par.
El uso de no paridad requiere dos bits de alto.

Como los caracteres son transmitidos serialmente:

Cada carácter o byte es enviado en este orden (de izquierda a derecha):

Bit menos significante (LSB)…….. Bit más significante (MSB)

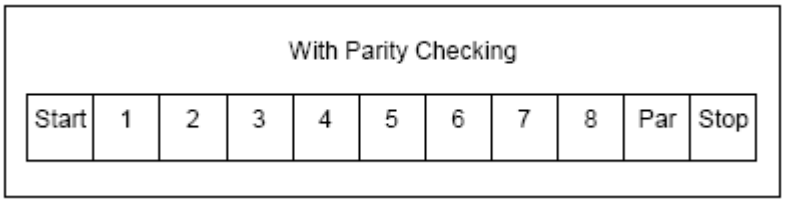

Figura 3.35 Secuencia de bit en modo RTU

Los dispositivos podrían aceptar configuración de verificación de paridad par, impar o no paridad. Si es implementada la no paridad, un bit adicional de alto es transmitido para llenar la trama del carácter, para llenar los 11-bits del carácter asíncrono.

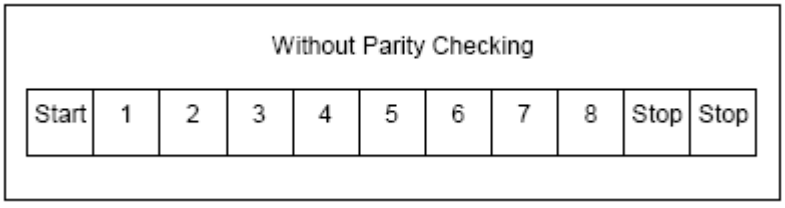

Figura 3.36 Secuencia de bit en modo RTU (caso específico de no paridad)

Campo de verificación de trama: verificación de redundancia cíclica (CRC)
Descripción de la trama:

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Figura 3.37 Trama de mensaje RTU

El tamaño máximo de una trama MODBUS RTU es 256 bytes

3.7.2.7 Entramado de un mensaje MODBUS RTU

Un mensaje MODBUS es colocado por el dispositivo transmisor en una trama que tiene inicio conocido y un punto de finalización. Esto permite al dispositivo que reciba una nueva trama a comenzar en el inicio del mensaje, y conocer cuando el mensaje es completado. Mensajes parciales deben ser detectados y los errores deben ser puestos como un resultado.

En modo RTU, las tramas de los mensajes están separados por un intervalo de silencio de menos de 3.5 tiempos de carácter. Este intervalo de tiempo es llamado t3,5.

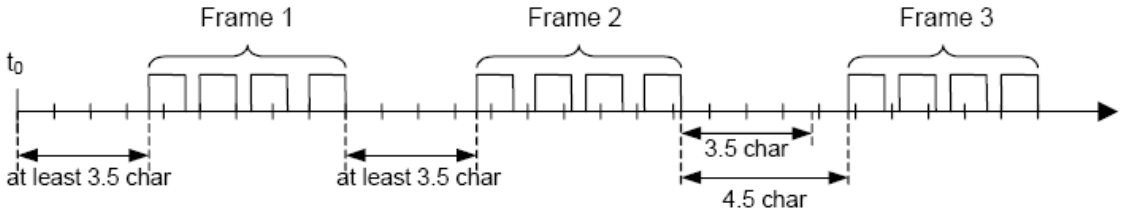

Figura 3.38 Entramado de mensaje RTU

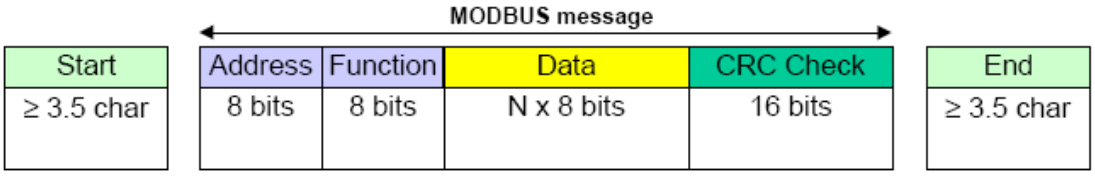

Figura 3.39 Trama de mensaje RTU

La trama del mensaje completo debe ser transmitida como un flujo continuo de caracteres.
Si un intervalo de silencio de mas de 1.5 tiempos de caracter ocurren entres dos caracteres, la trama del mensaje es declarada incompleta debería ser descartada por el receptor.

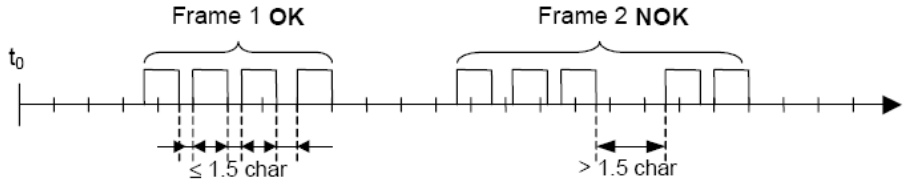

Figura 3.40 Entramado de mensaje RTU con un tiempo entre caracteres de más de 1.5.

La siguiente figura provee una descripción del diagrama de estado del modo de transmisión RTU. Ambos puntos de vista “maestro” y esclavo” están expresados en la misma figura.

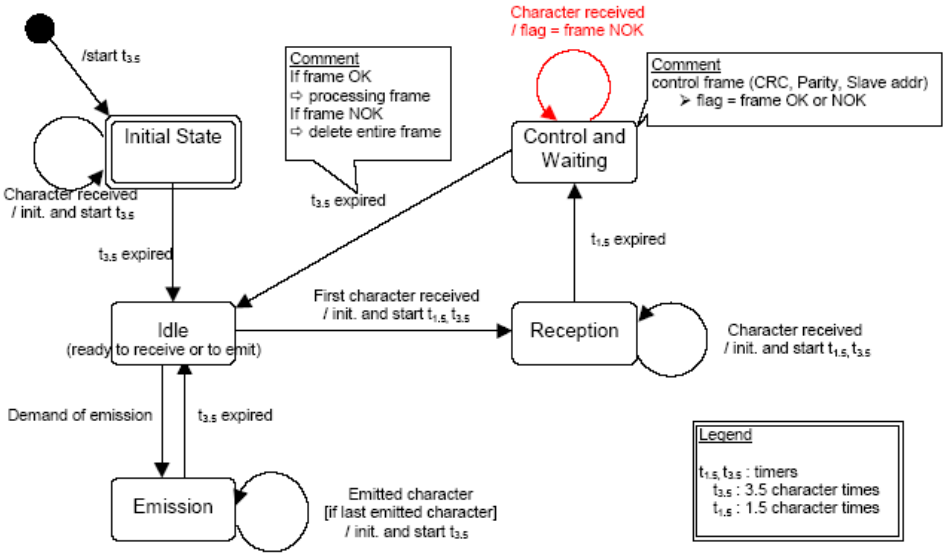

Figura 3.41 Diagrama de estado del modo de transmisión RTU

Algunas explicaciones del diagrama de estado:

- La transmisión de un “estado inicial” a uno “ocioso” necesita $t_{3.5}$ tiempo de expiración: que asegura el retardo entre las tramas.
- El estado “ocioso” es el estado normal cuando ni la emisión ni la recepción es activada.
- En el modo RTU, el enlace de la comunicación es declarado en modo “ocioso” cuando no hay actividad de transmisión después de un intervalo de tiempo igual o menor que 3,5 caracteres.
- Cuando el enlace está en estado “ocioso”, cada carácter transmitido y detectado en el enlace es identificado como el inicio de la trama. El enlace pasa a estado “activo”. Entonces, el final de la trama es identificado cuando no hay mas caracteres transmitidos en el enlace, después de un intervalo de tiempo $t_{3,5}$.
- Después de detectar el fin de la trama, el cálculo y verificación del CRC es completado. Después del campo de dirección es analizado para determinar si la trama es para el dispositivo. Si no la trama es descartada. Para reducir el tiempo del proceso de recepción, el campo de dirección es analizado tan pronto como es recibido sin esperar el final de la trama. En este caso el CRC es calculado y verificado solo si la trama es direccionada al esclavo (incluida la trama broadcast).

3.7.2.8 Verificación de CRC

El modo RTU incluye un campo de verificación de error que es basado en un método de verificación de redundancia cíclica (CRC) ejecutado en el contenido del mensaje.
El campo CRC verifica el contenido del mensaje completo. Es aplicado indiferentemente de cualquier método de verificación de paridad usado para los caracteres individuales del mensaje.

El campo CRC contiene un valor de 16-bits implementado como dos de 8-bits (bytes)

El campo CRC es añadido al mensaje como el último campo en el mensaje. Cuando este es completado, el byte de menor orden en el campo es añadido primero, seguido por el byte de mayor orden. El byte de mayor orden en el CRC es el último byte enviado en el mensaje.

El valor del CRC es calculado por el dispositivo que envía, el cual agrega el CRC al mensaje. El dispositivo receptor recalcula un CRC durante la recepción del mensaje, y compara el valor calculado al valor actual recibido en el campo del CRC. Si los dos valores no son iguales, da como resultado un error.

CAPÍTULO IV

COMPONENTES ELECTRÓNICOS

Introducción

En esta sección se describirán los componentes electrónicos a utilizar en la etapa de hardware a desarrollar. El hardware consiste de una tarjeta que le permitirá al medidor de energía conectarse a la red Ethernet, estableciendo una comunicación cliente/servidor para el monitoreo remoto de los parámetros eléctricos.

4.0 RTL8019AS

En esta sección se dará una descripción general de un Controlador de Interfase de Red NIC (Network Interface Controller). El énfasis estará colocado en su operación y en su uso.
La NIC posee todas las funciones de la capa de control de acceso al medio MAC requerida para la transmisión y recepción de paquetes de acuerdo con el estándar IEEE 802.3 CSMA/CD. El controlador actúa como un periférico avanzado y sirve como una interfase completa entre el sistema y la red. La FIFO y los canales DMA trabajan juntos para formar un esquema de la administración de paquetes.
El RTL8019AS es fabricado por *RealTek* y contiene todo lo necesario para transmitir y recibir paquetes Ethernet. Es un integrado potente para integrar en dispositivos de red, proporcionando comunicación Full-Duplex lo cual permite transmisión y recepción al mismo tiempo, duplicando el ancho de banda.

4.0.1 Características principales

- 100 pines PQFP
- Compatible a Ethernet II e IEEE 802.3 10Base5, 10Base2, 10BaseT
- Soporta función Full Duplex Ethernet para duplicar el ancho de banda del canal

- Soporta tres niveles de *power down*:
- Sleep
- Power down con reloj interno corriendo
- Power down con reloj interno detenido
- Soporta UTP, auto detección de AUI & BNC,
- Soporta corrección de autopolaridad 10BaseT
- Soporta 8 líneas IRQ
- Soporta opciones de direccionamiento basados en 16 I/O
- Soporta 16K, 32K, 64K y modo de acceso a 16K-pagina en BROM (Hasta 256 paginas con 16Kbytes/pagina)
- Soporta lectura/escritura de memoria Flash
- SRAM de 16kbytes
- Soporta 4 pines de LED de diagnostico con salidas programables

4.0.2 Diagrama de pines

El diagrama del RTL8019AS se muestra en la figura 4.2, donde se puede observar la dificultad para soldar los pines, por el tipo de empaquetamiento.

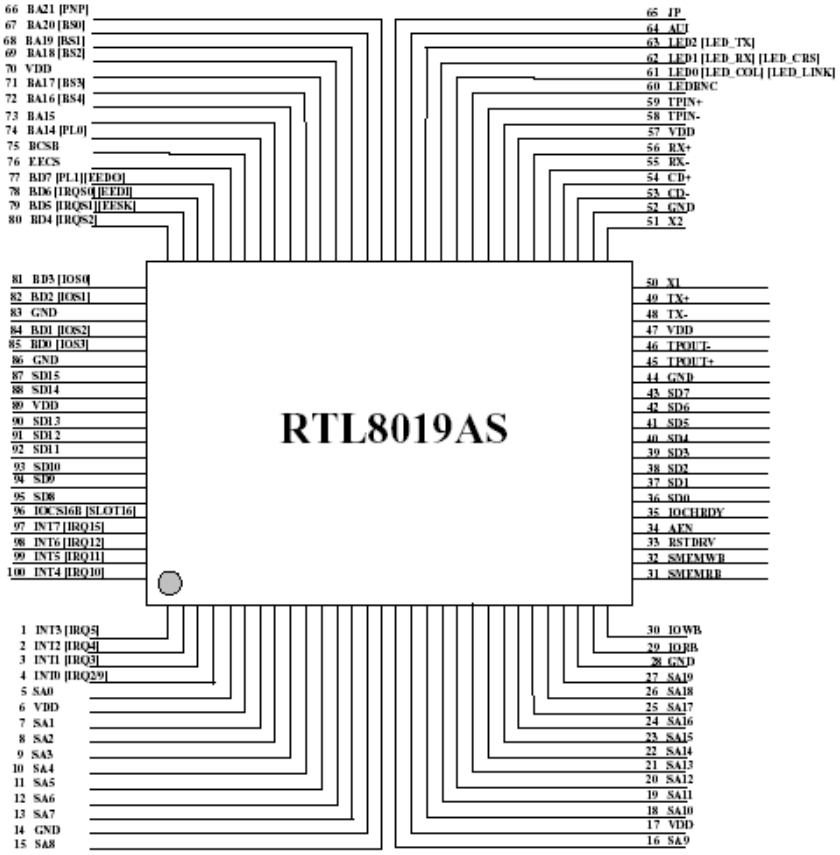

Figura 4.1 Diagrama de pines para el RTL8019AS

4.0.3 Descripción de pines

Pines de alimentación de potencia:

No	Nombre	Tipo	Descripción
6, 17, 47, 57, 70, 89	VDD	P	Alimentación + 5V DC
14, 28, 44, 52, 83, 86	GND	P	Ground

Tabla 4.1 Pines de Alimentación de Potencia

Pines de interfaz ISA BUS

No	Nombre	Tipo	Descripción
34	AEN	I	Habilitación de dirección. Esta señal ISA debe ser baja para un comando valido de I/O
97-100,1-4	INT7-0	O	Líneas de petición de interrupción que son mapeadas a IRQ15, IRQ12, IRQ11, IRQ10, IRQ5, IRQ4, IRQ3, IRQ2/9 respectivamente.
35	IOCHRDY	O	La señal ISA es llevada abajo para insertar un ciclo de espera al comando de escritura/lectura del computador actual.
96	IOCS16B [SLOT16]	O	Sobre una señal de reestablecimiento de energía.
29	IORB	I	Comando de lectura I/O en host
30	IOWB	I	Comando de escritura I/O en host
33	RSTDRV	I	Señal de reset activo alto del harwarw del bus ISA. Pulsos con nivel alto menor que 800ns son ignorados
27-18, 16-15, 13-7, 5	SA19-0	I	Dirección del bus del host. SA10 es agregado para implementar la decodificación completa de los puertos PnP, dirección 279h y A79h. In RTL8019, SA10 no es decodificada. In RTL8019AS, SA10 debería ser 0 para un acceso válido a los puertos PnP
87-88, 90-95, 43-36	SD15-0	I/O	Bus de datos del host
31	SMEMRB	I	Comando de lectura de memoria del host
32	SMEMWB	I	Comando de escritura de memoria del host. Este pin es agregado para decodificar el comando de escritura de una memoria flash

Tabla 4.2 Pines de interfaz ISA BUS

Pines de la Interfase de Memoria (incluyendo BROM, EEPROM)

No	Nombre	Tipo	Descripción
75	BCSB	O	Selección de chip BROM. Señal activa baja, afirma cuando BROM es leída. RTL8019AS maneja este pin bajo cuando SA19-14 igualan a la dirección base de memoria BROM seleccionada y también de las 2 condiciones mostradas abajo: (1) SMEMRB es baja (2) SMEMRB es baja y la función de lectura de la memoria flash del RTL8019AS esta habilitada.
76	EECS	O	Selección de chip 9346. Señal activa alta, afirma cuando 9346 es lectura/escritura
66-69, 71-74 77-82, 84-85	BA21-14 BD7-0	O I/O	Dirección BROM.* Bus de datos BROM
[79] [78] [77]	[EESK] [EEDI] [EEDO]	O O I	Reloj de datos serial 9346 Entrada de datos serial 9346 Salida de datos serial 9346
[66]	[PNP]	I	Los siguientes pines son definidos para opciones de jumper. Sus estados son cambiados al caer al borde de RSTDRV, entonces ellos son cambiados para servir como el bus SRAM. Cada uno de ellos es internamente bajado por una resistencia de 100KW. Por lo tanto, la entrada será baja cuando deja abierto y alto cuando sube externamente por una resistencia de 10K. Cuando es alto en el modo jumperless (i.e. JP=bajo), el RTL8019AS es forzado en el modo Plug and Play indiferente del contenido de 9346.
[72-71, 69-67]	[BS4-0]	I	Los siguientes pines no son atendidos en el modo jumperless (JP=bajo) Selección del tamaño y direcciones base de la BROM
[85-84, 82-81]	[IOS3-0]	I	Selección de direcciones base de I/O
[77, 74]	[PL1-0]	I	Selección del tipo de medio de la red
[80-78]	[IRQS2-0]	I	Selección de una línea de interrupción entre INT7-0
65	JP	I	Cuando es alto, este pin selecciona el modo jumper. Cuando es bajo, selecciona el modo jumperless (incluyendo RT jumperless y Plug and Play).

Tabla 4.3 Pines de la Interfaz de Memoria (incluyendo BROM, EEPROM)

Pines de la Interfaz del Medio

No	Nombre	Tipo	Descripción
64	AUI	I	Esta entrada es usada para detectar el uso de una MAU externa en la interfaz AUI. La entrada debería ser manejada baja para BNC embedida y alta para MAU externa. Cuando la entrada es alta, RTL8019AS fija al bit (bit5) del AUI en CONFIG0 y maneja al LEDBNC bajo inhabilitando al BNC. Si este pin no es usado, debería ser conectado a GND tal que el RTL8019AS actúe como el RTL8019.
54, 53	CD+, CD-	I	Este par de entradas de colisión AUI porta la señal de entrada del diferencial de colisión desde el MAU.
56, 55	RX+, RX-	I	Este par de entradas de recepción AUI porta la señal de entrada del diferencial de recepción desde el MAU.
49, 48	TX+, TX-	O	Este par de salidas de transmisión contiene el manejo de líneas diferenciales que envían datos codificados Manchester al MAU. Estas salidas son seguidores de fuente y requiere resistencias de 270 ohm para conexión a tierra.
59, 58	TPIN+, TPIN-	I	Ester par de entradas TP recibe los 10 Mbits/s diferenciales de datos codificados Manchester desde el cable de par trenzado.
45, 46	TPOUT+, TPOUT-	O	Este par porta el diferencial de salida transmitida TP. La señal de salida codificada Manchester será pre-deformada para prevenir sobrecarga sobre el par-trenzado y esto reduce el efecto de fluctuaciones.
50	X1	I	Cristal de 20Mhz o entrada de oscilador externa.
51	X2	O	Salida de realimentación de cristal. Esta salida es usada únicamente en conexiones del cristal. Esta debe de estar abierta cuando X1 es manejada con un oscilador externo.

Tabla 4.4 Pines de la Interfaz del Medio

Pines de salida del LED

No	Nombre	Tipo	Descripción
60	LEDBNC	O	Este pin va a nivel alto cuando el tipo de medio del RTL8019AS es establecido al modo 10Base2 o el modo de auto detección con la prueba de enlace fallida. De otra forma, este pin es bajo. Este pin puede ser usado para controlar la alimentación del convertidor DC para CX MAU y conectada a un LED para indicar el uso del tipo de medio.
61	LED0	O	Cuando el bit LEDS0 (en registro CONFIG3 de la página 3 del RTL8019AS) es 0, este pin actúa como LED_COL. Cuando LEDS0=1, este actúa como LED_LINK.
62, 63	LED1, LED2	O	Cuando el bit LEDS1 (en registro CONFIG3 de la página 3 del RTL8019AS) es 0, estos dos pines actúan como LED_RX y LED_TX respectivamente. Cuando LEDS1=1, estos pines actúan como LED_CRS y MCSB.

Tabla 4.5 Pines de salida del LED

4.0.4 Descripción de funcionamiento

4.0.4.1 Modo de operación

La NIC es usada como un dispositivo periférico estándar y es controlado por medio de un arreglo de registros dentro del mismo. Estos registros son usados durante la inicialización, la transmisión y recepción de paquetes, las operaciones DMA remotas. Durante la inicialización, los filtros de la dirección física y la dirección multicast son configuradas, la recepción, transmisión, y rutas de datos son configuradas, los canales DMA son preparados, y las interrupciones apropiadas son enmascaradas. El registro de Comandos CR (Command Register) es usado para iniciar las operaciones de transmisión y del DMA remoto.

Sobre la recepción de paquetes, finalización de transmisión de paquetes, conclusión del DMA remoto, o condiciones de errores, una interrupción es generada para indicar que una acción debe ser tomada. La ruina que maneja las interrupciones lee el Registro de Estado de Interrupciones ISR (Interrupt Status Register) para determinar que tipo de interrupción ha ocurrido, y realizar la acción apropiada.

4.0.4.2 Recepción de paquetes

El canal de recepción del DMA local usa una estructura de buffer de anillo que comprende una serie de buffers de longitudes fijas continuas de 256 bytes (128 palabras) para almacenar los paquetes recibidos.

La localización de un buffer de anillo receptor es programada en dos registros, un registro de inicio de página y un registro de finalización de página, los paquetes Ethernet consisten de una distribución de paquetes control de enlace cortos y paquetes de datos largos, la longitud del buffer de 256 bytes proporciona un buen compromiso entre paquetes cortos y paquetes largos para un uso eficiente de la memoria. Además estos buffers proveen recursos de memoria para el almacenamiento de paquetes back-to-back en redes cargadas.

La asignación de buffers para el almacenamiento de paquetes es controlado por el administrador lógico del buffer en la NIC. El administrador lógico del buffer proporciona tres funciones básicas: enlaces de los buffers de recepción para paquetes largos, recuperación de buffers cuando un paquete es rechazado, y recirculación de páginas del buffer que han sido leídas por el host.

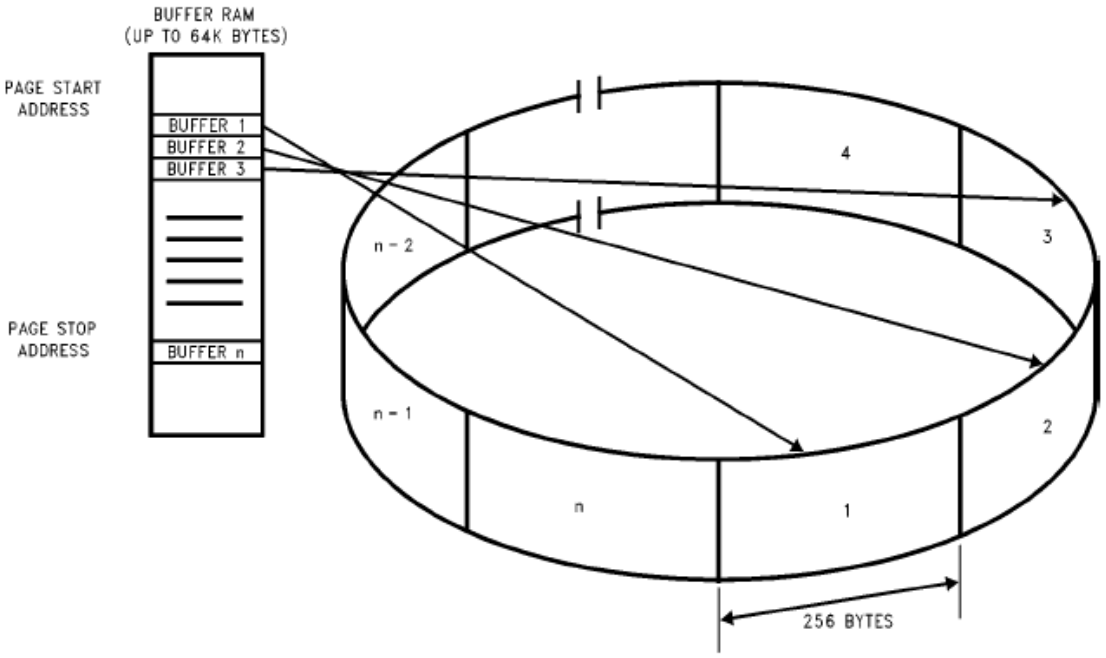

Figura 4.2 Buffer de anillo receptor de la NIC.

Al iniciar, una porción de 64 kbyte (o 32k palabras) espacios de dirección es reservado para el buffer de anillo de recepción. Dos registros, el registro de dirección de inicio de página (PSTART) y el registro de dirección de alto de

página (PSTOP) definen los límites físicos de donde se encontrará el buffer. La NIC trata la lista de buffers como un anillo lógico; siempre que el DMA alcanza la dirección de alto de página, el DMA es restaurado a la dirección de inicio de página.

Inicialización del buffer de anillo

Dos registros estáticos y dos registros de trabajo controlan la operación del buffer de anillo. Estos son el registro de inicio de página, y el registro de alto de página (ambos descritos previamente), el registro de página actual (Current) y el registro de puntero de límite (Boundary).

El registro de página actual apunta al primer buffer usado para almacenar un paquete y es usado para reestablecer el DMA para el estado de escritura al buffer de anillo o para reestablecer las direcciones del DMA en el evento de un paquete “Runt” (muy pequeño), un CRC, o un error de alineación de trama. El registro límite (Boundary) apunta al primer paquete en el anillo no leído todavía por el host, si la dirección del DMA local siempre alcanza al “Boundary”, la recepción es abortada. El puntero “Boundary” también es usado para inicializar el DMA remoto para remover un paquete y es adelantado cuando un paquete es removido. Una analogía simple para recordar la función de estos registros es que el registro de página actual (current) actúa como un puntero de escritura y el puntero límite (boundary) actúa como un puntero de lectura.

Se tiene que tener en cuenta que al momento de la inicialización el valor del registro de inicio de página debe ser cargado en ambos el registro de página actual (Current) y el registro de puntero límite (Boundary). También que el registro de inicio de página no debe ser inicializada en la dirección 00H.

4.0.4.3 Transmisión de paquetes

El DMA local es también usado durante la transmisión de un paquete. Tres registros controlan la transferencia DMA durante la transmisión, un registro de dirección de inicio de página de transmisión TPSR (Transmit Page Start Address Register) y los registros de contador de byte de transmisión TBCR0,1 (Transmit Byte Count Register). Cuando la NIC recibe el comando para transmitir un paquete apuntado por estos registros, el dato en memoria del buffer será movido a la FIFO como requerido durante la transmisión. La NIC generará y agregará el preámbulo, los campos synch y CRC.

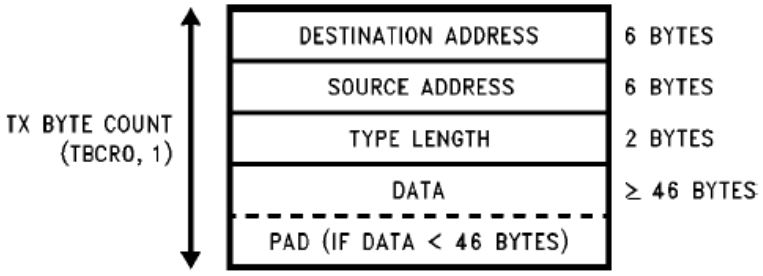

Figura 4.3 Formato de paquete a transmitir

Transmisión y ensamblaje de paquete

La NIC requiere un paquete ensamblado con el formato mostrado. El contador de bytes transmitidos incluye, la dirección de destino, dirección fuente, campo de longitud y dato. No se incluye un preámbulo ni un CRC. Cuando se transmite un dato más pequeño que 46 bytes, el paquete tiene que ser rellenado a un tamaño mínimo de 64 bytes.

Antes de la transmisión, el registro de inicio de página transmitida TPSR (Transmit Page Stara Register) y el TBCR0, TBCR1 (Transmit Byte Count Register) deben ser inicializados. Para iniciar la transmisión del paquete el TXP del registro de comando CR es puesto a 1. El registro de estado de transmisión (TSR) es limpiado. La NIC comienza a transmitir los datos de la memoria (a menos que la NIC esté actualmente recibiendo datos).

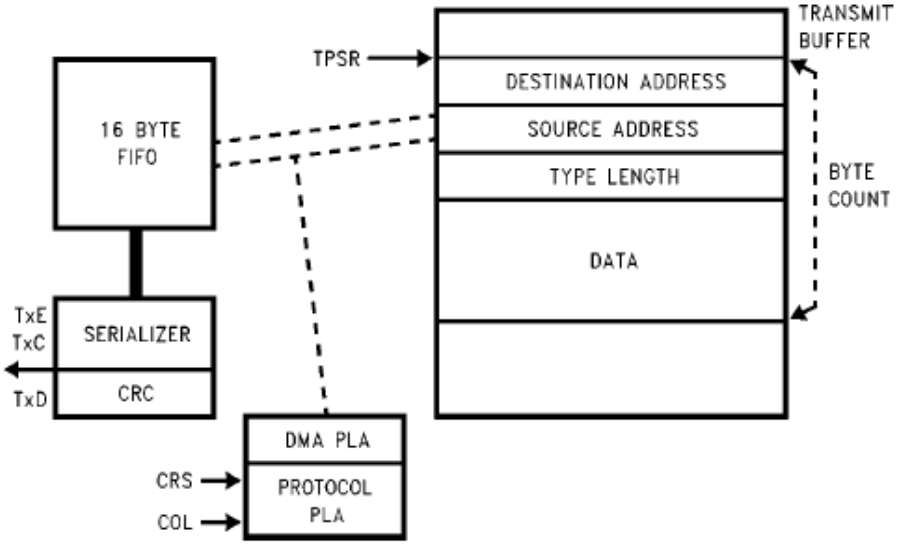

Figura 4.4 Transmisión de paquete

4.0.4.4 DMA remoto

El canal DMA remoto es usado para ambos, ensamblar el paquete para la transmisión y remover paquetes recibidos del buffer de anillo receptor. Puede también ser usado como un canal DMA esclavo de propósitos generales para mover bloques de datos o comandos entre la memoria del host y la memoria del buffer local. Hay tres modos de operación, escritura remota, lectura remota, o enviar paquete.

Dos pares de registros son usados para controlar al DMA remoto, un par de registros de dirección de inicio remoto RSAR0, RSAR1 (Remote Start Address Register) y otro de Contador de bytes remoto RBCR0, RBCR1 (Remote Byte Count Register). El par de registros de dirección de inicio apunta al inicio de un bloque a ser movido mientras el par de registros de contador de bytes se usa para indicar el número de bytes a ser transferidos.

Escritura Remota

La transferencia de escritura remota es usada para mover un bloque de datos del host a la memoria del buffer local. El remoto DMA leerá datos del puerto I/O y secuencialmente escribe en la memoria del buffer local comenzando en la dirección de inicio remoto (RSAR). La dirección del DMA será incrementada y el contador de bytes será decrementado después de cada transferencia, El DMA es finalizado cuando el registro contador de bytes remoto alcanza el valor de cero.

Lectura Remota

La transferencia de lectura remota es usada para mover un bloque de datos de la memoria del buffer local al host. El DMA remoto secuencialmente leerá datos de la memoria de buffer local comenzando en la dirección de inicio remoto y escribe los datos en el puerto I/O). La dirección del DMA será incrementada y el contador de bytes será decrementado después de cada transferencia, El DMA es finalizado cuando el registro contador de bytes remoto alcanza el valor de cero.

Comando de enviar paquete

El canal DMA remoto puede ser inicializado automáticamente para transferir un paquete sencillo del buffer de anillo receptor. El CPU comienza la transferencia emitiendo un comando enviar paquete “Send Packet”. El DMA será inicializado al valor del registro del puntero de límite (Boundary) y al registro de contador de byte remoto (RBCR0, RBCR1) será inicializado al valor de los campos del contador de byte recibidos encontrado en la cabecera del buffer de cada paquete. Después el dato es transferido, el puntero “Boundary” es adelantado para permitir a los buffers ser usados para nuevos paquetes recibidos. La lectura remota terminará cuando el contador de byte sea igual a cero. El DMA remoto

está entonces preparado para leer el próximo paquete recibido del buffer de anillo. Si el puntero DMA cruza el registro de alto de página, se formatea a la dirección de inicio de página. Esto permite al DMA remoto remover paquetes que tienen cubierto el contorno de la superficie del buffer del anillo.

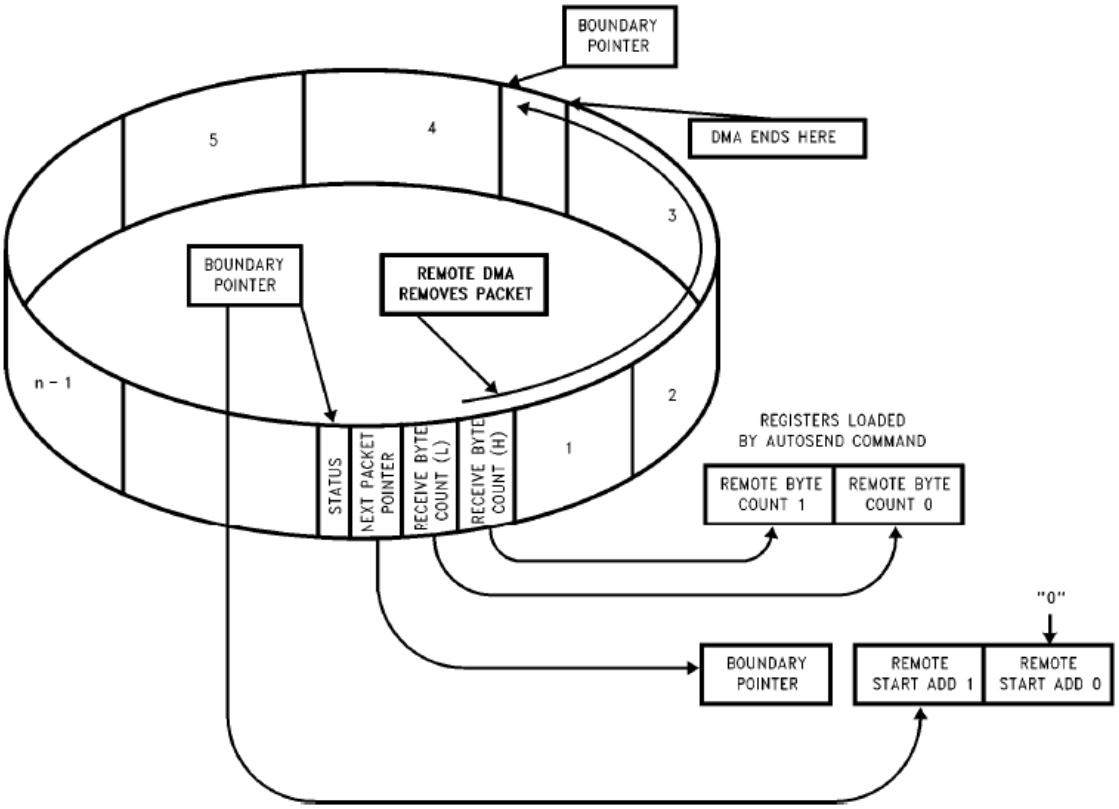

Figura 4.5 Auto inicialización del DMA remoto del buffer de anillo

4.0.5 Descripción de registros

Los registros en el RTL8019AS pueden ser bruscamente divididos en dos grupos por sus direcciones y funciones, uno para NE2000, y el otro para Plug and Play (PnP).

Grupo 1: Registros NE2000
Este grupo incluye cuatro páginas de registros las cuales son seleccionadas por el bit PS0 y PS1 en el registro CR. Cada página contiene 16 registros. Además esos registros son compatibles con NE2000, el RTL8019AS define algunos para configuración de software y la mejora de las características.

Tabla de Registro

No (Hex)	Página 0		Página 1	Página 2	Página 3	
	[R]	[W]	[R/W]	[R]	[R]	[W]
00	CR	CR	CR	CR	CR	CR
01	CLDA0	PSTART	PAR0	PSTART	9346CR	9346CR
02	CLDA1	PSTOP	PAR1	PSTOP	BPAGE	BPAGE
03	BNRY	BNRY	PAR2	-	CONFIG0	-
04	TSR	TPSR	PAR3	TPSR	CONFIG1	CONFIG1
05	NCR	TBCR0	PAR4	-	CONFIG2	CONFIG2
06	FIFO	TBCR1	PAR5	-	CONFIG3	CONFIG3
07	ISR	ISR	CURR	-	-	TEST
08	CRDA0	RSAR0	MAR0	-	CSNSAV	-
09	CRDA1	RSAR1	MAR1	-	-	HLTCLK
0A	8019ID0	RBCR0	MAR2	-	-	-
0B	8019ID1	RBCR1	MAR3	-	INTR	-
0C	RSR	RCR	MAR4	RCR	-	FMWP
0D	CNTR0	TCR	MAR5	TCR	CONFIG4	-
0E	CNTR1	DCR	MAR6	DCR	-	-
0F	CNTR2	IMR	MAR7	IMR	-	-
10-17	Puerto remoto DMA					
18-1F	Puerto Reset					

Tabla 4.6 Tabla de registros del RTL8019AS

Página 0 (PS1=0, PS0=0)

No	Nombre	Tipo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	CLDA0	R	A7	A6	A5	A4	A3	A2	A1	A0
	PSTART	W	A15	A14	A13	A12	A11	A10	A9	A8
02H	CLDA1	R	A15	A14	A13	A12	A11	A10	A9	A8
	PSTOP	W	A15	A14	A13	A12	A11	A10	A9	A8
03H	BNRY	R/W	A15	A14	A13	A12	A11	A10	A9	A8
04H	TSR	R	OWC	CDH	0	CRS	ABT	COL	-	PTX
	TPSR	W	A15	A14	A13	A12	A11	A10	A9	A8
05H	NCR	R	0	0	0	0	NC3	NC2	NC1	NC0
	TBCR0	W	TBC7	TBC6	TBC5	TBC4	TBC3	TBC2	TBC1	TBC0
06H	FIFO	R	D7	D6	D5	D4	D3	D2	D1	D0
	TBCR1	W	TBC15	TBC14	TBC13	TBC12	TBC11	TBC10	TBC9	TBC8
07H	ISR	R/W	RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX
08H	CRDA0	R	A7	A6	A5	A4	A3	A2	A1	A0
	RSAR0	W	A7	A6	A5	A4	A3	A2	A1	A0
09H	CRDA1	R	A15	A14	A13	A12	A11	A10	A9	A8
	RSAR1	W	A15	A14	A13	A12	A11	A10	A9	A8
0AH	8019ID0	R	0	1	0	1	0	0	0	0
	RBCR0	W	RBC7	RBC6	RBC5	RBC4	RBC3	RBC2	RBC1	RBC0
0BH	8019ID1	R	0	1	1	1	0	0	0	0
	RBCR1	W	RBC15	RBC14	RBC13	RBC12	RBC11	RBC10	RBC9	RBC8
0CH	RSR	R	DFR	DIS	PHY	MPA	0	FAE	CRC	PRX
	RCR	W	-	-	MON	PRO	AM	AB	AR	SEP
0DH	CNTR0	R	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	TCR	W	-	-	-	OFST	ATD	LB1	LB0	CRC
0EH	CNTR1	R	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	DCR	W	-	FT1	FT0	ARM	LS	LAS	BOS	WTS
0FH	CNTR2	R	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
	IMR	W	-	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Tabla 4.7 Página de registro 0 del RTL8019AS

Página 1 (PS1=0, PS0=1)

No	Nombre	Tipo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	PAR0	R/W	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
02H	PAR1	R/W	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
03H	PAR2	R/W	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
04H	PAR3	R/W	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
05H	PAR4	R/W	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
06H	PAR5	R/W	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40
07H	CURR	R/W	A15	A14	A13	A12	A11	A10	A9	A8
08H	MAR0	R/W	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
09H	MAR1	R/W	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
0AH	MAR2	R/W	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
0BH	MAR3	R/W	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
0CH	MAR4	R/W	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
0DH	MAR5	R/W	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
0EH	MAR6	R/W	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
0FH	MAR7	R/W	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

Tabla 4.8 Página de registro 1 del RTL8019AS

Página 2 (PS1=1, PS0=0)

No	Nombre	Tipo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	PSTART	R	A15	A14	A13	A12	A11	A10	A9	A8
02H	PSTOP	R	A15	A14	A13	A12	A11	A10	A9	A8
03H	-									
04H	TPSR	R	A15	A14	A13	A12	A11	A10	A9	A8
05H | 0BH	-									
0CH	RCR	R	-	-	MON	PRO	AM	AB	AR	SEP
0DH	TCR	R	-	-	-	OFST	ATD	LB1	LB0	CRC
0EH	DCR	R	-	FT1	FT0	ARM	LS	LAS	BOS	WTS
0FH	IMR	R	-	RDCE	CNTE	OVWE	TXEE	TXEE	PTXE	PRXE

Tabla 4.9 Página de registro 2 del RTL8019AS

Página 3 (PS1=1, PS0=1)

No	Nombre	Tipo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00H	CR	R/W	PS1	PS0	RD2	RD1	RD0	TXP	STA	STP
01H	9346CR	R	EEM1	EEM0	-	-	EECS	EESK	EEDI	EEDO
		W	EEM1	EEM0	-	-	EECS	EESK	EEDI	-
02H	BPAGE	R/W	BP7	BP6	BP5	BP4	BP3	BP2	BP1	BP0
03H	CONFIG0	R/W	VerID1	VerID0	AUI	PNPJP	JP	BNC	0	0
04H	CONFIG1	R	IRQEN	IRQS2	IRQS1	IRQS0	IOS3	IOS2	IOS1	IOS0
		W	IRQEN	-	-	-	-	-	-	-
05H	CONFIG2	R	PL1	PL0	BSELB	BS4	BS3	BS2	BS1	BS0

		W	PL1	PL0	BESLB	-	-	-	-	-
06H	CONFIG3	R	PNP	FUDUP	LEDS1	LEDS0	-	SLEEP	PWRDN	ACTIVEB
		W	-	-	-	-	-	SLEEP	PWRDN	-
07H	TEST	R/W	Reservado, no escribir							
08H	CSNSAV	R	CSN7	CSN6	CSN5	CSN4	CSN3	CSN2	CSN1	CSN0
09H	HLTCLK	W	HLT7	HLT6	HLT5	HLT4	HLT3	HLT2	HLT1	HLT0
0AH	-	-	Reservado							
0BH	INTR	R	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
0CH	FMWP	W*	Protector de escritura de la memoria flash							
0DH	CONFIG4	R	-	-	-	-	-	-	-	IOMS
0EH | 0FH	-	-	Reservado							

Tabla 4.10 Página de registro 3 del RTL8019AS

4.0.6 Funciones de Registros

Registro de Comandos
CR: Command Register (00H; Type=R/W)
Este registro es usado para seleccionar las páginas de registros, habilitar o deshabilitar las operaciones del DMA remoto y la emisión de comandos.

Registro de estado de interrupción
ISR: Interrupt Status Register (07H; Type=R/W in Page0)
Este registro refleja el estado de la NIC. El host lo lee para determinar la causa de la interrupción. Los bits individuales son limpiados escribiendo un “1” en el bit correspondiente. Este registro debe ser limpiado después del encendido.

Registro de Mascara de Interrupción
IMR: Interrupt Mask Register (0FH; Type=W in Page0, Type=R in Page2)
Todos los bits corresponden a los bits del registro ISR. En encendido todos se configuran a cero. Configurando los bits individuales habilitando las correspondientes interrupciones.

Registro de Configuración de Datos
DCR: Data Configuration Register (0EH; Type=W in Page0, Type=R in Page2)

Registro de Configuración de Transmisión
TCR: Transmit Configuration Register (0DH; Type=W in Page0, Type=R in Page2)

Registro de Estado de Transmisión
TSR: Transmit Status Register (04H; Type=R in Page0)
Este registro indica el estado de la transmisión de un paquete

Registro de Configuración de Recepción
RCR: Receive Configuration Register (0CH; Type=W in Page0, Type=R in Page2)

Registro de Estado de Recepción
RSR: Receive Status Register (0CH; Type=R in Page0)

Registros del DMA local actual
CLDA0, 1: Current Local DMA Registers (01H & 02H; Type=R in Page0)
Estos dos registros pueden ser leídos para obtener la dirección actual del DMA local.

Registro de Inicio de Página
PSTART: Page Start Register (01H; Type=W in Page0, Type=R in Page 2)
El registro de inicio de página configura la dirección de la página de inicio del buffer de anillo receptor.

Registro de alto de página
PSTOP: Page Stop Register (02H; Type=W in Page0, Type=R in Page2)
El registro de alto de página configura la dirección de alto de página del buffer de anillo receptor. En modo de 8 bits el registro PSTOP no debería exceder a 0x60, en modo de 16 bits el registro PSTOP no debería exceder a 0x80.

Registro límite
BNRY: Boundary Register (03H; Type=R/W in Page0)
Este registro es usado para prevenir la sobre escritura de el buffer de anillo. Es usado típicamente como un puntero que indica la última página del buffer recibida que el host ha leído.

Registro de Inicio de Página Transmitida
TPSR: Transmit Page Start Register (04H; Type=W in Page0)
Este registro configura la dirección de la página de inicio del paquete a ser transmitido

Registros de Contador de Byte Transmitido
TBCR0, 1: Transmit Byte Count Registers (05H & 06H; Type=W in Page0)
Estos dos registros configuran el contador de byte del paquete a ser transmitido

Registros de Dirección de Inicio Remoto
RSAR0, 1: Remote Start Address Registers (08H & 09H; Type=W in Page0)
Estos dos registros configuran la dirección de inicio del DMA remoto.

Registros de Contador de Bytes Remotos
RBCR0,1: Remote Byte Count Registers (0AH & 0BH; Type=W in Page0)
Estos dos registros configuran los contadores de bytes remotos del DMA remoto.

Registros de Dirección Física
PAR0-5: Physical Address Registers (01H - 06H; Type=R/W in Page1)
Estos registros contienen mi dirección del nodo Ethernet y son usados para comparar la dirección destino de un paquete entrante para aceptarlo o rechazarlo.

Registro de Página Actual
CURR: Current Page Register (07H; Type=R/W in Page1)
Este registro apunta a la dirección de la primera página del buffer recibida a ser usado para la recepción de un paquete.

Registro de Dirección Multicast
MAR0-7: Multicast Address Register (08H - 0FH; Type=R/W in Page1)
Estos registros proporcionan filtrado de direcciones multicast.

4.1 Microcontrolador PIC16F877

El microcontrolador le permite a un dispositivo independencia en su funcionamiento, una vez programado es el cerebro del sistema y si pierde la alimentación de energía no se borra su programación. Este IC es fabricado por Microchip, siendo uno de los PIC con mayor documentación en la Internet.
La ventaja del microcontrolador radica en que posee periféricos embebidos dentro del mismo IC, a diferencia de un microprocesador en el cual se tiene que desarrollar la lógica exterior para cada periférico requerido.

4.1.1 Características de Funcionamiento

- El microcontrolador PIC16F877 es un circuito integrado de 40 pines fabricado con tecnología CMOS, existen en diferentes tipos de encapsulado como por ejemplo DIP, SOIC y PLCC.
- La memoria programable es de 8Kx 14 palabras tipo FLASH, la memoria de datos (RAM) es de 368x 8 bytes y la memoria de datos no volátil (EEPROM) es de 256x 8 bytes.
- Capacidad de atender interrupciones por hardware y software
- Rango de voltaje de operación de 2.0 a 5.5 V y 25mA.
- La velocidad de operación máxima de reloj es de 20MHz, cada instrucción se ejecuta en dos ciclos de reloj por tanto, cada instrucción se ejecuta en 200ns.
- Posee Comunicación Síncrona SPI (Serial Peripheral Interfase) y Comunicación Asíncrona SCI (Serial Comunication Interfase).

4.1.2 Diagrama de pines

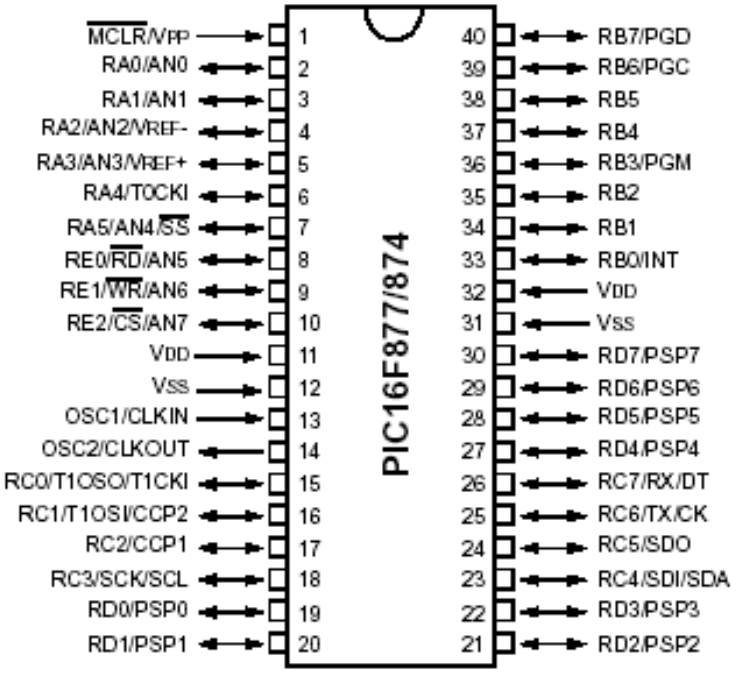

Figura 4.6 Diagrama de pines para el PIC16F877

4.1.3 Descripción de pines

Pin	Nemónico	Descripción
25, 26	RX, TX	Interfase de comunicación serial asíncrona SCI
23, 24, 18	SDI, SDO, SCK	Interfase de comunicación serial síncrona SPI
19-22, 27-30, 8-10	PSP0-PSP7, /RD, /WR, /CS	Interfase de comunicación paralela
1, 39, 40	/MCLR, PGD, PGC	Interfase de programación
1-5, 7-10	AN0 – AN7	Convertidor Analógico Digital
2-7	RA0 – RA5	Puerto A de datos bidireccional
33-40	RB0 – RB7	Puerto B de datos bidireccional
15-18, 23-26	RC0 – RC7	Puerto C de datos bidireccional
19-22, 27-30	RD0 – RD7	Puerto D de datos bidireccional
8-10	RE0 – RE2	Puerto E de datos bidireccional
33	INT	Petición de interrupción
13,14	CLKIN, CLKOUT	Entrada y salida de reloj respectivamente
11	VDD	Alimentación 5 V
12	VSS	Tierra digital.

Tabla 4.11 Descripción de pines del PIC16F877

4.1.4 Comunicación Asíncrona SCI

El SCI (Serial Comunications Interfase) esta formado por una unidad de transmisión y una unidad de recepción que son totalmente independientes, lo que permite que las comunicaciones sean bidireccionales, es decir, se permite la comunicación Full-duplex.
La unidad de información a transmitir está formada por un registro de un byte; al introducir un valor en este registro, comienza a desplazarse hacia la derecha el contenido del registro, enviando los bits por la línea de transmisión, a una velocidad que es configurada por el usuario. De la misma manera la unidad de recepción dispone de otro registro, que recibe los bits en serie y los va desplazando hasta obtener un dato en paralelo que puede ser leído.

Tanto el registro de transmisión como el de recepción están mapeados en la misma dirección de memoria, al escribir en esa dirección de memoria, el dato se cargara en el registro de transmisión. Al efectuar una lectura el dato se leerá del registro de recepción. Ambos registros comparten la misma dirección física de la memoria pero se trata de dos registros diferentes.

La interfase SCI reduce el número de líneas usadas para establecer una comunicación de datos, comparado con el uso del puerto paralelo. Solamente utiliza tres pines: el Transmisor de datos (TXD), el receptor de datos (RXD) y una línea de referencia denominada DGND (Tierra Digital).

4.1.5 Comunicación Síncrona SPI

La comunicación SPI (Serial Peripheral Interfase) es un sistema de comunicación serie síncrona de alta velocidad. El SPI puede ser utilizado para comunicar varios dispositivos entre sí, ya sean simples periféricos o varios microcontroladores.

Para realizar la comunicación, el Microcontrolador permite seleccionar entre dos modos de funcionamiento: **el modo maestro** y **el modo esclavo**. Cuando se realizan redes de comunicación (entre dos o más dispositivos) solamente esta permitido la existencia de un solo maestro, mientras que la de esclavos esta indefinida. Realmente su número viene dado por las necesidades del sistema en desarrollo.

La potencia de la interfase SCI llega al límite al permitir transmisiones full duplex (en ambos sentidos simultáneamente) entre un maestro y un esclavo. A partir de aquí, es posible realizar desde una simple comunicación unidireccional entre un microcontrolador y un periférico, hasta construir enlaces jerárquicos complejos entre un microcontrolador y/o periféricos.

Protocolo

Cuando el maestro tiene que mandar un mensaje a uno o varios esclavos deben proceder a realizar una selección de los mismos, trabajando al igual que si se tratara de un chip-enable. De esta forma, al ser activado el esclavo, recibe el dato manteniendo el sincronismo gracias a una señal de reloj conjunta. Es posible que cuando un esclavo sea activado con el fin de recibir un dato, este desee enviar una trama de respuesta al maestro, esto será posible mientras su línea de activación la mantenga el maestro, de modo que si es necesario la transmisión se efectuara simultáneamente en los dos sentidos.
Existen cuatro líneas asociados a la SPI mediante las cuales es posible montar los diferentes enlaces:

SDO (Serial Data Out): esta es la línea por donde circulan los datos que el maestro quiere enviar a los esclavos, por lo tanto será la señal de salida de datos de la unidad que funciona como maestro y la señal de entrada de datos para los esclavos.

SDI (Serial Data In): por esta línea viajarán los datos que sean enviados desde algún esclavo hacia el maestro, de esta forma será una señal de entrada para el maestro y las respectivas salidas para los esclavos.

SCK (Serial Clock): representa la señal de reloj con la que se producen las comunicaciones, si bien es posible que cada unidad configure mediante software la velocidad deseada, es previsible pensar que en una comunicación solo podrá prevalecer una, siendo esta la del maestro. Por lo tanto, para los esclavos representará una señal de entrada mientras que para el maestro será una salida.

/SS (Slave Select): esta línea tiene una funcionalidad muy concreta en las unidades esclavas ya que representa sus respectivas entradas de habilitación.
Es notable el ahorro de líneas de conexión que se genera en comparación con el puerto paralelo, donde es necesario como mínimo el cableado del bus de datos.

4.2 MAX232

4.2.1 Descripción

La familia de MAX220 a MAX249 de la línea de controladores / receptores está pensada para todas las interfases de comunicación EIA/TIA-232E and V.28/V.24. El MAX232 dispone de cuatro conversores de niveles TTL al bus estándar RS232 y viceversa, para comunicación serie. Lo interesante es que sólo necesita una alimentación de 5V, ya que genera internamente algunas tensiones que son necesarias para el estándar RS232. Otros integrados que manejan las líneas RS232 requieren dos voltajes, +12V y -12V.

El circuito integrado lleva internamente dos conversores de niveles TTL a RS232 y otros dos de RS232 a TTL, con lo que en total podemos manejar cuatro señales del puerto serie de la computadora, por lo general las más usadas son: TX, RX, RTS y CTS.

Para que el MAX232 funcione correctamente debemos colocar unos condensadores externos, esto lo podemos ver en la siguiente figura 4.8 en las que solo se han cableado las líneas TX y RX que son las que se usan más frecuentemente para casi cualquier aplicación.

Este integrado es usado para comunicar un microcontrolador (PIC16F877A) o sistema digital con la computadora o sistema basado en el bus serie RS232.

4.2.2 Diagrama de pines

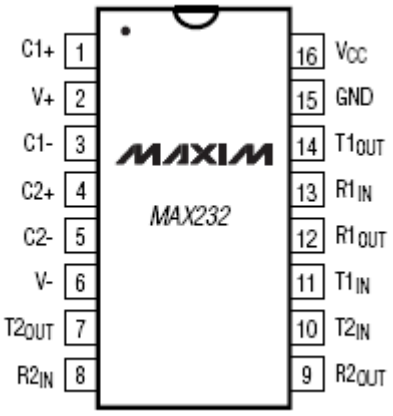

Figura 4.7 Diagrama de pines del MAX232

4.2.3 Descripción de pines

Pin	Nemónico	Descripción
1, 3, 4, 5	C1+, C1-, C2+, C2-	Capacitores
16	VCC	Alimentación 5V
15	GND	Tierra
13, 8	$R1_{IN}$, $R2_{IN}$	Entradas RS232
14, 7	$T1_{OUT}$, $T2_{OUT}$	Salidas RS232
12, 9	$R1_{OUT}$, $R2_{OUT}$	Salidas TTL / CMOS
11, 10	$T1_{IN}$, $T2_{IN}$	Entradas TTL / CMOS
2, 6	V+, V-	Alimentación

Tabla 4.12 Descripción de pines del MAX232

4.2.4 Circuito de operación típica

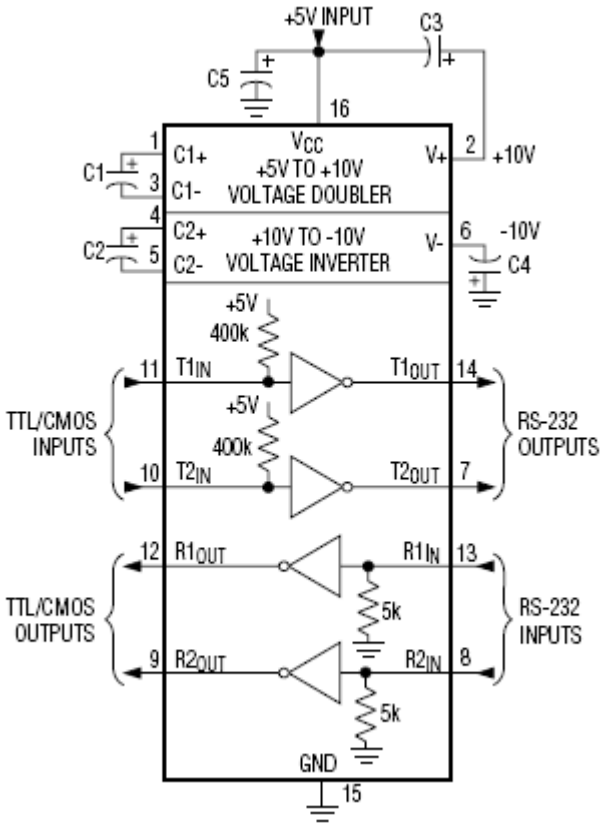

Figura 4.8 Circuito de operación típica del MAX232

Capacitor	Valor
C1	1.0 µF
C2	1.0 µF
C3	1.0 µF
C4	1.0 µF
C5	1.0 µF

Tabla 4.13 Valores de C para una operación típica del MAX232

CAPÍTULO V

HARDWARE Y SOFTWARE

Introducción

El objetivo final del mismo es diseñar e implementar una tarjeta de comunicación que permita monitorear remotamente los paramentos del medidor de energía creado en la escuela de Ingeniería Eléctrica.
Para la creación de la tarjeta se realizo una selección de los IC a utilizar en el mismo, que fueron descritos en el capitulo cuatro. Como se menciono, el RTL8019AS presenta dificultades para soldarse por el tipo de empaquetamiento, debido a esta razón se compro un RTL8019AS embebido en una tarjeta que además de brindar un pin-out más amigable, posee la polarización del RTL y un jack RJ45.
Dicha tarjeta que posee embebido el RTL8019AS es distribuida por EDTP Electronics y es llamada *Packet Whacker*. Este dispositivo se acoplara al hardware creado (tarjeta), para facilitar su implementación.

El software principal desarrollado comprende el programa descargado en el PIC16F877 para el manejo de los protocolos involucrados en la comunicación sobre Ethernet y sobre el medio serial. En segundo plano esta el software para el monitoreo de los parámetros eléctricos del medidor, desarrollado en el cliente, que se creó en Labview.

5.0 Packet Whacker

El Packet Whacker es el dispositivo más básico distribuido por EDTP Electronics. Este dispositivo es diseñando para trabajar con líneas de I/O de un microcontrolador externo. El microcontrolador corre un programa que inicializa el RTL8019AS y controla el flujo hacia y desde Ethernet.
Una de las razones principales por las que se opto por adquirir esta tarjeta es por el tipo de empaquetamiento que presenta el RTL8019AS, para lo cual es recomendable comprar una tarjeta como la *Packet Whacker* que posea embebido el RTL y además brinde un pin-out más amigable.

El packet Whacker incluye:

- Ethernet IC RTL8019AS
- Cristal de 20MHz
- Magnetos aislados con jack RJ45
- 12 capacitores y resistores necesarios para polarización.

Las características principales son:

- Bajo Precio
- Ampliamente usado con microcontroladores
- RJ-45 integrado y magnetos
- Pin-out de 0.1 pulgadas
- Operación 10BaseT
- LED´s de estado

El Packet Whacker se muestra en la figura 5.1.

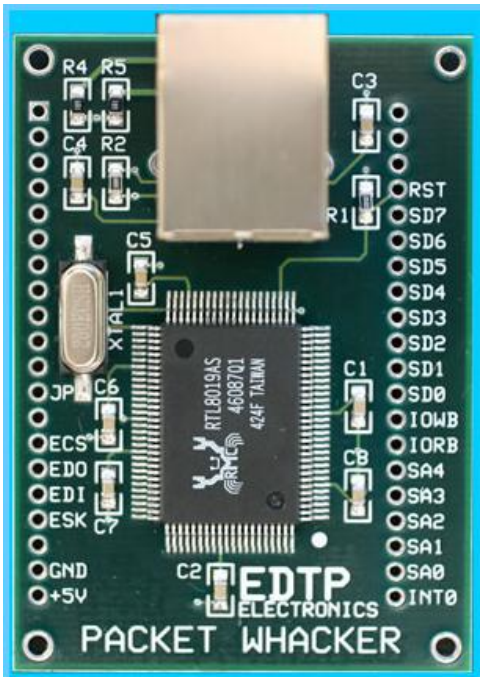

Figura 5.1. RTL8019AS embebido en PACKET WHACKER

5.0 Diagrama de bloques

Los IC principales en el diagrama de bloques son el PIC16F877 y el PACKET WHACKET, los cuales son la base para la comunicación. Alrededor de ellos existirán otros elementos como memoria, puertos RS232, etc. El diagrama de bloques se muestra en la figura 5.2.

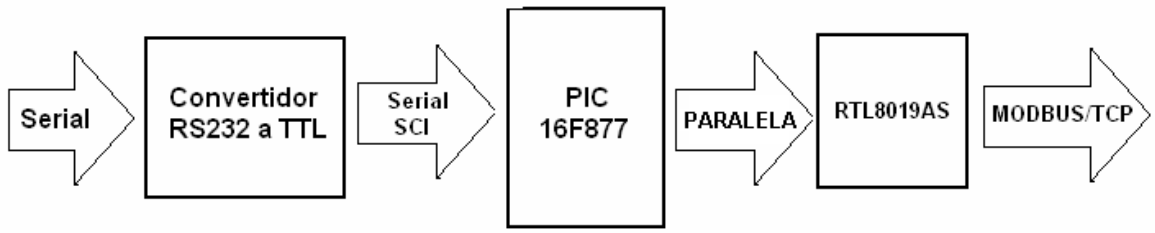

Figura 5.2 Diagrama de bloques para la tarjeta a desarrollar

En la figura anterior el RTL8019AS representa el Packet Whacket. Al final la tarjeta a desarrollar formara parte de un sistema como se muestra en la figura 5.3, creando el puente de enlace entre la comunicación serial RS232 y Ethernet.

Este sistema involucra manejar tanto comunicación TCP/IP a través de Ethernet como serial por el puerto RS232.

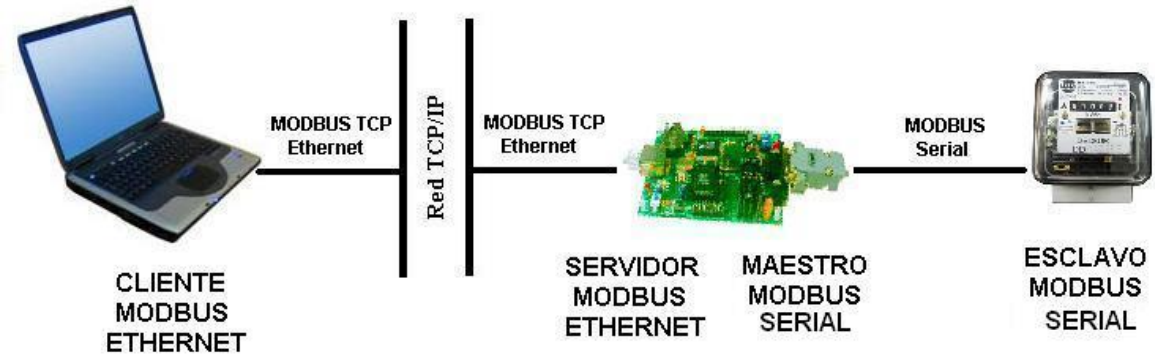

Figura 5.3 Sistema completo de monitoreo de parámetros del medidor de energía

5.1 Diagrama de conexión

El diagrama de conexión final de la tarjeta, se muestra en la figura 5.4

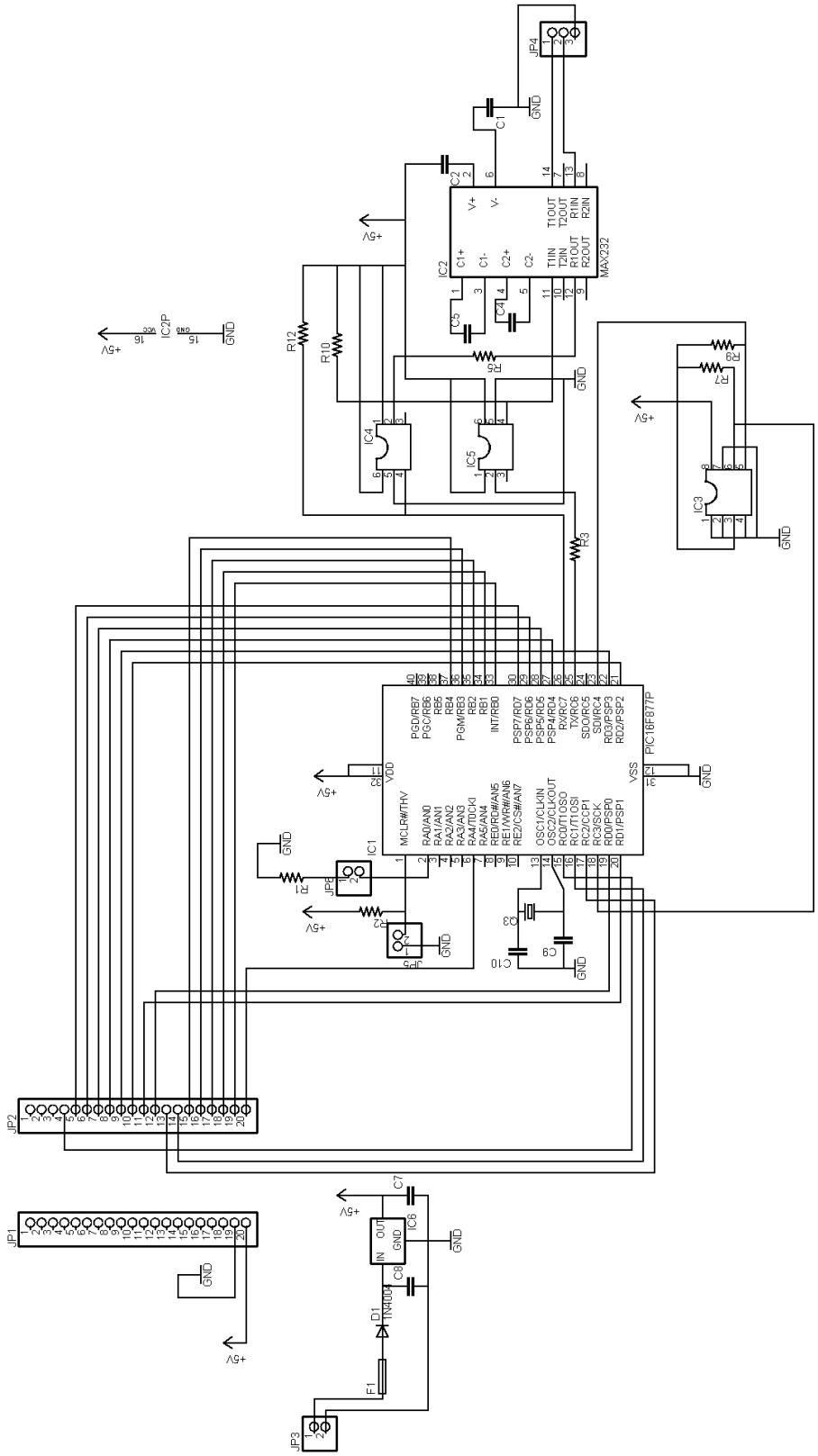

Figura 5.4 Diagrama final de conexión del sistema de monitoreo de parámetros del medidor de energía

5.2 Software del sistema de comunicación del medidor

5.2.1 Software en la tarjeta (PIC16F877)

Se basa en el desarrollo de un programa en lenguaje ensamblador que implementa la lógica de la pila de protocolos que trabajan sobre una red Ethernet y sobre la línea serial.
El programa maneja el flujo de datos desde el medidor hasta la computadora por medio de la tarjeta, el medidor envía los datos por medio del puerto RS232, pasando por la tarjeta y está lo envía hasta la red Ethernet hasta llegar a la PC.

Para llevar esto a cabo se ha implementado MODBUS sobre el medio serial y sobre TCP/IP. Dentro de estos protocolos se ha implementado el Servidor MODBUS TCP/IP y el Maestro MODBUS serial. El programa además controlara el funcionamiento del RTL8019AS por medio de la configuración de sus registros, los cuales ayudaran a implementar la lógica TCP/IP sobre Ethernet.

5.2.2 Software en la computadora (Interfase Gráfica LabVIEW)

El software en el cliente se encarga de la recepción y presentación de los datos adquiridos desde el medidor de energía. Debido a que se esta utilizando un protocolo de comunicación estándar (MODBUS TCP) y que es ampliamente usado a nivel mundial, se ha desarrollado un software utilizando librerías proporcionadas por una institución ampliamente reconocida en el ámbito industrial y de distribución a nivel internacional. Esto confirmaría la correcta implementación del protocolo MODBUS, y garantizaría su correcto funcionamiento. Este software se ha desarrollado en el programa LabVIEW, utilizando librerías MODBUS desarrolladas por National Instruments.

5.2.3 Software en el medidor (PIC16F877)

El software del medidor es prácticamente la modificación del programa en lenguaje ensamblador de la tesis previa “Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754” que implementa todas las funciones del medidor. Este programa se ha modificado ya que el protocolo de comunicación que estaba implementado en el medidor era un protocolo propietario de los que desarrollaron el medidor, y ya que estamos utilizando

protocolos estándar, se ha sustituido el protocolo propietario y en su lugar se ha implementado MODBUS sobre línea serial, específicamente un esclavo MODBUS serial.

5.3 Interfase Gráfica

5.3.1 El lenguaje de programación LabVIEW

LabVIEW es un lenguaje de programación gráfico altamente productivo para construir sistemas de adquisición de datos e instrumentación. Con este programa es posible crear interfases gráficas para instrumentos. Ver figura 5.5.

Una de las ventajas más grandes de LabVIEW es la ganancia de productividad que ofrece; porque se tiene la posibilidad de cambiar de manera fácil el código en un tiempo corto. LabVIEW ha llegado a ser una herramienta estándar en la industria para: pruebas y medición, control de procesos, automatización de fábricas, monitoreo de máquina, investigaciones y análisis.

Figura 5.5 Lenguaje de Programación Gráfica LabVIEW

Los Instrumentos Virtuales (VI) de LabVIEW son modulares en su diseño, así cualquier VI puede ejecutarse individualmente o ser usado como parte de otro VI. Es posible crear iconos para los VI's propios.

5.3.2 Flujograma de la interfase gráfica del usuario

5.3.2.1 Programa Principal

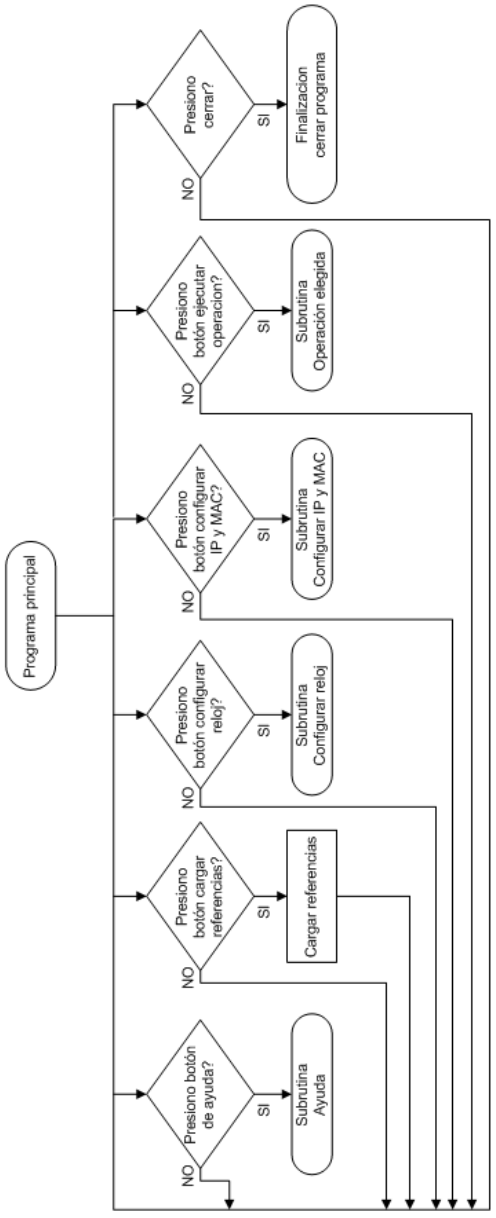

Figura 5.6 Flujograma del programa principal

5.3.2.2 Subrutina para leer datos actuales del medidor

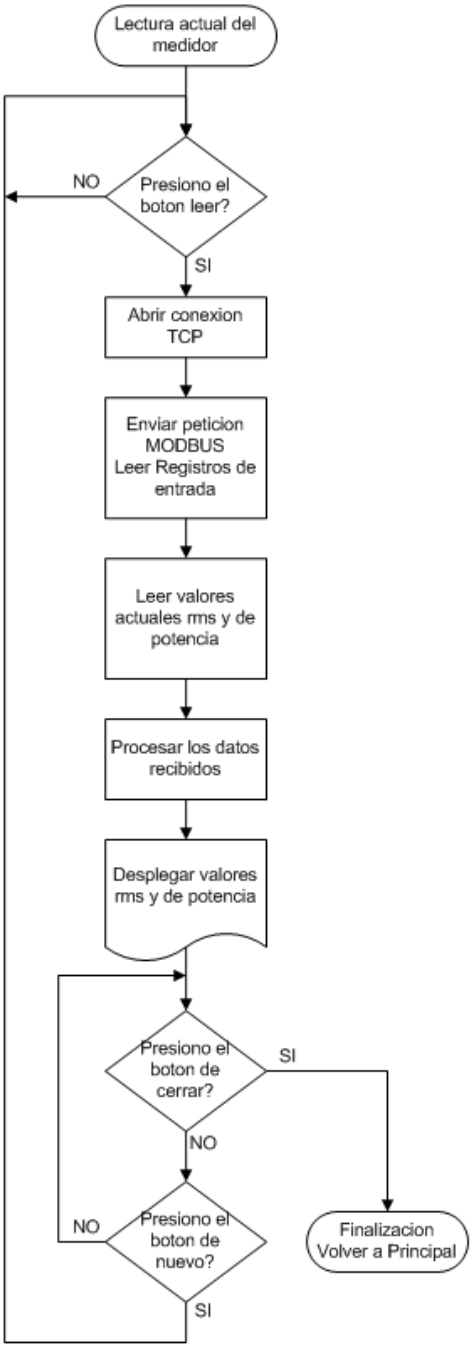

Figura 5.7 Flujograma de subrutina de lectura de datos actuales del medidor

5.3.2.3 Subrutina de descarga del medidor de valores rms y de potencia

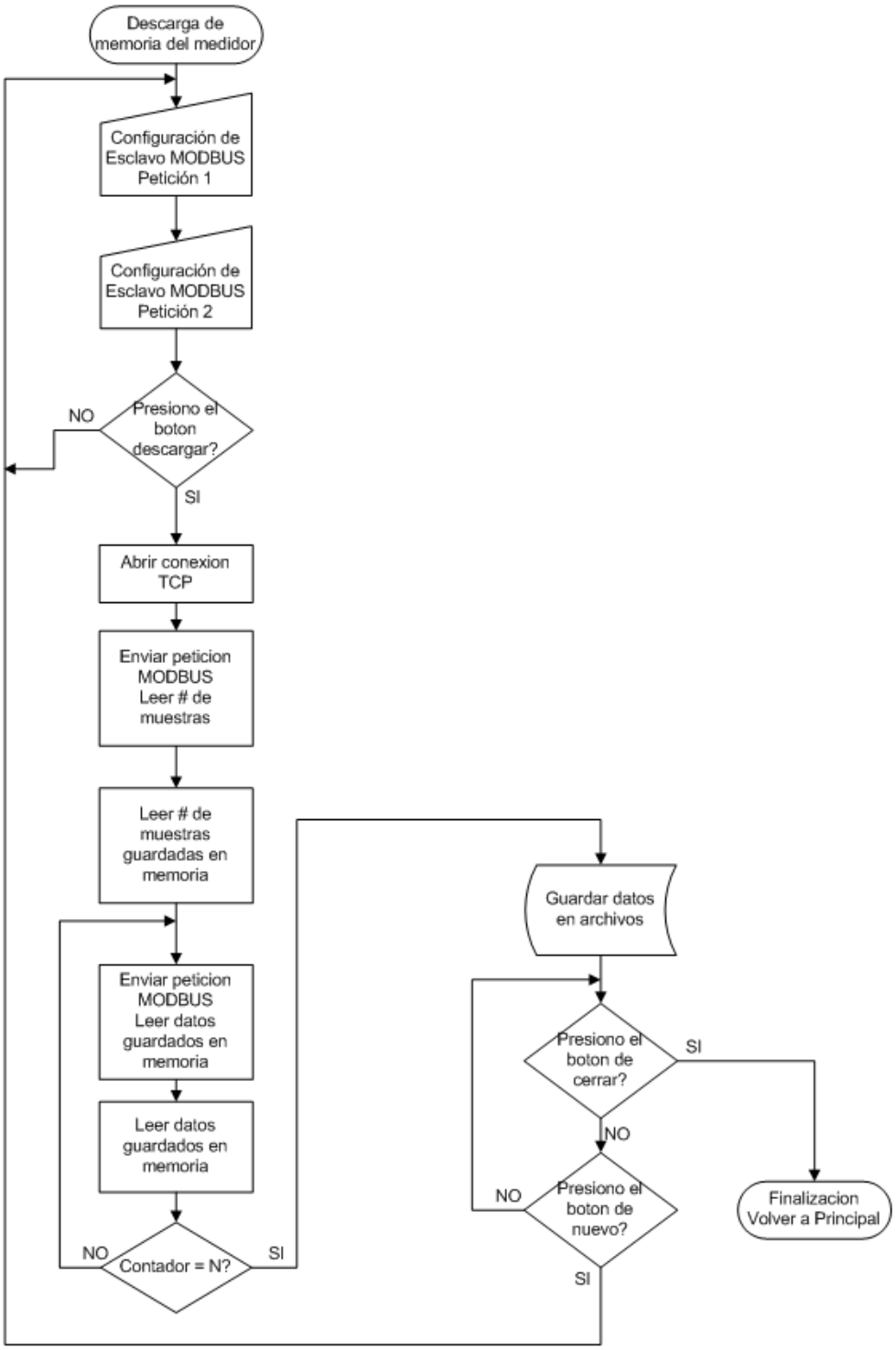

Figura 5.8 Flujograma de subrutina de descarga de memoria del medidor de valores rms y de potencia

5.3.2.4 Subrutina de generación de repote HTML

Figura 5.9 Flujograma de subrutina de generación de reporte HTML

5.3.3 Interfase gráfica del usuario

La Interfase gráfica del usuario (GAU) es desarrollada bajo el lenguaje de programación de alto nivel LabVIEW.
Un VI consta de dos partes principales: un panel frontal y un diagrama de bloques. El panel frontal sirve como la interfase para el usuario, aquí se colocan los controles y las plantillas de salida de datos del sistema. El diagrama de bloques es la estructura del programa. Para construir el VI se seleccionan los objetos y se utilizan los conectores para unirlos entre si; en un lenguaje de programación convencional se escriben muchas líneas de código, en LabVIEW cada icono representa una función.
Los VI´s que conforman la interfase son:

- Programa Principal
- Programa de calibración

Pero en este documento solo se abordará el programa principal

5.3.3.1 VI Principal

En este se puede escoger la operación a realizar, introducir la relación de los transformadores de corriente (TI) y de los transformadores de tensión (TT), ajustar el reloj del medidor, configurar la IP y MAC de la tarjeta, configurar el puerto remoto (puerto 502 MODBUS), acceder a la ayuda o cerrar la interfase gráfica.

Entre las operaciones a realizar están las siguientes:

- Lectura actual del medidor de valores rms y de potencia
- Descarga del medidor de valores rms y de potencia
- Generación de gráficas de energía.
- Generación de reporte HTML

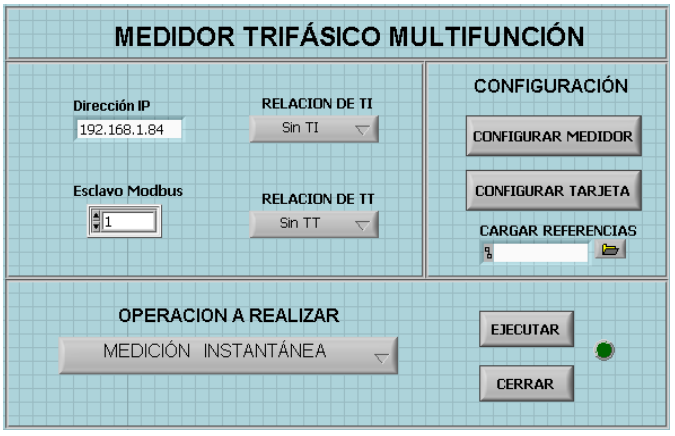

Figura 5.10 VI Principal

5.3.3.2 VI de lectura actual del medidor de valores rms y de potencia

Con este VI es posible obtener los valores actuales de cada uno de los seis canales analógicos, los tres de voltaje y los tres de corriente; además de los valores actuales de potencia activa, aparente y reactiva del medidor. Se puede elegir la IP del Instrumento a monitorear, como el puerto remoto, el esclavo y el Timeout.

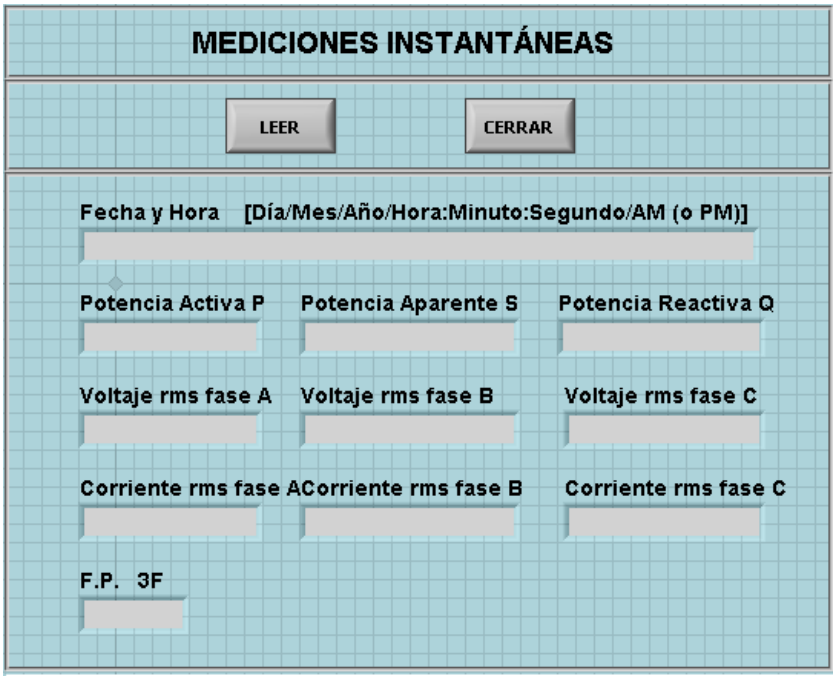

Figura 5.14 VI Lectura actual del medidor de valores rms y de potencia

5.3.4 VI de descarga del medidor de valores rms y de potencia

Con este VI es posible descargar el contenido de la memoria utilizada, de 128Kx8 bits disponibles, en un archivo de texto. Posee una pantalla de salida que muestra el %Terminado.

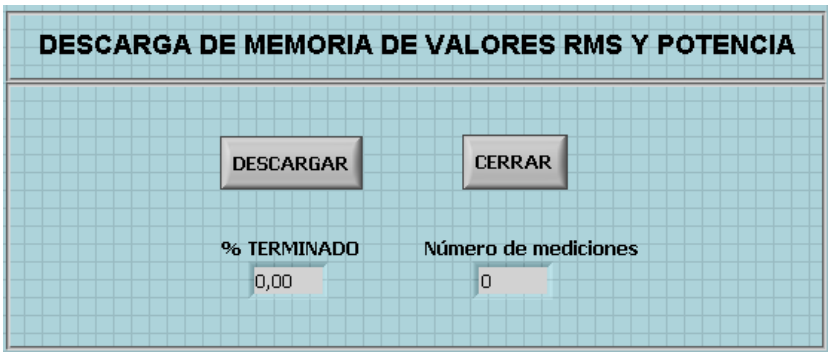

Figura 5.17 VI Descarga de valores rms y de potencia

5.3.5 VI de generación de reporte HTML

Con este VI es posible crear archivos de reporte HTML, a partir de un archivo de texto previamente descargado de memoria.

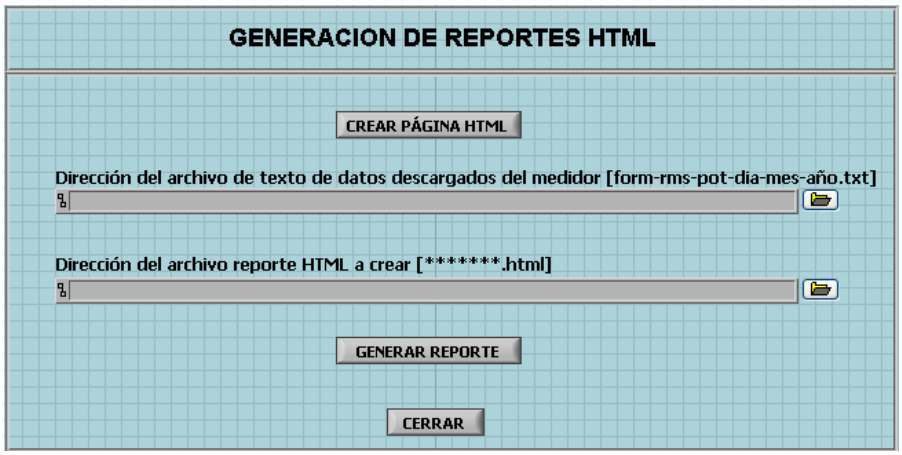

Figura 5.18 VI Generación de reportes HTML

5.3.6 VI de generación de Gráficas de Energía

Este VI se utiliza para desplegar las distintas gráficas de energía tanto activa, reactiva, como aparente estas gráficas, así como los valores máximos y mínimos de los parámetros del medidor y todo esto se obtiene a partir de los datos descargados de la memoria.

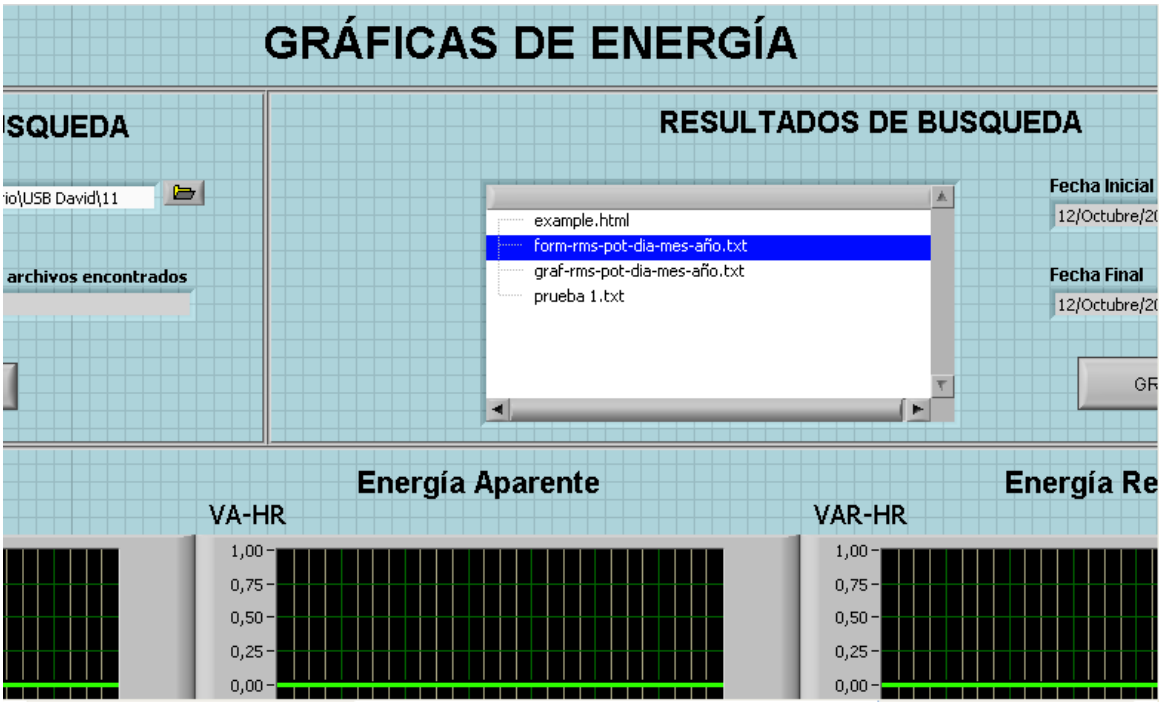

Figura 5.19 VI Gráfica de energía

RECOMENDACIONES

- ☑ Es importante hacer énfasis en la necesidad de utilización de protocolos de comunicación estándar en los trabajos y proyectos de la Escuela de Ingeniería Eléctrica, para que estos no queden aislados de los instrumentos y software comerciales.

- ☑ Para mejorar la tarjeta se puede implementar un servidor web embebido, para que tenga acceso directo al dispositivo a través de un navegador de Internet, tal como Microsoft Internet Explorer o Netscape Navigator.

- ☑ Utilizar el IC ENC28J60 en sustitución del RTL8019AS, debido a bajo costo y reducción en las líneas de comunicación.

- ☑ Utilizar la familia de PIC18F en sustitución del PIC16F, debido a los requerimientos de memoria de programa y memoria de datos necesarios para la implementación de los protocolos TCP/IP. Además de la posibilidad de incrementar la velocidad del cristal arriba de los 20 MHz limites del PIC16F877A.

CONCLUSIONES

- ☑ La utilización de protocolos estándar fomentará la fácil interconexión de distintos dispositivos o instrumentos, que se desarrollen en la Escuela de Ingeniería Eléctrica, y con equipos y software de otras partes del mundo que hablen este mismo protocolo (Modbus/TCP, Modbus Serial).

- ☑ Debido a que Modbus tanto TCP como serial, son protocolos de distribución gratuita, y que su aceptación esta creciendo grandemente en distintos entornos especialmente en redes Ethernet, son los protocolos ideales para ser implementados en futuros trabajo o proyectos, y es el que se implementó en este trabajo.

- ☑ El desarrollo del stack TCP/IP sobre el PIC16F877A teniendo como interfaz el RTL8019AS brinda una velocidad de procesamiento de paquetes aceptable, pero que puede ser mejorada emigrando a la familia de microcontroladores PIC18F.

- ☑ Para demostrar que la tarjeta “habla” un protocolo estándar, en este caso el protocolo Modbus/TCP, se implementó una interfase en un programa que interprete dichos protocolos como lo es LabVIEW de Nacional Instrument, garantizando así un correcto funcionamiento.

- ☑ Para aumentar la velocidad de transferencia de datos desde el dispositivo remoto se debe implementar comunicación TCP/IP directamente sin pasar por el medio serial, ya que éste ultimo limita la velocidad y la cantidad de datos soportados en una sola transferencia.

REFERENCIAS BIBLIOGRÁFICAS

[1] Arévalo Navas, Fernando Alberto; Cortez Franco, Daniel Antonio; López Hernández, Douglas Alberto. **"Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754"**. Tesis para optar al grado de: Ingeniero Electricista, Biblioteca de las Ingenierías, Universidad de El Salvador, 2005.

[2] IEEE Std 802.3™-2002 (Revision of IEEE Std 802.3, 2000 Edition). "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications"

[3] "MODBUS Aplication Protocol Specification V1.1a"; http://www.modbus.org/

[4] "MODBUS Messaging on TCP/IP Implementation Guide V1.0a"; http://www.modbus.org/

[5] "MODBUS over Serial Line Specification & Implementation guide V1.0"; http://www.modbus.org/

[6] Postel, J. (ed.), "Transmission Control Protocol - DARPA Internet Program Protocol Specification ", RFC 793, USC/Information Sciences Institute, Septiembre 1981.

[7] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, Septiembre 1981. (N.T. Versión en castellano por P.J. Ponce de León: "Protocolo Internet", Mayo 1999)

[8] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", RFC 792, USC/Information Sciences Institute, Septiembre 1981.

ANEXOS

A.1 Mapeo de Registros para el Protocolo MODBUS/TCP

Lectura actual del medidor de valores rms y de potencia

Registros 0x0000 – 0x0015

Registros	Bits																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0000	————								Potencia Activa P								MSB
0001	Potencia Activa P								Potencia Activa P								LSB
0002	————								Potencia Aparente S								MSB
0003	Potencia Aparente S								Potencia Aparente S								LSB
0004	————								Potencia Reactiva Q								MSB
0005	Potencia Reactiva Q								Potencia Reactiva Q								LSB
0006	————								Voltaje rms fase A								MSB
0007	Voltaje rms fase A								Voltaje rms fase A								LSB
0008	————								Voltaje rms fase B								MSB
0009	Voltaje rms fase B								Voltaje rms fase B								LSB
000A	————								Voltaje rms fase C								MSB
000B	Voltaje rms fase C								Voltaje rms fase C								LSB
000C	————								Corriente rms fase A								MSB
000D	Corriente rms fase A								Corriente rms fase A								LSB
000E	————								Corriente rms fase B								MSB
000F	Corriente rms fase B								Corriente rms fase B								LSB
0010	————								Corriente rms fase C								MSB
0011	Corriente rms fase C								Corriente rms fase C								LSB
0012	Fecha y Hora								Fecha y Hora								MSB
0013	Fecha y Hora								Fecha y Hora								-
0014	Fecha y Hora								Fecha y Hora								-
0015	Fecha y Hora								Fecha y Hora								LSB

Tabla A.1.1 Mapeo de registros de lectura actual

Contador

Registros 0x0016 – 0x0017

Registros	Bits																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0016	Contador 3								Contador 2								MSB
0017	Contador 1								Contador 0								LSB

Tabla A.1.2 Mapeo de registros del contador

No de esclavo Serial

Registro 0x0018

Registros	Bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0018	Esclavo serial								Esclavo serial							

Tabla A.1.3 Mapeo de registros del número de esclavo serial MODBUS

Calibración

Registros 0x0019 – 0x004C

Registros	Bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0019	Bandera								Bandera							
001A	AWG1_MSB								AWG1_LSB							
001B	AWG0_MSB								AWG0_LSB							
001C	BWG1_MSB								BWG1_LSB							
001D	BWG0_MSB								BWG0_LSB							
001E	CWG1_MSB								CWG1_LSB							
001F	CWG0_MSB								CWG0_LSB							
0020	AVAG1_MSB								AVAG1_LSB							
0021	AVAG0_MSB								AVAG0_LSB							
0022	BVAG1_MSB								BVAG1_LSB							
0023	BVAG0_MSB								BVAG0_LSB							
0024	CVAG1_MSB								CVAG1_LSB							
0025	CVAG0_MSB								CVAG0_LSB							
0026	APHCAL0_MSB								APHCAL0_LSB							
0027	BPHCAL0_MSB								BPHCAL0_LSB							
0028	CPHCAL0_MSB								CPHCAL0_LSB							
0029	AAPOS1_MSB								AAPOS1_LSB							

002A	AAPOS0_MSB	AAPOS0_LSB
002B	BAPOS1_MSB	BAPOS1_LSB
002C	BAPOS0_MSB	BAPOS0_LSB
002D	CAPOS1_MSB	CAPOS1_LSB
002E	CAPOS0_MSB	CAPOS0_LSB
002F	LINCYC1_MSB	LINCYC1_LSB
0030	LINCYC0_MSB	LINCYC0_LSB
0031	CFNUM1_MSB	CFNUM1_LSB
0032	CFNUM0_MSB	CFNUM0_LSB
0033	CFDEN1_MSB	CFDEN1_LSB
0034	CFDEN0_MSB	CFDEN0_LSB
0035	AIRMSOS_1_MSB	AIRMSOS_1_LSB
0036	AIRMSOS_0_MSB	AIRMSOS_0_LSB
0037	BIRMSOS_1_MSB	BIRMSOS_1_LSB
0038	BIRMSOS_0_MSB	BIRMSOS_0_LSB
0039	CIRMSOS_1_MSB	CIRMSOS_1_LSB
003A	CIRMSOS_0_MSB	CIRMSOS_0_LSB
003B	AVRMSOS_1_MSB	AVRMSOS_1_LSB
003C	AVRMSOS_0_MSB	AVRMSOS_0_LSB
003D	BVRMSOS_1_MSB	BVRMSOS_1_LSB
003D	BVRMSOS_0_MSB	BVRMSOS_0_LSB
003F	CVRMSOS_1_MSB	CVRMSOS_1_LSB
0040	CVRMSOS_0_MSB	CVRMSOS_0_LSB
0041	AAPGAIN1_MSB	AAPGAIN1_LSB
0042	AAPGAIN0_MSB	AAPGAIN0_LSB
0043	BAPGAIN1_MSB	BAPGAIN1_LSB
0044	BAPGAIN0_MSB	BAPGAIN0_LSB
0045	CAPGAIN1_MSB	CAPGAIN1_LSB
0046	CAPGAIN0_MSB	CAPGAIN0_LSB
0047	AVGAIN1_MSB	AVGAIN1_LSB
0048	AVGAIN0_MSB	AVGAIN0_LSB
0049	BVGAIN1_MSB	BVGAIN1_LSB
004A	BVGAIN0_MSB	BVGAIN0_LSB
004B	CVGAIN1_MSB	CVGAIN1_LSB
004C	CVGAIN0_MSB	CVGAIN0_LSB

Tabla A.1.4 Mapeo de registros de Calibración

Descarga de memoria del medidor

Registros 0x004D – 0xFFF3

Bloque n. Registros (0x004D + 0x0016*n) – (0x0062 + 0x0016*n)

Donde 0 ≤ n ≤ 0XB9D (2973 en decimal)

Registros	Bits																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
004D+16*n	————								Potencia Activa P								MSB
004E+16*n	Potencia Activa P								Potencia Activa P								LSB
004F+16*n	————								Potencia Aparente S								MSB
0050+16*n	Potencia Aparente S								Potencia Aparente S								LSB
0051+16*n	————								Potencia Reactiva Q								MSB
0052+16*n	Potencia Reactiva Q								Potencia Reactiva Q								LSB
0053+16*n	————								Voltaje rms fase A								MSB
0054+16*n	Voltaje rms fase A								Voltaje rms fase A								LSB
0055+16*n	————								Voltaje rms fase B								MSB
0056+16*n	Voltaje rms fase B								Voltaje rms fase B								LSB
0057+16*n	————								Voltaje rms fase C								MSB
0058+16*n	Voltaje rms fase C								Voltaje rms fase C								LSB
0059+16*n	————								Corriente rms fase A								MSB
005A+16*n	Corriente rms fase A								Corriente rms fase A								LSB
005B+16*n	————								Corriente rms fase B								MSB
005C+16*n	Corriente rms fase B								Corriente rms fase B								LSB
005D+16*n	————								Corriente rms fase C								MSB
005E+16*n	Corriente rms fase C								Corriente rms fase C								LSB
005F+16*n	Fecha y Hora								Fecha y Hora								MSB
0060+16*n	Fecha y Hora								Fecha y Hora								-
0061+16*n	Fecha y Hora								Fecha y Hora								-
0062+16*n	Fecha y Hora								Fecha y Hora								LSB

Tabla A.1.5 Mapeo de registros de descarga de datos del medidor

Bloque n = 0x0. Registros 0x004D – 0x0062

Registros	Bits																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
004D	————								Potencia Activa P								MSB
004E	Potencia Activa P								Potencia Activa P								LSB
004F	————								Potencia Aparente S								MSB
0050	Potencia Aparente S								Potencia Aparente S								LSB
0051	————								Potencia Reactiva Q								MSB
0052	Potencia Reactiva Q								Potencia Reactiva Q								LSB
0053	————								Voltaje rms fase A								MSB
0054	Voltaje rms fase A								Voltaje rms fase A								LSB
0055	————								Voltaje rms fase B								MSB
0056	Voltaje rms fase B								Voltaje rms fase B								LSB
0057	————								Voltaje rms fase C								MSB

0058	Voltaje rms fase C	Voltaje rms fase C	LSB
0059	————————	Corriente rms fase A	MSB
005A	Corriente rms fase A	Corriente rms fase A	LSB
005B	————————	Corriente rms fase B	MSB
005C	Corriente rms fase B	Corriente rms fase B	LSB
005D	————————	Corriente rms fase C	MSB
005E	Corriente rms fase C	Corriente rms fase C	LSB
005F	Fecha y Hora	Fecha y Hora	MSB
0060	Fecha y Hora	Fecha y Hora	-
0061	Fecha y Hora	Fecha y Hora	-
0062	Fecha y Hora	Fecha y Hora	LSB

Tabla A.1.6 Mapeo de registros de descarga de datos del medidor con n=0x0

Bloque n = 0xB9E. Registros 0xFFE1 – 0xFFF6

Registros	Bits																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FFE1	————————								Potencia Activa P								MSB
FFE2	Potencia Activa P								Potencia Activa P								LSB
FFE3	————————								Potencia Aparente S								MSB
FFE4	Potencia Aparente S								Potencia Aparente S								LSB
FFE5	————————								Potencia Reactiva Q								MSB
FFE6	Potencia Reactiva Q								Potencia Reactiva Q								LSB
FFE7	————————								Voltaje rms fase A								MSB
FFE8	Voltaje rms fase A								Voltaje rms fase A								LSB
FFE9	————————								Voltaje rms fase B								MSB
FFEA	Voltaje rms fase B								Voltaje rms fas								LSB
FFEB	————————								Voltaje rms fase C								MSB
FFEC	Voltaje rms fase C								Voltaje rms fase C								LSB
FFED	————————								Corriente rms fase A								MSB
FFEE	Corriente rms fase A								Corriente rms fase A								LSB
FFEF	————————								Corriente rms fase B								MSB
FFF0	Corriente rms fase B								Corriente rms fase B								LSB
FFF1	————————								Corriente rms fase C								MSB
FFF2	Corriente rms fase C								Corriente rms fase C								LSB
FFF3	Fecha y Hora								Fecha y Hora								MSB
FFF4	Fecha y Hora								Fecha y Hora								-
FFF5	Fecha y Hora								Fecha y Hora								-
FFF6	Fecha y Hora								Fecha y Hora								LSB

Tabla A.1.7 Mapeo de registros de descarga de datos del medidor con n=0xB9E

Registros reservados

Registros 0XFFF7 – 0xFFFF

Registros	Bits															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFF7	**RESERVADO PARA USO FUTURO**															
FFF8																
FFF9																
FFFA																
FFFB																
FFFC																
FFFD																
FFFE																
FFFF																

Tabla A.1.8 Mapeo de registros de registros reservados para uso futuro

A.2 Manual de usuario

MANUAL DE USUARIO DEL SOFTWARE DEL MEDIDOR

Para facilitar al usuario la utilización del software del medidor, se ha elaborado este manual del usuario, aunque el software provee una ayuda en cada uno de los VI´s que puede ser usado para manejar la interfase gráfica sin necesidad de ver este manual de usuario, aunque, este manual provee una explicación más amplia del uso de la interfase.
El software del medidor fue hecho en el lenguaje de programación de alto nivel LabVIEW. Las aplicaciones desarrolladas son dos: el VI de Calibración y el VI Principal.

VI DE CALIBRACIÓN

Con esta aplicación es posible ajustar el valor del pulso de frecuencia de salida CF, mediante la configuración de los registros del ADE7754.

Como se puede apreciar en la figura A1.1, este VI contiene entradas, salidas y botones. Las salidas de texto son los cuadros de fondo gris que tienen números en su interior; así como las entradas de texto se pueden distinguir por tener un fondo blanco. Como se puede observar en el VI, existen dos columnas para cada registro de calibración, de izquierda a derecha, la primera columna proporciona los valores de los registros configurados o los datos guardados de los registros en la EEPROM por default, la segunda columna es para escribir los nuevos valores para los registros que luego aparecerán en la primera columna, cuando se vuelva a leer los valores actuales.

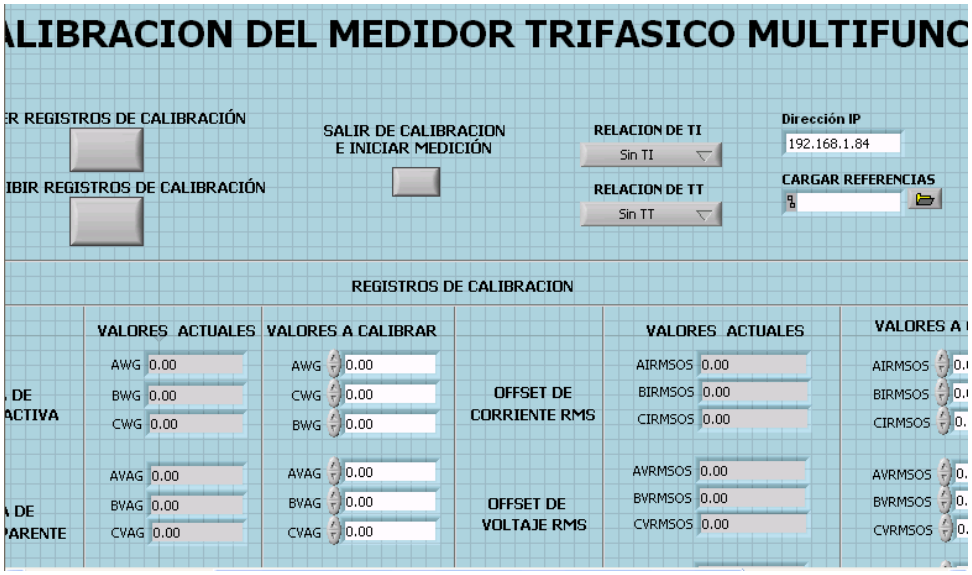

Figura A.2.1 VI de calibración del medidor

Para una mejor comprensión, se muestran secciones de este VI para poder apreciar más claramente los detalles. La barra de herramientas proporciona varias opciones, de las cuales nos interesa el botón RUN, que sirve para ejecutar el VI, este botón es una flecha que apunta a la derecha.

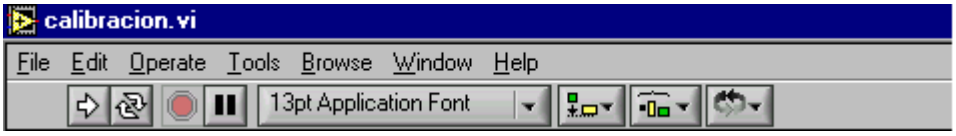

Figura A.2.2 Barra de ejecución de LabVIEW

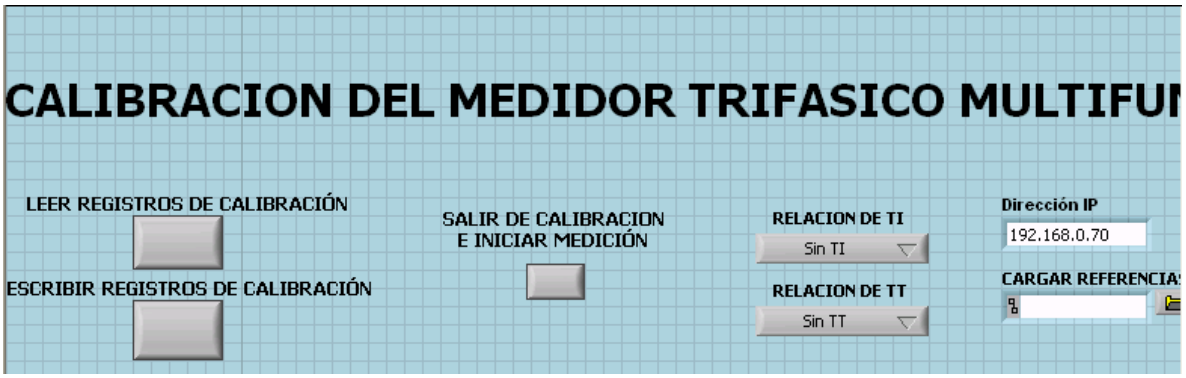

Figura A.2.3 VI de calibración sección 1

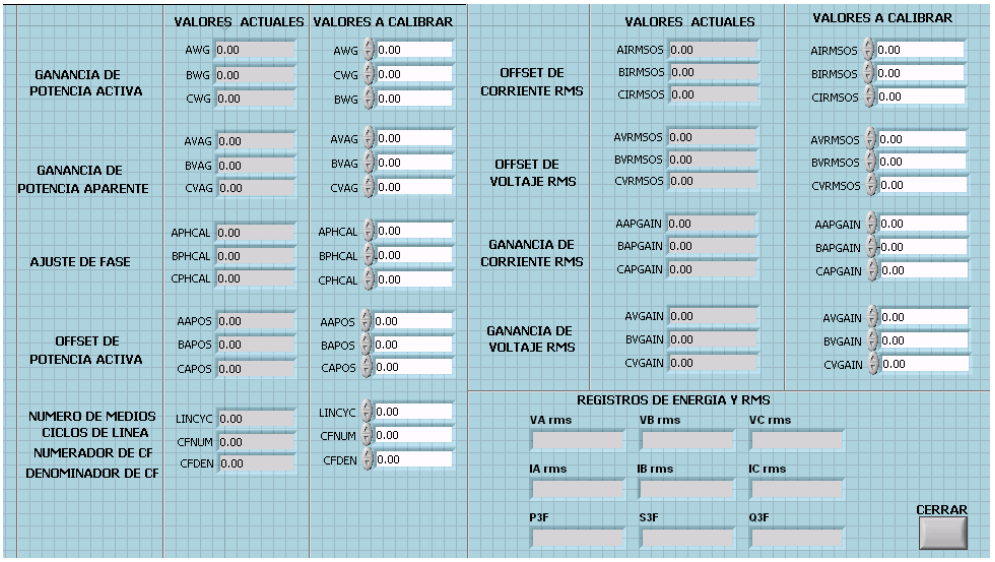

Figura A.2.4 VI de calibración sección 2

PROCEDIMIENTO DE CALIBRACION

1. Para activar este VI se presiona el botón con la flecha en la barra de ejecución de LabVIEW. Si se desea detener antes de terminar el proceso de calibración presione el botón con el círculo rojo. Ver figura A.2.2. Para calibrar el medidor es necesario que el medidor este en modo calibración.

2. Se digita la dirección IP de la tarjeta, en el espacio de Dirección IP como se muestra en la figura A.2.3. La dirección IP por defecto es la 192.168.0.70

3. Se selecciona la relación de transformación de los TI, así como la relación de transformación del los TT, en el la figura A.2.3. Esto se hace de la siguiente manera:

   600/5 significa que cuando se tenga una lectura de 600 A en el primario se tendrá una lectura de 5 A en el secundario, del transformador de corriente.

4. Se presiona el botón de cargar referencias de la figura A.2.3. Cuando se presiona el botón aparece un menú emergente, ver figura A.2.5, el cual solicita la ubicación del archivo de referencias. Al cargar las referencias, se recomienda utilizar las referencias dadas con el software de aplicación.

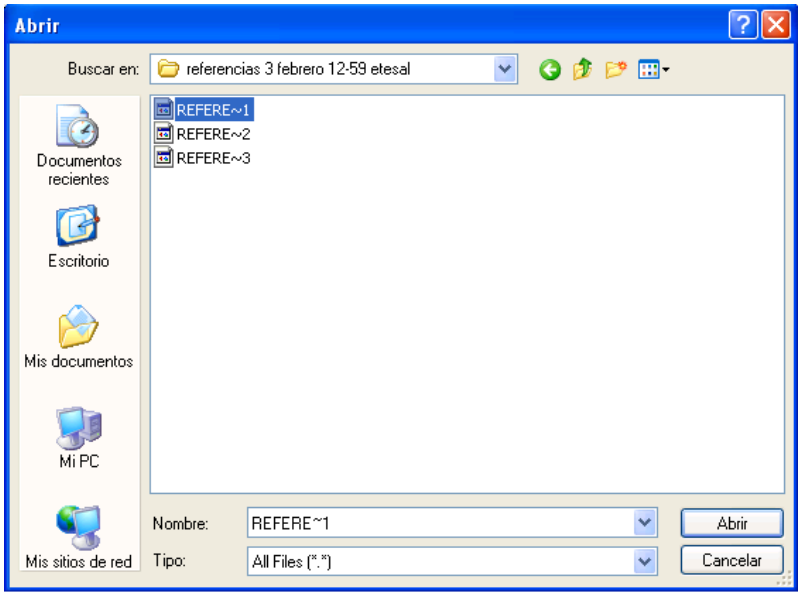

Figura A.2.5 Menú emergente para abrir un archivo

5. Se presiona el botón LEER REGISTROS DE CALIBRACIÓN correspondiente de la figura A.2.3; el medidor mostrará los valores de los registros de calibración que se encuentran en la memoria EEPROM, o los valores que se escribieron en la última calibración, así como los registros de potencia y de valores rms actuales.

6. Se observarán las salidas numéricas de la primera columna correspondiente a los últimos registros guardados de calibración.

   Los registros de calibración:
   Registros de ganancia de potencia activa
   Registros de ganancia de potencia aparente
   Registros de ganancia de fase
   Registros de offset de potencia activa
   Números de medios ciclos de línea
   Multiplicador o numerador de CF
   Divisor o denominador de CF

   Registros de offset de corriente rms
   Registros de offset de voltaje rms
   Registros de ganancia de corriente rms
   Registros de ganancia de voltaje rms

   Los registros de potencia, de voltaje y de corriente:
   Registros de potencia activa
   Registros de potencia aparente
   Registros de potencia reactiva
   Registros de voltaje rms
   Registros de corriente rms
   Registros de potencia trifásica

7. Para ajustar el pulso de salida de frecuencia, se tiene acceso a los registros del ADE7754. Los registros de ganancia y de offset son por fase. Cabe destacar que el los valores para los registros de ganancia deben estar en el rango de –2047 a 2047. Los registros de ajuste de fase entre –15 a 15, el registro de CFNUM entre 0 a 4095, el registro de CFDEN entre 1 a 4095, y el número de ciclos de línea entre 20 a 2000.

8. Se escribe en los campos de entrada de texto correspondientes a cada uno de los registros de configuración, los valores deseados para calibrar el medidor, ver figura A.2.4

9. Se presiona el botón ESCRIBIR VALORES DE CALIBRACIÓN de la figura A.2.3

10. Si se desea modificar los valores de calibración nuevamente, se presiona el botón ESCRIBIR VALORES DE CALIBRACIÓN de la figura A.2.3, pero sino desea modificarlos, se debe presionar el botón SALIR DE CALIBRACIÓN E INICIAR MEDICIÓN de la figura A.2.3, en este modo el medidor se queda con los valores calibrados escritos, y el pulso de salida de frecuencia trifásico se activa; este pulso es de una gran utilidad, debido a que con este se logra determinar el % de error en la exactitud del medidor. Cuando no se utiliza más el modo de calibración, se puede presionar el botón CERRAR ver figura A.2.3, y se cierra la aplicación, y el medidor inicia en modo medición.

VI PRINCIPAL

El VI principal es que permite comunicarnos con el medidor para realizar las siguientes funciones:

- Ajustar el reloj de tiempo real
- Hacer lecturas de valores instantáneas
- Hacer descargas de datos de la memoria del medidor
- Graficas de potencia

Estas son las operaciones básicas que puede realizar el medidor para el tratamiento de los datos.
Pero como se muestra en la figura A.2.6 tiene opciones de configuración, tanto del medidor como de la tarjeta.

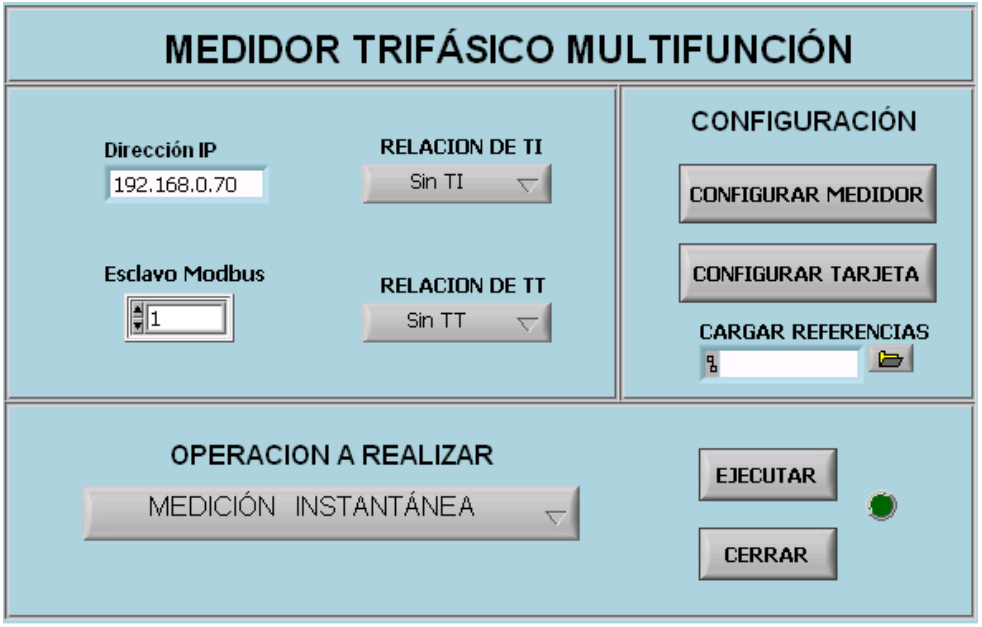

Figura A.2.6 VI principal de la interfase gráfica

Cuando se selecciona una operación aparece el VI emergente correspondiente, que al cerrarlo, retorna a esta aplicación. Este VI emergente tiene su respectiva ayuda que puede ser de mucha ayuda para el usuario.

Hay funciones que necesitan que el medidor esté en funcionamiento, como la lectura de valores instantáneos, la descarga de los valores guardados en memoria y la configuración de la fecha y hora del medidor.

PROCEDIMIENTO PARA UTILIZAR EL PROGRAMA PRINCIPAL

1. Digitar la dirección IP de la tarjeta. Ver figura A.2.6. La IP por defecto que posee es la 192.168.0.70 y aparece en la parte lateral de la tarjeta del medidor.

2. Seleccionar el número de esclavo MODBUS del medidor, el esclavo que posee este medidor es el número 1, pero puede ser modificado.

3. Se debe conocer la relación de transformación de los TI y de los TT, y para ello se tienen los botones relación de TT y TI de la figura A.2.6.

4. El medidor posee opciones de configuración y una de ella es la configuración del reloj del medidor. Siempre que se encienda el medidor se tiene que cambiar el reloj, ya que el medidor no guarda los valores que se le han configurado al reloj cada vez que se apaga.

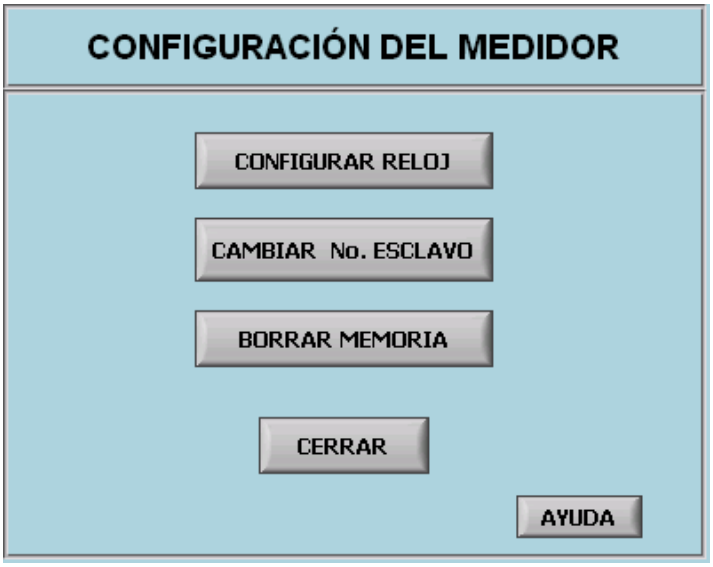

Figura A.2.7 VI de configuración del medidor

Para ello se presiona el botón CONFIGURAR RELOJ de la figura A.2.7, y aparecerá un VI emergente de LECTURA Y MODIFICACIÓN DE HORA Y FECHA DEL MEDIDOR, ver figura A.2.8.

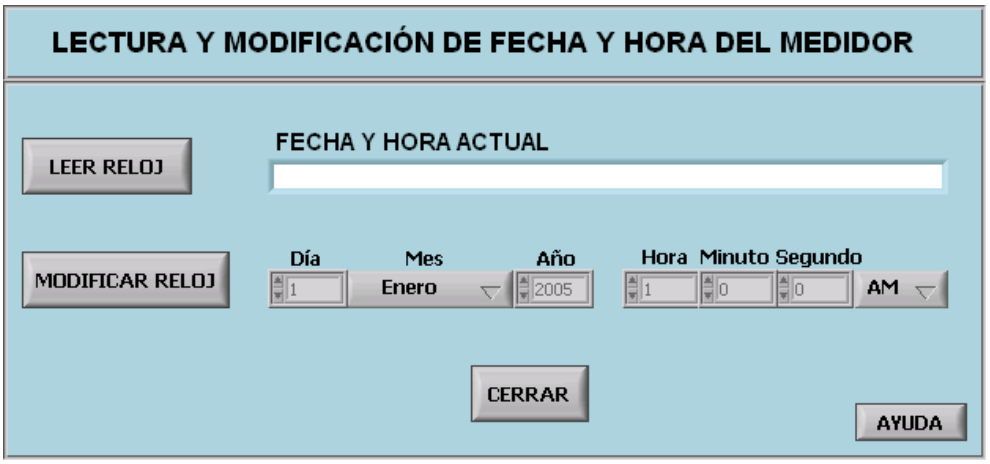

Figura A.2.8 VI de modificar fecha y hora del medidor.

Se presiona el botón LEER RELOJ figura A.2.8, y aparece la hora y la fecha actual en el campo de salida, si se desea modificar el reloj, hay que seleccionar la fecha y hora, luego de tener la fecha y hora a cambiar se debe presionar el botón MODIFICAR RELOJ figura A.2.8; en caso contrario, se presiona el botón CERRAR de la misma figura.

Si se desea modificar nuevamente el reloj se realiza el mismo procedimiento descrito. Al introducir la fecha, se hace en el siguiente formato:

Día/Mes/Año/Hora/Minuto/Segundo/AM (o PM)

5. La segunda opción de configuración es cambiar el número de esclavo MODBUS del medidor. Ver figura A.2.7. Se debe presionar el botón CAMBIAR No ESCLAVO y aparecerá el VI emergente de configuración del esclavo MODBUS.

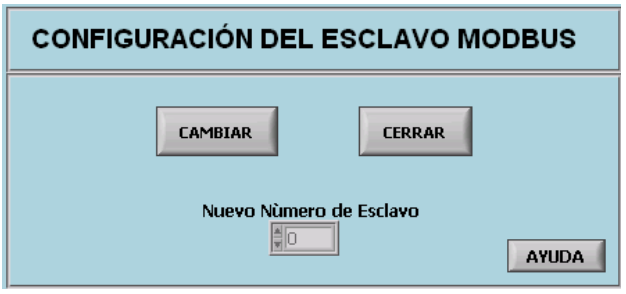

Figura A.2.9 VI Configurar Esclavo MODBUS.

6. Digitar el nuevo número de esclavo, se debe asignar un número de esclavo que no esté asignado y debe de ir desde 1 hasta 247. Presionar CAMBIAR y luego CERRAR para salir de este VI.

7. La tercera opción de configuración es la de borrar la memoria del medidor. Ver figura A.2.7. Cuando se presione el botón BORRAR MEMORIA, aparecerá el VI emergente BORRAR DATOS EN MEMORIA. Si se está completamente seguro de borrar la memoria, presionar el botón BORRAR DATOS y luego CERRAR para salir de este VI.

Figura A.2.10 VI Borrar datos en memoria.

8. Presionar CONFIGURAR TARJETA, de la figura A.2.6 para cambiar la dirección IP de la tarjeta, aparecerá el VI emergente CONFIGURACIÓN DE LA TARJETA. Digitar la nueva IP y presionar el botón modificar IP para cambiar la dirección IP.

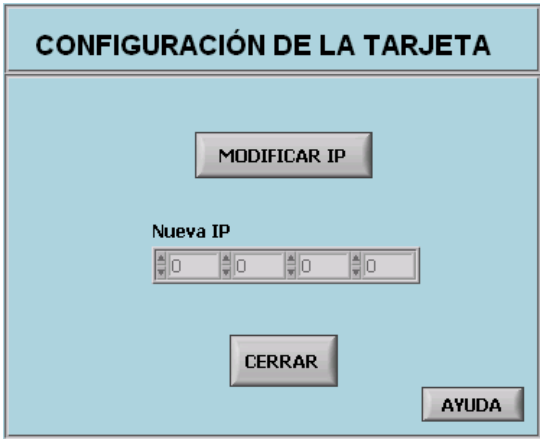

Figura A.2.11 VI Configuración de la tarjeta.

9. Volviendo al VI principal, se deben cargar las referencias de calibración presionando el botón CARGAR REFERENCIAS figura A.2.6, y aparecerá un menú emergente solicitando el archivo de

referencias como en la figura A.2.5, como se realizó en el VI de calibración.

10. Se escoge la función que se desea, presionando el botón OPERACIÓN A REALIZAR figura A.2.6, con el cual se observarán cuatro opciones al presionarlo. Ver figura A.2.6

Estas opciones son:

- Ajustar el reloj de tiempo real
- Hacer lecturas de valores instantáneas
- Hacer descargas de datos de la memoria del medidor
- Graficas de potencia

11. Se presiona el botón EJECUTAR, aparecerá un VI emergente correspondiente a la operación que se eligió. Cuando el VI emergente se cierra, inmediatamente regresa al VI principal.

12. Después de que el VI emergente regresa al VI principal, se tienen dos opciones se realiza otra operación, repitiendo todos los pasos anteriores, se presiona el botón CERRAR figura A.2.7, y se cierra la aplicación.

VI DE LECTURA ACTUAL DE POTENCIA Y DE VALORES RMS

Cuando se selecciona la operación de lectura actual de potencia y de valores rms, y se presiona el botón EJECUTAR del programa principal; aparece el VI emergente que se muestra en la figura A.2.12.

PROCEDIMIENTO DE LECTURA DE VALORES INSTANTÁNEOS

Se presiona el botón LEER figura A.2.12, luego aparecen los valores medidos por el medidor en ese momento, en los cuadros de salida:

Hora y fecha
Potencia activa trifásica
Potencia aparente trifásica
Potencia reactiva trifásica
Factor de potencia trifásica
Voltaje rms de la fase A
Voltaje rms de la fase B
Voltaje rms de la fase C
Corriente rms de la fase A
Corriente rms de la fase B
Corriente rms de la fase C

Si se desea leer otra vez, se presiona nuevamente el botón LEER figura A.2.12; de lo contrario se puede presionar el botón CERRAR.

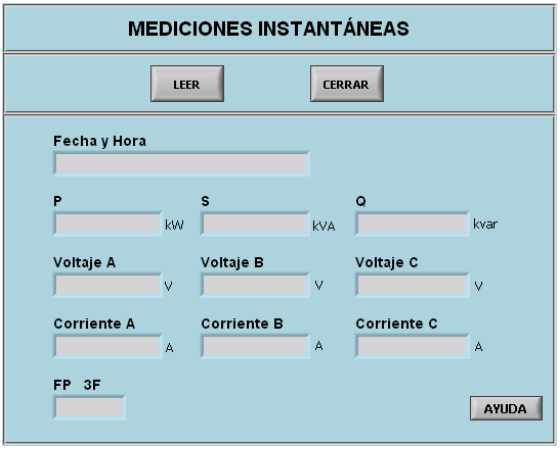

Figura A.2.12 VI de mediciones instantáneas

VI DE DESCARGA DE MEMORIA

Este VI descarga el contenido de la memoria en un archivo de texto.

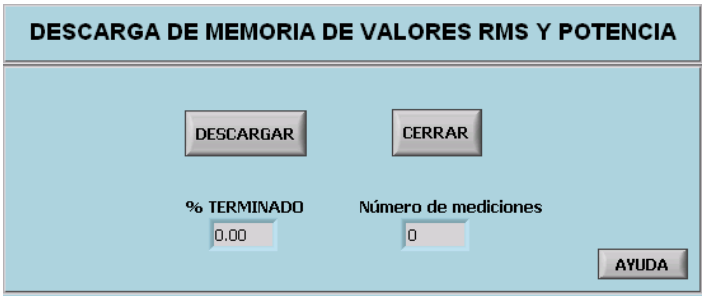

Figura A.2.10 VI descarga de valores almacenados en memoria

PROCEDIMIENTO DE DESCARGA DE MEMORIA

1. Se presiona el botón DESCARGAR figura A.2.10, y aparecen un menú emergente, este solicita donde se desea guardar el archivo con formato de fecha y aparece el nombre “form-rms-pot-día-mes-año.txt”.

   P, es la potencia activa trifásica
   S, es la potencia aparente trifásica
   Q, es la potencia reactiva trifásica
   VA, es el voltaje rms de la fase A

VB, es el voltaje rms de la fase B
VC, es el voltaje rms de la fase C
IA, es la corriente rms de la fase A
IB, es la corriente rms de la fase B
IC, es la corriente rms de la fase C

2. Se puede observar el indicador que muestra el progreso de la descarga mediante la salida % TERMINADO literal 3 figura A.1.30.

3. Se presiona el botón CERRAR literal 2 figura A.1.30.

VI DE GRÁFICAS DE POTENCIA

1. Este VI genera gráficas de potencia de los archivos descargados del medidor; posee opciones de búsqueda de archivos.

2. Seleccionar la carpeta donde se han guardado los archivos *.txt.

3. Al presionar el botón para buscar archivos los resultados obtenidos se muestran en RESULTADOS DE BUSQUEDA.

4. Seleccionar un archivo y presionar el botón GRAFICAR y se generarán las gráficas de potencia, la fecha inicial y Final de las mediciones que se encuentran en el archivo seleccionado y los valores máximos y mínimos de voltaje, corriente y potencia.

5. Seleccionar otro archivo y se graficarán los datos de otra descarga.

6. Presionar CERRAR para salir de este VI.

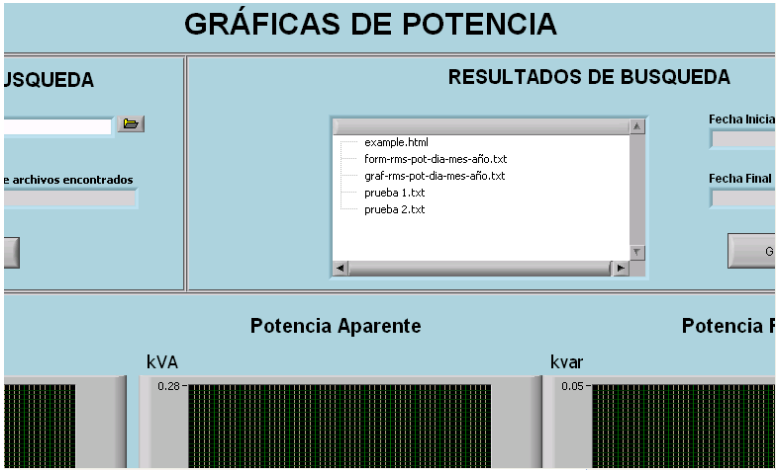

Figura A.2.11 VI Graficas de potencia

VI DE GENERACIÓN DE REPORTES HTML

Este VI crea archivos HTML a partir de archivos de texto descargados de la memoria del medidor previamente.

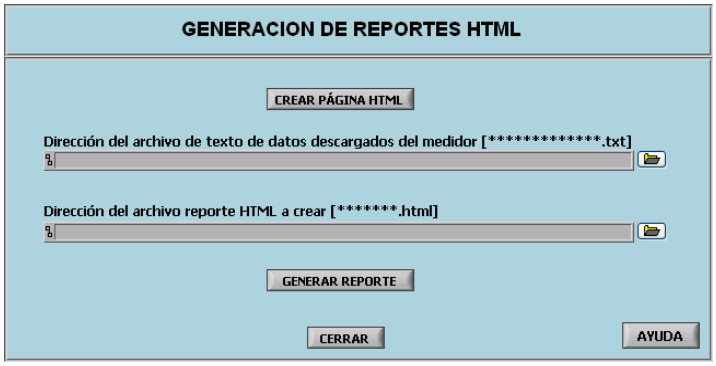

Figura A.2.12 VI generación de reporte HTML

PROCEDIMIENTO DE GENERACIÓN DE REPORTES HTML

1. Presione el botón CREAR PAGINA figura A.2.12, y aparecerá un menú emergente solicitando el nombre y la dirección del archivo HTML en blanco a crear.

2. Digitar el nombre y la ubicación e del archivo de texto con los datos de interés en el campo de texto de entrada figura A.2.12, o presionar el botón figura A.2.12 para cargar el archivo.

3. Digitar el nombre y la ubicación del archivo HTML que se creo en el paso 1 en el campo de texto de entrada figura A.2.12, o presionar el botón para cargar el archivo.

4. Presionar el botón GENERAR REPORTE figura A.2.12, luego aparecerá un menú emergente con dos botones ver figura A.2.13. Se presiona el botón Replace de la figura A.2.13.

Figura A.2.12 Menú emergente para reemplazar archivo.

5. Se presiona el botón CERRAR figura A.2.12.

MANUAL DE USUARIO DEL HARDWARE DEL MEDIDOR

ELEMENTOS DEL INSTRUMENTO

1 - MEDIDOR DE PARAMETROS ELECTRICOS

El medidor se encarga de obtener los paramentos eléctricos de voltaje, corriente y potencia del sistema a monitorear. En la figura A.2.13 se muestra la vista frontal del medidor en donde se identifica cada una de sus partes.

Figura A.2.13 Vista frontal del Medidor

IRQ: Este LED permanece constantemente encendido y se apaga durante la recepción de una interrupción por parte del ADE7754. Al finalizar el tratamiento a la interrupción el led IRQ enciende de nuevo.

RELOJ: Este led intermitente cambia cada segundo e indica la correcta operación del reloj interno del medidor.
PULSO DE FRECUENCIA: Este led sirve para propósitos de calibración realizando para ello una conversión de potencia a frecuencia.

CONECTOR RS232: Sirve para interconectar el medidor con la tarjeta.
ETHERNET-SERIAL diseñada ésta tesis. El cable debe ser serial recto.

INTERRUPTOR DE ENCENDIDO: Una vez conectada la carga y el medidor, el interruptor energiza todos los elementos del medidor para iniciar con las operaciones.

SELECTOR CAL/MEDICION: Este interruptor indica en que modo iniciara el medidor. Al iniciar el medidor verifica el estado de este interruptor y dependiendo de ello podrá responder a peticiones de calibración o de medición.
Si se inicia en modo calibración, desde el software se debe volver al modo medición. No ser puede entrar en modo medición simplemente cambiando el interruptor al estado de medición mientras el equipo esta encendido.

RESET: Sirve para reiniciar las operaciones de todo el medidor. Nuevamente el medidor revisa el estado del interruptor CAL/MED para iniciar en el modo indicado.

ENTER: Cuando se iniciar en modo medición (colocando en interruptor.

CAL/MED en la posición de medición), se debe esperar aproximadamente 3 segundos y luego mantener presionado el botón ENTER hasta que el led IRQ encienda, esto indica el inicio de las mediciones de potencia por parte del medidor.
Cuando se inicia en modo calibración se debe esperar aproximadamente 5 segundos y luego mantener presionado el botón ENTER hasta que el led IRQ encienda. Luego de esto el medidor ha entrado en modo calibración y aceptara peticiones para ser calibrado hasta recibir la instrucción por medio de software para entrar en modo medición.

En la parte trasera del medidor se encuentras las entradas para el voltaje y corriente de la carga a monitorear, en la figura A.2.14 se identifican estos conectores.

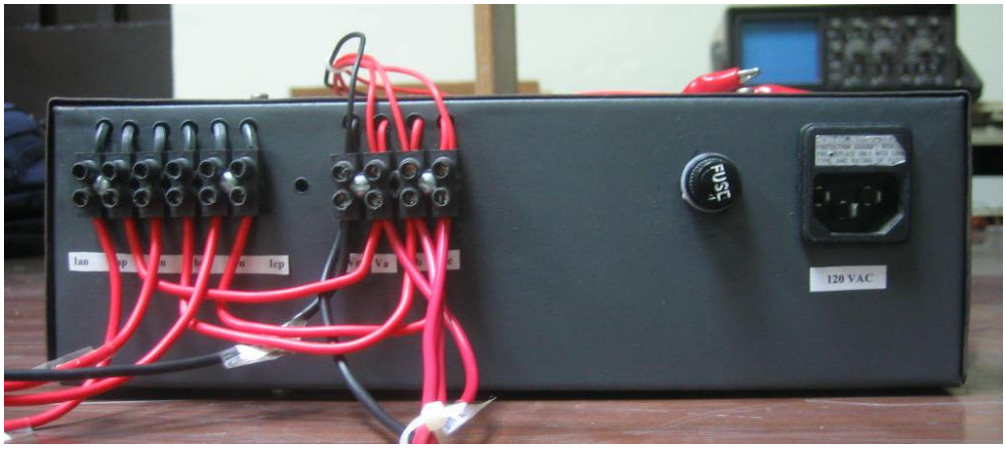

Figura A.2.14 Vista trasera del Medidor

ENTRADAS DE CORRIENTE: Son la entradas para la medición de la corriente de la carga y las conexiones soportadas se pueden conocer en el documento de tesis “Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754”.
ENTRADAS DEL VOLTAJE: Son las entradas para la medición del voltaje de la carga y las conexiones soportadas se pueden conocer en el documento “Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754”

FUSIBLE: Es un fusible de protección por sobre-corrientes en la alimentación del medidor.

ENTRADA DE PODER: Es la alimentación para el medidor.

Además el medidor incluye un cable de potencia para su alimentación a 110V y un cable RS232 para comunicarse con la tarjeta ETHERNET-SERIAL.

2 - PROTECCIONES PARA EL EQUIPO DE MEDICIÓN

Este instrumento protege al medidor de sobre-voltajes o sobre-corrientes en el lado de la carga, esta diseñado para operar hasta 220 V línea a línea y con una corriente máxima de 5A. En la figura A.2.15 y A.2.16 se muestra el instrumento en cuestión.

Figura A.2.15 Vista frontal de las protecciones

Figura A.2.16 Vista trasera de las protecciones

En la figura A.2.17 se muestra el diagrama eléctrico de las protecciones.

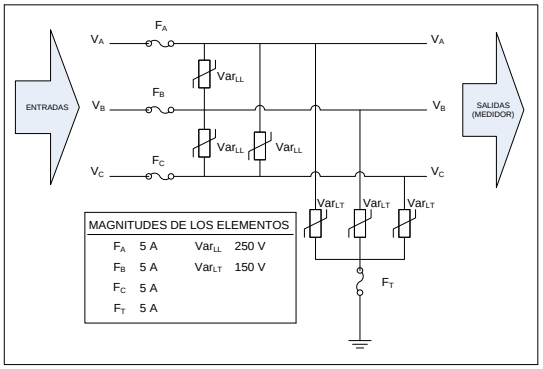

Figura A.2.17 Diagrama eléctrico de la protección

3 - MODULO ETHERNET-SERIAL

Este dispositivo se encarga de cambiar del medio SERIAL a ETHERNET, y utiliza el protocolo MODBUS (capa de aplicación) para la comunicación en ambos medios.
En el medio Ethernet opera como un SERVIDOR MODBUS – TCP y únicamente responde a las peticiones hechas por un CLIENTE MODBUS – TCP que debe estar dentro de la misma red local (LAN).
En el medio Serial el modulo opera como un MAESTRO MODBUS SERIAL quien interroga a los dispositivos ESCLAVOS (en este caso el Medidor) para la obtención de datos o configuración de los mismos. En el medio serial debe existir un solo MAESTRO y hasta 255 esclavos que únicamente transmitirán cuando el MAESTRO se los indique.
La vista frontal del modulo se puede observar en la figura A.2.18, indicando sus partes.

Figura A.2.18 Vista frontal del modulo ETHERNET-SERIAL

CONECTOR ETHERNET: Permita la conexión a la red LAN a través de un conector RJ45.

En la figura A.2.19 se muestra la parte trasera del modulo, indicando los elementos que contiene.

Figura A.2.19 Vista trasera del modulo ETHERNET-SERIAL

CONECTOR RS232: Permite la conexión de un cable serial RS-232 recto para comunicar el medidor con la tarjeta ETHERNET-SERIAL.

VOLTAJE DE ENTRADA: Es la alimentación de la tarjeta ETHERNE-SERIAL y esta en el rango de 12-30 VDC.

En la figura A.2.20 se muestra la cara lateral del modulo.

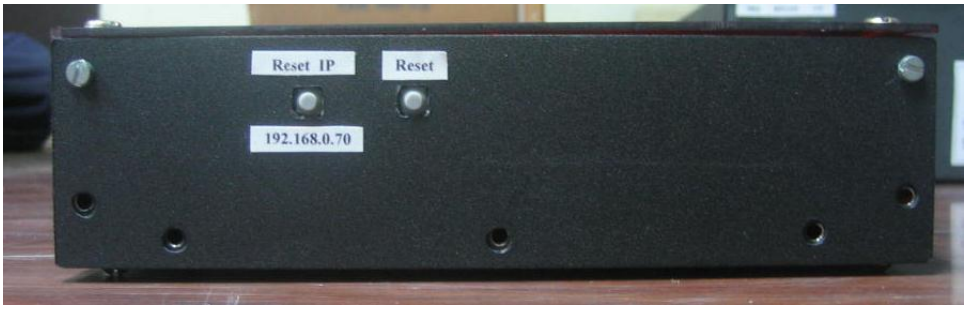

Figura A.2.20 Vista lateral del modulo ETHERNET-SERIAL

RESET: Sirve para re-iniciar la tarjeta ETHERNE-SERIAL completa.

RESET IP: Esta opción hace que la tarjeta vuelva a la IP 192.168.0.70 y no la perderá al ser apagada.

PASOS PARA REALIZAR MEDICIÓN DE PARAMETROS ELECTRICOS

A continuación se describen los pasos a seguir para conectar el medidor junto a sus componentes de forma adecuada e inicializar el monitoreo de los parámetros eléctricos de un sistema trifásico.

1- Se configura el medidor según el tipo de conexión al cual se conectará. Esto se logra moviendo los jumpers JP32 y JP33. En la tabla A.2.1 y la A.2.2, se muestran los tipos de conexiones soportadas por el medidor.

Numero de hilos	Tipo de conexión	Tipo de base	Número de elementos	Fórmula
3 hilos	Delta	5S/13S	3 elementos	P = VA * IA + VB * IB + VC * IC
4 hilos	Estrella	6S/14S	2 ½ elementos	P = VA*(IA-IB) + VC*(IC-IB)
4 hilos	Delta	8S/15S	2 ½ elementos	P = VA*(IA-IB) + VC * IC
4 hilos	Estrella	9S/16S	3 elementos	P = VA * IA + VB * IB + VC * IC

Tabla A.2.1 Conexiones soportadas por el medidor para potencia activa

Numero de hilos	Tipo de conexión	Tipo de base	Número de elementos	Fórmula
3 hilos	Delta	5S/13S	3 elementos	S = VArms * IArms + VBrms * IBrms + VCrms * Icrms
4 hilos	Estrella	6S/14S	2 ½ elementos	S = VArms * IArms + (Varms+VCrms)/2 * IBrms + VCrms * Icrms
4 hilos	Delta	8S/15S	2 ½ elementos	S = VArms * IArms + VBrms * IBrms + VCrms * Icrms
4 hilos	Estrella	9S/16S	3 elementos	S = VArms * IArms + VBrms * IBrms + VCrms * Icrms

Tabla A.2.2 Conexiones soportadas por el medidor para potencia aparente

2- Se verifica que el switch de CALIBRACION/MEDICION (ver vista frontal del medidor) este ubicado en MEDICIÓN, de lo contrario esperara los comandos para calibración o salir de calibración.
3- Se alimenta con el cable de potencia al medidor de energía, éste debe ser conectado a 120 V AC.
4- Se realizar las conexiones desde las líneas trifásicas hacia las protecciones y posteriormente al medidor según se indica en las siguientes figuras.

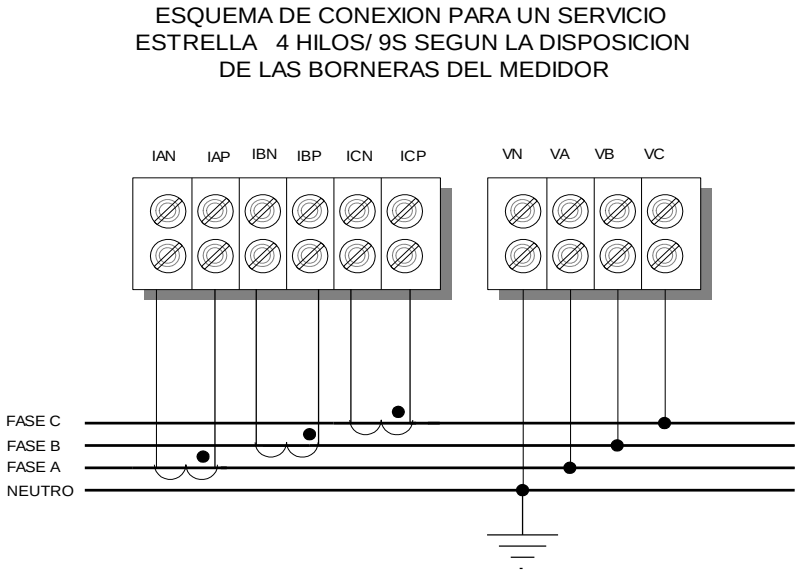

Figura A.2.21 Conexión para un sistema

5- Se realizan la conexión desde las protecciones hasta el medidor según el tipo de conexión al cual se conectara. Para mayor información refiérase a la tesis “Diseño e implementación de un medidor trifásico multifunción utilizando el IC ADE7754”
6- Luego de finalizar las conexiones del lado de la carga ser procede a encender el medidor por medio del interruptor de encendido.
7- Ahora se espera aproximadamente 3 segundos y ser mantiene presionado el botón ENTER hasta que el led IRQ encienda.
8- Ahora el medidor ha iniciado con la medición de potencia y acepta peticiones desde el medio serial. Para ello se conecta el cable serial RS232 recto entre el medidor y la tarjeta ETHERNE-SERIAL.
9- Se conecta la tarjeta ETHERNET-SERIAL a la red LAN.
10- Se energiza la tarjeta ETHERNET-SERIAL.
11- Ahora el sistema esta listo para ser interrogado desde el software en la PC usando la red LAN.

A.3 Lista de Componentes y Precios

Lista de precios de los componentes de la tarjeta.				
Parte	**Descripción**	**P. Unitario**	**cantidad**	**totales ($)**
C1, C2 C4, C5	Capacitor electrolítico de 1uF/160v	0.30	4	1.20
C7	Capacitor electrolítico de 0.1uF/50v	0.30	1	0.30
C8	Capacitor electrolítico de 10uF/50v	0.30	1	0.30
C9, C10	Capacitor de pastilla de 33 pF	0.12	2	0.24
R1	Resistencia de precisión 10K ohm/1/4 W 0.1%	0.12	1	0.12
R2	Resistencia de 2.2K ohm 2%	0.12	1	0.12
R3. R5	Resistencia de 650 ohm	0.12	2	0.24
R7, R9	Resistencia de 10.5K ohm 5%	0.12	2	0.24
R10, R12	Resistencia de 1.2K ohm 5%	0.12	2	0.24
IC1	Microcontrolador PIC16F877/P	5.00	1	5.00
IC2	Convertidor TTL-RS232 MAX232	3.00	1	3.00
IC3	Memoria 24LC1025 (Uso futuro)	——	1	——
IC4, IC5	DIP-6 SCMITT TRIGG optoacoplador H11L1M	6.00	2	12.00
IC6	Regulador de voltaje de 5V 7805	0.75	1	0.75
IC7	Packet Whacker	35.00	1	35.00
D1	Diodo zener 1N4082	0.12	1	0.12
F1	Fusible de 1 A	0.15	1	0.15
Q3	Cristal de 20 MHz	0.75	1	0.15
JP3 – JP6	Jumper de una posición	0.10	4	0.40
	Tarjeta impresa	8.00	1	8.00
		P. Total tarjeta		67.57

Tabla A.3.1 Lista de componentes y precios de la tarjeta

A.4 Circuito Impreso de la Tarjeta

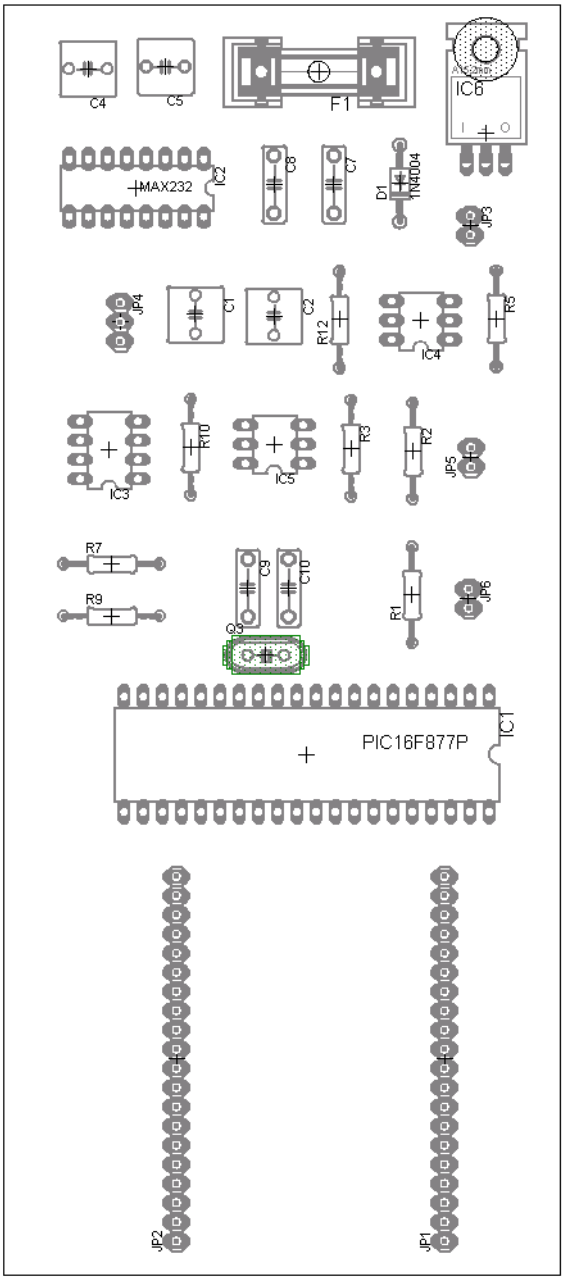

Figura A.4.1 Componentes utilizados en la tarjeta impresa

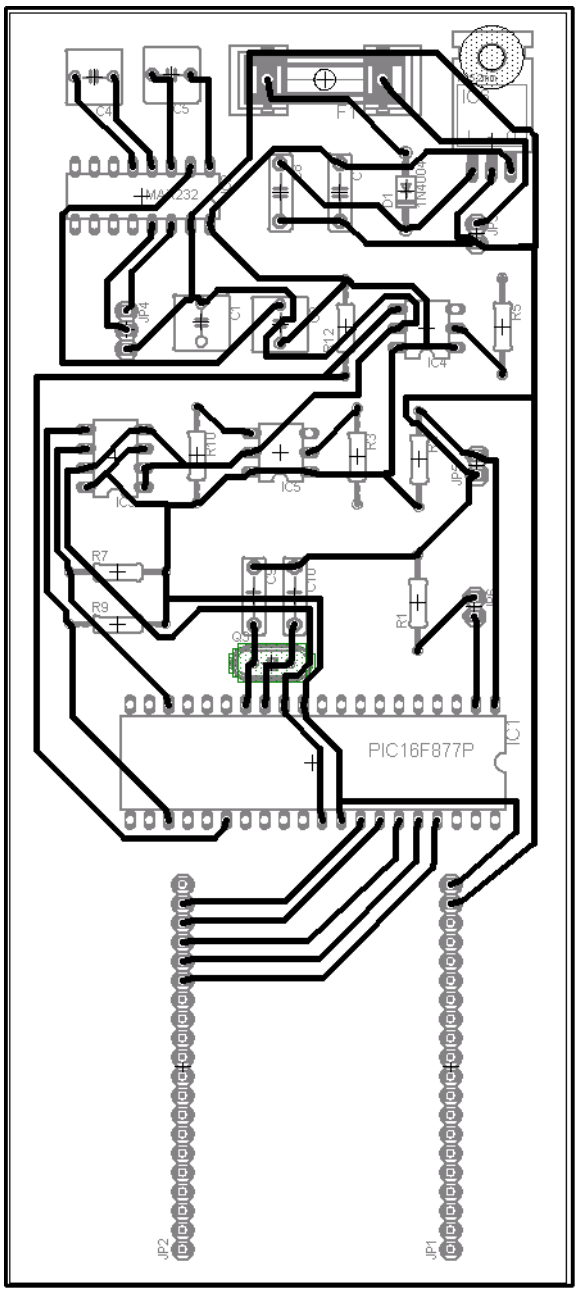

Figura A.2.2 Tarjeta impresa parte inferior

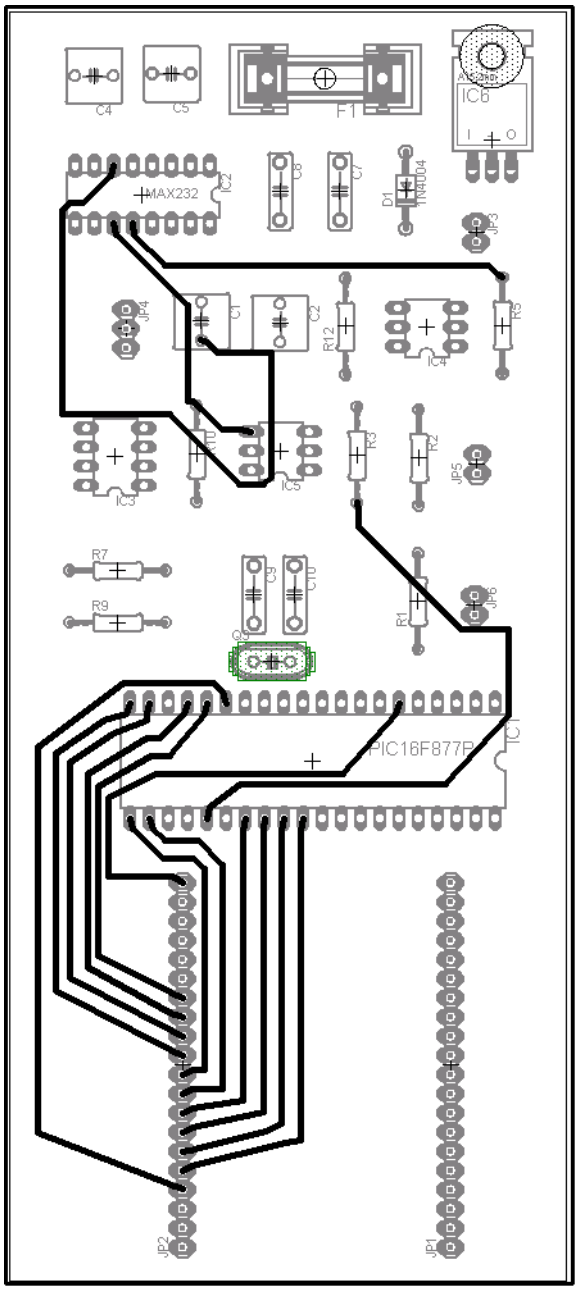

Figura A.4.3 Tarjeta impresa parte superior

A.5 Flujogramas del Programa del Microcontrolador

Flujogramas de la CAPA 1, IEEE Std 802.3

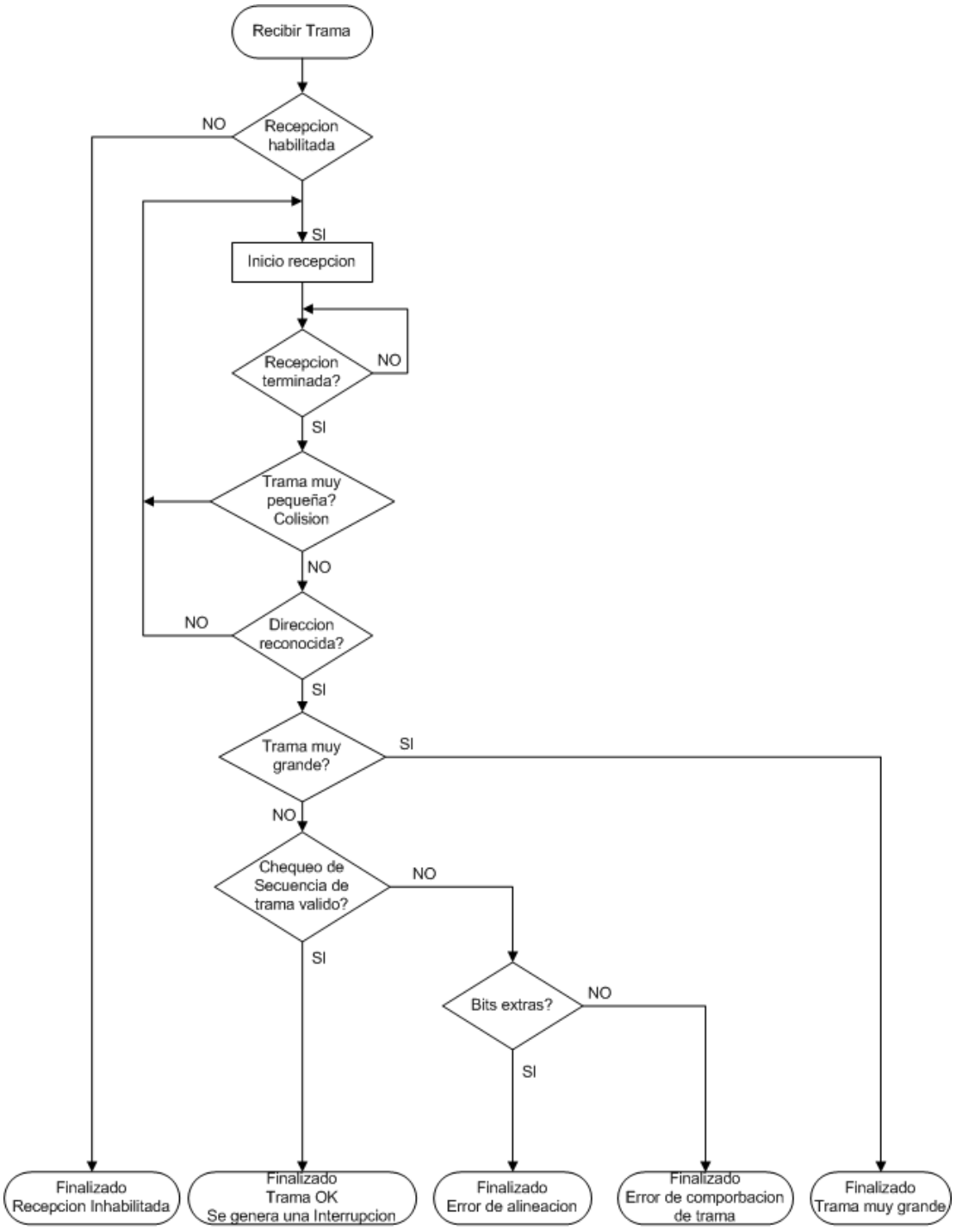

Figura A.5.1 Flujograma de recepción y verificación de trama por el RTL8019AS para generar una interrupción

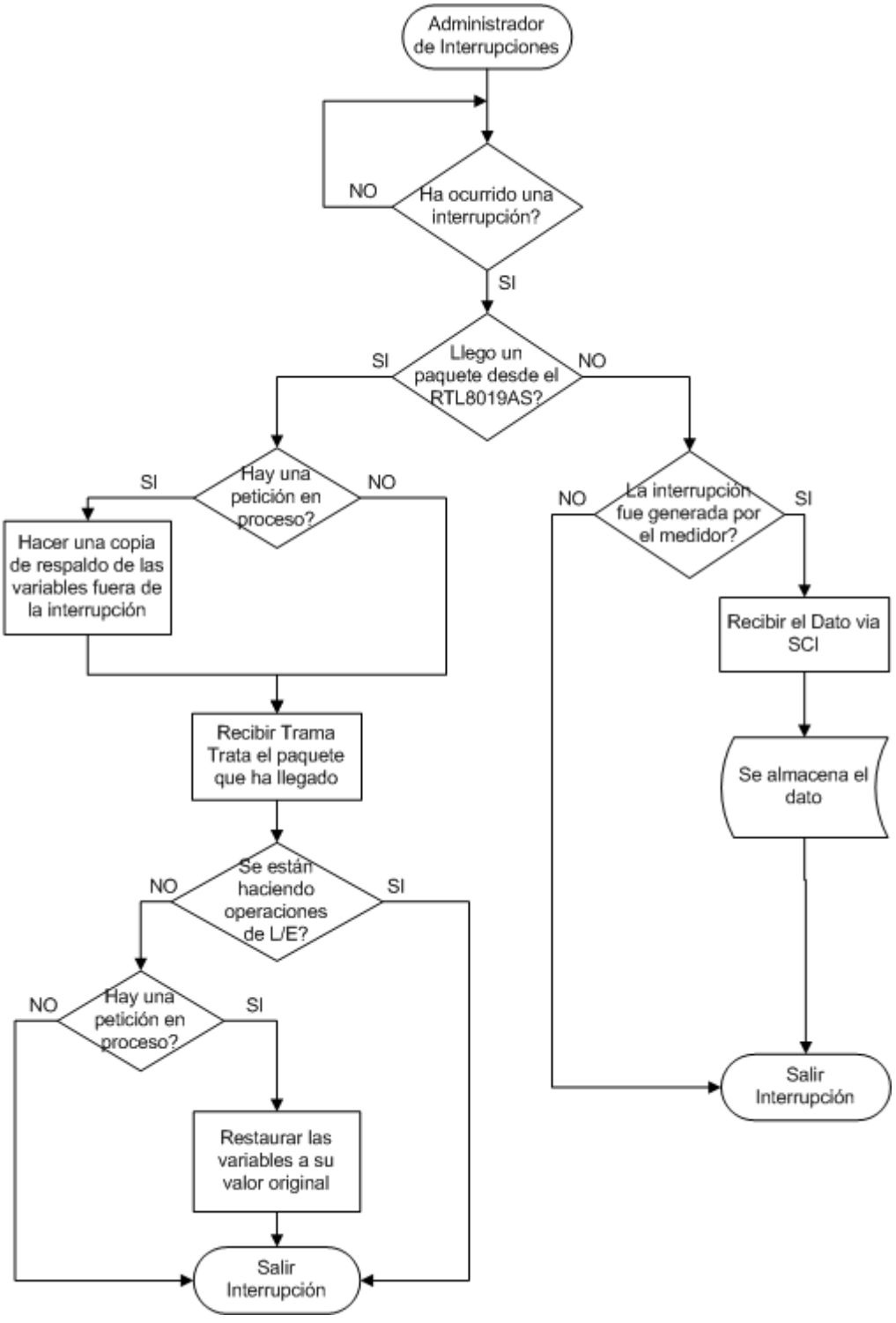

Figura A.5.2 Flujograma del Administrador de Interrupciones

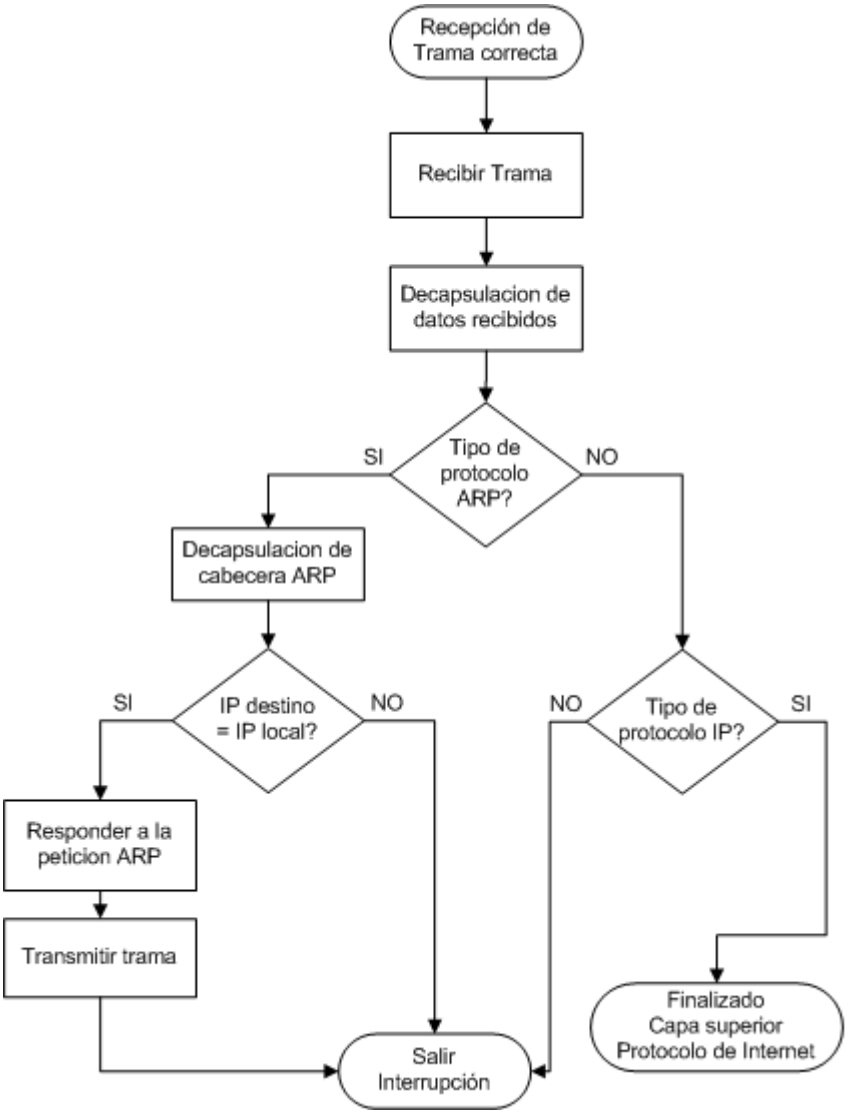

Figura A.5.3 Flujograma de recepción de trama de acuerdo al IEEE802.3

Flujogramas de la CAPA 2, Protocolo de Internet

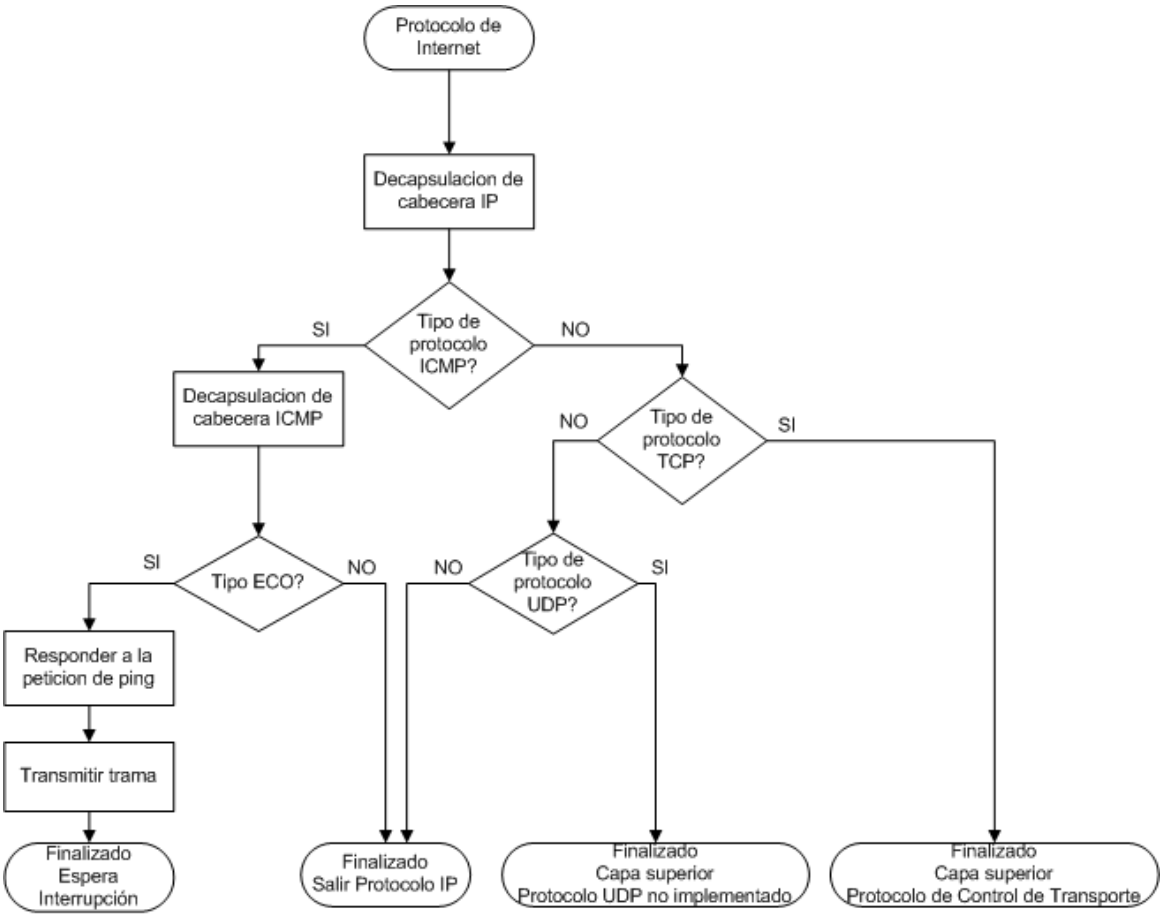

Figura A.5.4 Flujograma del Protocolo de Internet (IPv4)

Flujogramas de la CAPA 3, Protocolo de Control de Transporte

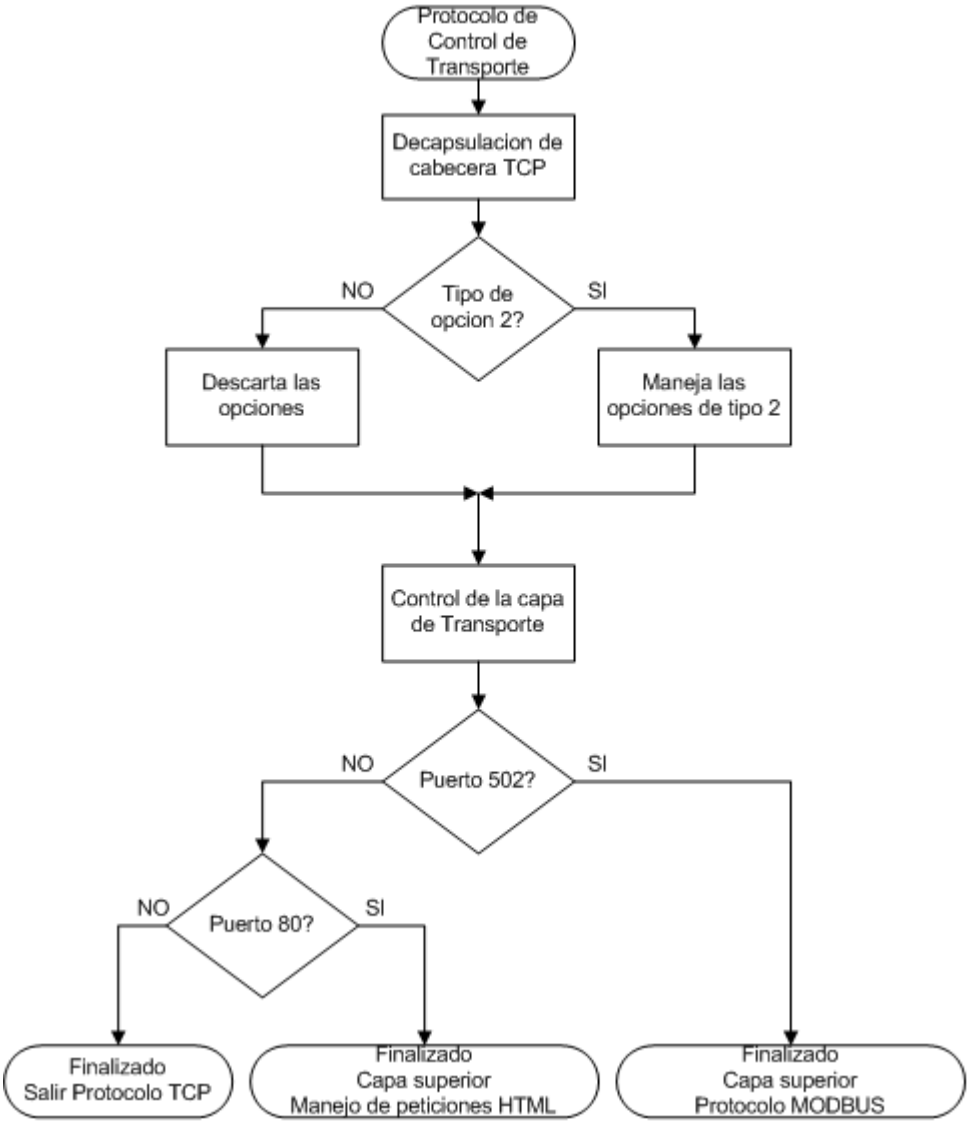

Figura A.5.5 Flujograma del Protocolo de Control de Transporte

Flujogramas de la CAPA 4, Protocolo MODBUS/TCP

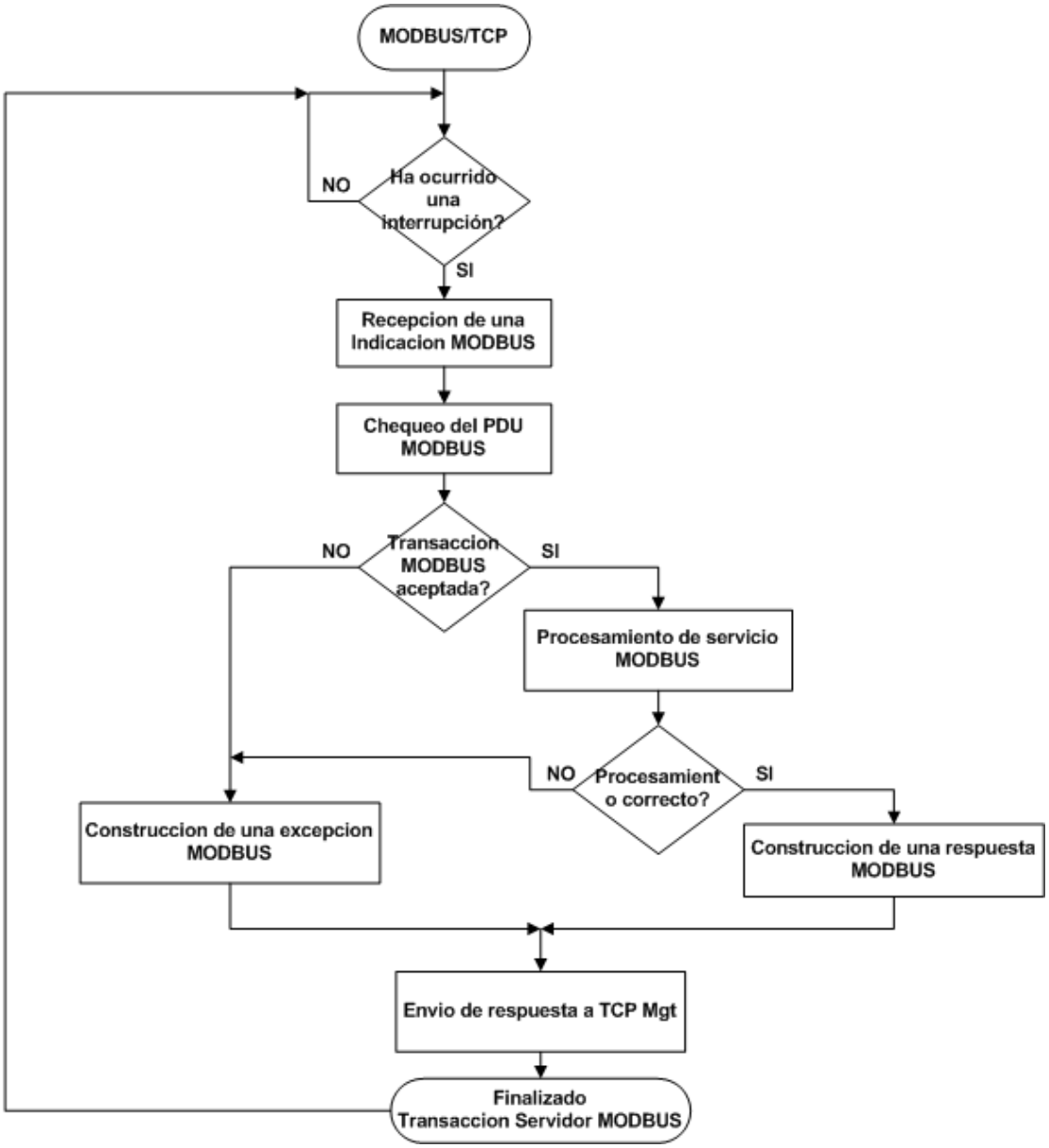

Figura A.5.6 Flujograma del Protocolo de Aplicación MODBUS/TCP

A.6 Código del Programa

El código del programa del Medidor Trifásico Multifunción se encuentra en el disco compacto de este Trabajo de graduación, la razón por la cual no se agrego al documento es porque es muy grande. Este código ha sido modificado, y es para usar esta tarjeta tiene que utilizarse es código que se proporciona en el disco de este Trabajo de Graduación.

Los nombres de los códigos son:

- ✓ **Código PIC maestro MEDIDOR.asm**
- ✓ **Código PIC RELOJ MEDIDOR.asm**
- ✓ **Código PIC MEMORIA MEDIDOR.asm**

Solamente se agrego el código de la Tarjeta Serial-Ethernet MODBUS.

Código Tarjeta Serial-Ethernet.asm

```
;*******************************************************
;* INICIALIZACIÓN Y LECTURA DE PAQUETES                *
;*******************************************************

        include     "P16F877A.INC"
        include "C:\Prog_Pic2\macros.mac"
        include "C:\Prog_Pic2\constantes_eeprom.inc"
        __config H'3f72'
        __idlocs H'7777'

;Asigno los puertos B, C y D a los buses de dirección, control
;y datos respectivamente
INT_BUS         equ     0x05        ;Puerto A [RA5-RA0]
ADDR_BUS  equ   0x06        ;Puerto B [RB7-RB0]
CTRL_BUS  equ   0x07        ;Puerto C [RC7-RC0]
DATA_BUS  equ   0x08        ;Puerto D [RD7-RD0]


;Establesco las conexiones de los pines de reset, lectura y
;escritura del RTL8019AS
RESET           equ     0x00        ;Bit 0 del puerto C
IORB            equ     0x01        ;Bit 1 del puerto C
IOWB            equ     0x02        ;Bit 2 del puerto C
INT             equ     0x04        ;Bit 4 del puerto A


;Definición de los registros del NIC RTL8019AS
;Registros de la PAGINA 0 (Registros de lectura)
CR              equ     0x00    ; Registro de Comando
CLDA0           equ     0x01    ; Registro local del DMA actual 0
CLDA1           equ     0x02    ; Registro local del DMA actual 0
BNDRY           equ     0x03    ; Registro de borde
TSR             equ     0x04    ; Registro de estados de transmisión
NCR             equ     0x05    ; Registro de numero de colisiones
FIFO            equ     0x06    ; Registro primero en entrar, primero en salir
ISR             equ     0x07    ; Registro de estados de interrupción
CRDA0           equ     0x08    ; Registro de la dirección del canal remoto DMA actual
CRDA1           equ     0x09    ; Registro de la dirección del canal remoto DMA actual
RES1            equ     0x0A    ; Reservado
RES2            equ     0x0B    ; Reservado
RSR             equ     0x0C    ; Registro de estados de recepción
CNTR0           equ     0x0D    ; Registro del contador de error de alineación de trama
CNTR1           equ     0x0E    ; Registro del contador de error de CRC
CNTR2           equ     0x0F    ; Registro del contador de paquetes perdidos
```

```
;Registros de la PAGINA 0 (Registros de escritura)
PSTART      equ         0x01    ; Registro de comienzo de  pagina
PSTOP                   equ         0x02    ; Registro de final de pagina
TPSR                    equ         0x04    ; Dirección de inicio de pagina a transmitir
TBCR0                   equ         0x05    ; Registro del contador de bytes transmitidos 0
TBCR1                   equ         0x06    ; Registro del contador de bytes transmitidos 0
RSAR0                   equ         0x08    ; Registro de de direccion de inicio remito 0
RSAR1                   equ         0x09    ; Registro de de direccion de inicio remito 0
RBCR0                   equ         0x0A    ; Registro del contador de bytes remoto 0
RBCR1                   equ         0x0B    ; Registro del contador de bytes remoto 1
RCR                     equ         0x0C    ; Registro de configuracion de recepción
TCR                     equ         0x0D    ; Registro de configuracion de transmisión
DCR                     equ         0x0E    ; Registro de configuración de datos
IMR                     equ         0x0F    ; Registro de mascaras de interrupción


;Registros de la PAGINA 1
PAR0                    equ         0x01    ; Registro de dirección fisica  0
PAR1                    equ         0x02    ; Registro de dirección fisica  1
PAR2                    equ         0x03    ; Registro de dirección fisica  2
PAR3                    equ         0x04    ; Registro de dirección fisica  3
PAR4                    equ         0x05    ; Registro de dirección fisica  4
PAR5                    equ         0x06    ; Registro de dirección fisica  5
CURR                    equ         0x07    ; Registro de pagina actual
MAR0                    equ         0x08    ; Registro de dirección multicast 0
MAR1                    equ         0x09    ; Registro de dirección multicast 1
MAR2                    equ         0x0A    ; Registro de dirección multicast 2
MAR3                    equ         0x0B    ; Registro de dirección multicast 3
MAR4                    equ         0x0C    ; Registro de dirección multicast 4
MAR5                    equ         0x0D    ; Registro de dirección multicast 5
MAR6                    equ         0x0E    ; Registro de dirección multicast 6
MAR7                    equ         0x0F    ; Registro de dirección multicast 7


;Registro de datos canal DMA y reset del RTL8019AS
NIC_DATA    equ         0x10        ; Puerto remoto DMA
NIC_RESET equ           0x18        ; Puerto de reinicio


;Definición del inicio de los buffers de recepción de paquetes y transmisión de los mismos
RCV_BUF_START equ   0x40  ; Espacio para la recepción de 3 paquetes completos
XMT_BUF_START equ   0x54  ; Espacio libre para la transmisión de 2 paquetes completos


;Definición de los Protocolos Implementados en la tarjeta
;Protocolo ARP              (Address Resolution Protocol) -> 0x0806
ARP0        equ         0x06
ARP1        equ         0x08

;Protocolo IP (Internet Protocol) -> 0x0800
IP0                     equ         0x00
IP1                     equ         0x08

;Protocolo ICMP             (Internet Control Message Protocol) -> 0x01
ICMP        equ         0x01

;Protocolo TCP              (Transport Control Protocol) -> 0x06
TCP                     equ         0x06

;Protocolo UDP (User Datagram Protocol) -> 0x17
;(NO IMPLEMENTADO)
UDP                     equ 0x17


;Para configurar la lectura o escritura remota del canal DMA
READ                    equ 0x00
WRITE                   equ         0x01

;------------ Constantes de la capa 2 - INTERNET ------------------
HOST_IP0    equ         0XC0 ;192
HOST_IP1    equ         0XA8 ;168
HOST_IP2    equ         0X02 ;2
HOST_IP3    equ         0X45 ;69

;------------ Constantes de la capa 3 - TRANSPORTE ------------------
;Banderas en la cabecera del protocolo TCP
TCP_FIN     equ         0x01
TCP_SYN     equ         0x02
TCP_RST     equ         0x04
TCP_PSH     equ         0x08
TCP_ACK     equ         0x10
ECHO        equ         0x08
REPLY       equ         0x00

;Estados del protocolo TCP
LISTEN                  equ 0x00
SYN_RCVD    equ         0x01
ESTAB                   equ         0x02
FIN                                 equ         0x03

;------------ Constantes para manejar los TIMERS --------------------
;Se establece el numero de rebasamietos del timer1
```

```
;con el proposito de esperara la  respuetas del esclavo MODBUS (medidor)
MAX_OVERFLOW_TIMER1_1   equ     0xFF
MAX_OVERFLOW_TIMER1_2   equ     0x03


;Definen los tiempo para el reloj de 3.5 caracteres serial
TIMER3.5_H equ     0XA4
TIMER3.5_L equ     0XF7


;Definen los tiempo para el reloj de 1.5 caracteres serial
TIMER1.5_H equ     0XCD
TIMER1.5_L equ     0X23

;------------ Constantes para la capa 4 - APLICACIÓN ----------------
;Excepciones MODBUS para el manejo de errores
EXCEPTION_01    equ     0x01    ;Falla por codigo de función ilegal
EXCEPTION_02    equ     0x02    ;Falla por dirección de inicio ilegal
EXCEPTION_03    equ     0x03    ;Falla por valor de datos ilegal
EXCEPTION_04    equ     0x04    ;Falla en el dispositivo esclavo


;Se definen constantes para los tipos de puertos aceptados
PORT_MODBUS             equ 0x00    ;Modbus ocupa el puerto 502
PORT_HTML               equ     0x01        ;Peticiones HTML ocupan el puerto 80
PORT_CONFIG             equ     0x02        ;Simboliza la utilización de un puerto no asignado
                                                        ;utilizado para configurar la tarjeta.


;---------- Reservación de memoria para variables a utilizar --------

;VARIABLES EN BANCO CERO (bank 0)
CBLOCK 0x20

        ;VARIABLES DE USO GENERAL
        ;Variables para generar retardo de 1ms
        Mili_Second                     ;Almacena los milisegundos del retardo
        Delay_Int                       ;Contardor para retardo de 1 ms
        Delay_Ext                       ;Counter para retardo de 1 ms
        ;Contador de bytes encapsulados en la trama a enviar
        FrameCount                      ;Numero de bytes encapsulados en la trama a enviar
        numero1_0                       ;Para operaciones de suma y resta de 2 bytes
        numero1_1
        numero2_0
        numero2_1
        resultado0                      ;Almacena el resultado de la suma o resta de 2 bytes
        resultado1
        Header_Eth_IP_TCP       ;Cantidad de bytes en cabeceras Ethernet, IP y TCP( incluye MODBUS)
        Counter                         ;Contador
        CarryOut                        ;Para conocer si existio desbordamiento en la suma o resta

        ;VARIABLES PARA LA LECTURA/ESCIRTURA DE LOS PAQUETES EN EL RTL (CAPA 1)
        ;Variables para lectura y escritura desde el canal remoto DMA
        Register_Address        ;Direccion del Register_Address que se accedera
        Register_Value          ;Register_Value que se pondra en el Register_Address
        ;Variable para configurar la lectura remota por el canal DMA
        Read_Write              ;Selecciona Lectura o WRITE para el canal DMA
        Addr0                           ;Dirección de Register_Address DMA para lectura/WRITE remota LSB
        Addr1                           ;Dirección de Register_Address DMA para lectura/WRITE remota MSB
        ;Variables en la cabecera del paquete almacenado en el buffer de anillo del RTL
        Next_Ptr_Pck            ; Segundo byte de la cabecera del paquete en el anillo
                                                ; Apunta a al inicio del siguiente paquete recivido
        Len_pack0               ;Longitud del paquete a ser enviado al medio fisico LSB
        Len_pack1               ;Longitud del paquete a ser enviado al medio fisico MSB
        Out_Word0               ; Primer byte de la palabra a cargar en el buffer de anillo
        Out_Word1               ; Segundo byte de la palabra a cargar en el buffer de anillo
        ;Apuntadores al la memoria de anillo
        In                                      ; Puntero de entrada actual CURR
        Out                                     ; Puntero de salida actual BNDRY


        ;VARIALBES UTILIZADAS EN PROTOCOLO ETHERNET (CAPA 1)
        Host_Addr0              ;Dirección MAC0 Local
        Host_Addr1              ;Dirección MAC1 Local
        Host_Addr2              ;Dirección MAC2 Local
        Host_Addr3              ;Dirección MAC3 Local
        Host_Addr4              ;Dirección MAC4 Local
        Host_Addr5              ;Dirección MAC5 Local
        sourceParam0            ;Dirección MAC0 del emisor
        sourceParam1            ;Dirección MAC1 del emisor
        sourceParam2            ;Dirección MAC2 del emisor
        sourceParam3            ;Dirección MAC3 del emisor
        sourceParam4            ;Dirección MAC4 del emisor
        sourceParam5            ;Dirección MAC5 del emisor
        lengthOrTypeParam0      ; La longitud o tipo de paquete Ethernet (LSB)
        lengthOrTypeParam1      ; La longitud o tipo de paquete Ethernet (MSB)


        ;VARIALBES UTILIZADAS EN PROTOCOLO IP (CAPA 2)
        Host_IP0                ;Dirección IP0 de este Host
        Host_IP1                ;Dirección IP1 de este Host
        Host_IP2                ;Dirección IP2 de este Host
```

```
        Host_IP3                ;Dirección IP3 de este Host
        Source_IP0              ; Dirección IP0 del paquete fuente
        Source_IP1              ; Dirección IP1 del paquete fuente
        Source_IP2              ; Dirección IP2 del paquete fuente
        Source_IP3              ; Dirección IP3 del paquete fuente
        Dest_IP0                ; Dirección IP0 destino
        Dest_IP1                ; Dirección IP1 destino
        Dest_IP2                ; Dirección IP2 destino
        Dest_IP3                ; Dirección IP3 destino
        Opt_Len                         ;En cabecera IP - Longitud de las opciones
        hdr_len                         ;En cabecera IP - Longitud de la cabecera
        IP_Length0              ;En cabecera IP - Longitud Total del paquete IP (LSB)
        IP_Length1              ;En cabecera IP - Longitud Total del paquete IP (MSB)
        Identification0         ;En cabecera IP - Valor de identificación asignado por el remitente
                                        ;como ayuda en el emsamblaje de fragmentos de un datagrama LSB
        Identification1         ;En cabecera IP - Valor de identificación asignado por el remitente
                                        ;como ayuda en el emsamblaje de fragmentos de un datagrama MSB
        Fragment0               ;En cabecera IP - Almacena las banderas del paquete IP LSB
        Fragment1               ;En cabecera IP - Almacena las banderas del paquete IP MSB
        IP_Protocol             ;En cabecera IP - Protocolo en cabecera IP
        Chksum0                         ;En cabecera IP - Suma de Control de Cabecera
        Chksum1                         ;En cabecera IP - Suma de Control de cabecera
        IP_send_protocol        ;En cabecera IP - Define el protocolo del siguiente nivel
        IP_packet_length0       ;En cabecera IP - Para el calculo de la longitud del paquete IP
        IP_packet_length1
        prev_Chksum0            ;En cabecera IP - De uso general para almacenar un valor anterior
        prev_Chksum1            ;del Checksum y compararlo con otro


        ;VARIALBES UTILIZADAS EN PROTOCOLO ICMP (CAPA 2)
        ICMP_Type               ;En cabecera ICMP - tipo de ICMP recivido
        ICMP_Code               ;En cabecera ICMP - Codigo
        ICMP_Chksum0            ;En cabecera ICMP - Suma de Control LSB
        ICMP_Chksum1            ;En cabecera ICMP - Suma de Control MSB
        ICMP_Identifier0        ;En cabecera ICMP - Identificador LSB
        ICMP_Identifier1        ;En cabecera ICMP - Identificador MSB
        ICMP_Sequence0          ;En cabecera ICMP - Numero de Secuencia LSB
        ICMP_Sequence1          ;En cabecera ICMP - Numero de Secuencia MSB

ENDC


;VARIABLES EN TODOS LOS BANCO
CBLOCK 0X70
        Request_MODBUS_present  ;Controla las banderas cuando se esta tratando una petición MODBUS
                                                        ;bit 0 -> Indica que se esta tratando una petición MODBUS
                                                        ;                       0 = No hay transacciónes
MODBUS en proceso
                                                        ;                       1 = Existe una transacción en
proceso
                                                        ;bit 1 -> Indica que se estan esperando datos desde el RS232
                                                        ;                       0 = No se esperan datos
desde el RS232
                                                        ;                       1 = Se esperan datos desde
el RS232
                                                        ;bit 2 -> Indica que se estan haciendo operaciones de
lectura/escritura
                                                        ;                       en la trama actual y por lo
tanto no debe cerrarse
                                                        ;                       0 = No se haran mas
operaciones de lectura/escritura
                                                        ;                       1 = Se continuaran con las
operaciones de lectura/escritura
                                                        ;bit 3 -> Indica la recepción de una petición para cambio de IP
                                                        ;                       0 = No existe una nueva IP
en la EEPROM
                                                        ;                       1 = Existe una nueva IP en la
EEPROM
        W_copy                  ;Es una copia del registro W cuando atiende interrupciones
        STATUS_copy             ;Es una copia del registro STATUS cuando atiende interrupciones
        PCLATH_copy             ;Es una copia del registro PCLATH cuando atiende interrupciones
ENDC


;VARIABLES EN BANCO UNO (bank 1)
CBLOCK 0XA0

        ;VARIABLES PARA EL MANEJO DE LA TRAMA MODBUS TCP(CAPA 7)
        Receive_CRC_LSB         ;CRC recivido serialmente desde el esclavo MODBUS (LSB)
        Receive_CRC_MSB         ;CRC recivido serialmente desde el esclavo MODBUS (MSB)
        CRC_LSB                         ;CRC para las tramas MODBUS enviadas y recividas serialmente LSB
        CRC_MSB                         ;CRC para las tramas MODBUS enviadas y recividas serialmente MSB
        Data_Modbus_Serial      ;Variable que contiene el dato a calcular el CRC
        MB_PDU_error            ;Bandera que indica la presencia de un error en el PDU MODBUS
        MB_Header_error         ;Bandera que indica la presencia de un error en la cabecera
                                        ;modbus (sin PDU)
        Error_flag              ;Bandera que indica la presencia de un error
                                        ;en el PDU o en la transacción en general
        Error_Code              ;Codigo del error ocurrido en una transacción MODBUS
        Exception_Code          ;Codigo que identifica que tipo de error ha ocurrido
        TCP_bytes_data0         ;Cantidad de bytes enviados ante la petición de un cliente MODBUS LSB
        TCP_bytes_data1         ;Cantidad de bytes enviados ante la petición de un cliente MODBUS MSB
```

```
        length_data_modbus      ;Cantidad de datos enviados en la trama MODBUS
        ;Variables para administrar las transacciones MODBUS
        TCP_transaction_id0             ;identificador de transacción MODBUS LSB
        TCP_transaction_id1             ;identificador de transacción MODBUS MSB
        TCP_protocol_id0                ;identificador de protocolo LSB en cabecera MODBUS
        TCP_protocol_id1                ;identificador de protocolo MSB en cabecera MODBUS
        TCP_length_data0                ;longitud de los datos en el paquete MDOBUS LSB
        TCP_length_data1                ;longitud de los datos en el paquete MODBUS MSB
        TCP_Unit_identifier             ;Identificador de unidad en el paquete MODBUS
        TCP_function_code               ;Codigo de funcion del PDU MODBUS
        TCP_starting_address0   ;Dirección de inicio para la lectura de datos MODBUS LSB
        TCP_starting_address1   ;Dirección de inicio para la lectura de datos MODBUS MSB
        TCP_quantity_inputs0    ;Cantidad de lecturas a realizar LSB
        TCP_quantity_inputs1    ;Cantidad de lecturas a realizar MSB
        TCP_quantity_inputs0_ORG;Cantidad de lecturas a realizar LSB (copia de respaldo)
        sci_data0                       ;Para recivir datos via sci
        sci_data1


        ;VARIABLES PARA LA LECTURA/ESCRITURA EN EL BUFFER DEL RTL
        Register_Address1       ;Direccion del Register_Address que se accedera
        Register_Value1                 ;Register_Value que se pondra en el Register_Address
        Counter1


        ;VARIABLES PARA EL MANEJO DEL PROTOCOLO TCP (CAPA 3)
        tcp_source_port1        ; Puero origen de la perición TCP (MSB)
        tcp_source_port0        ; Puero origen de la perición TCP (LSB)
        tcp_dest_port1                  ; Puero destino de la perición TCP (MSB)
        tcp_dest_port0                  ; Puero destino de la perición TCP (LSB)
        seq0_0                                  ; Número de secuencia palabra 0 (LSB)
        seq0_1                                  ; Número de secuencia palabra 0 (MSB)
        seq1_0                                  ; Número de secuencia palabra 1 (LSB)
        seq1_1                                  ; Número de secuencia palabra 1 (MSB)
        ack0_0                                  ; Número de acuse de recibo palabra 0 (LSB)
        ack0_1                                  ; Número de acuse de recibo palabra 0 (MSB)
        ack1_0                                  ; Número de acuse de recibo palabra 1 (LSB)
        ack1_1                                  ; Número de acuse de recibo palabra 1 (MSB)
        tcp_options                     ; Cantidad de opciones de la cabecera TCP
        flags                                   ; Banderas de la cabecera TCP
        max_seg0                        ; Opciones en la cabecera TCP LSB
        max_seg1                        ; Opciones en la cabecera TCP MSB
        window0                                 ; Valor de la ventana de la cabecera TCP byte 0 (LSB)
        window1                                 ; Valor de la ventana de la cabecera TCP byte 1 (MSB)
        kind                                    ; Tipo de opciones
        data_flags0                     ; Banderas obtenidas en la cabecera TCP LSB
        data_flags1                     ; Banderas obtenidas en la cabecera TCP LSB
        offset                                  ; Offset obtenido en la caberca TCP
        tcp_data_Len0                   ; cantidad de datos contenida en la trama TCP LSB
        tcp_data_Len1                   ; cantidad de datos contenida en la trama TCP LSB
        tcp_Chksum0                             ; Valor del Checksum para la cabecera TCP LSB
        tcp_Chksum1                             ; Valor del Checksum para la cabecera TCP MSB
        tcp_length0                     ; Longitud de cabecera LSB
        tcp_length1                     ; Longitud de cabecera MSB
        ;Variables de socket
        html_socket                     ; Estados del protocolo TCP
        port0                                   ; Establece el puerto del socket actualmente en uso LSB
        port1                                   ; Establece el puerto del socket actualmente en uso MSB
        ip0                                             ; Establece el numero de IP que establecio un socket
        ip1                                             ; Establece el numero de IP que establecio un socket
        ip2                                             ; Establece el numero de IP que establecio un socket
        ip3                                             ; Establece el numero de IP que establecio un socket
        xm_ack0_0                       ; Almacena el numero de acuse a transmitir
        xm_ack0_1                       ; Almacena el numero de acuse a transmitir
        xm_ack1_0                       ; Almacena el numero de acuse a transmitir
        xm_ack1_1                       ; Almacena el numero de acuse a transmitir

        xm_seq0_0                       ; Almacena el numero de secuencia a transmitir
        xm_seq0_1                       ; Almacena el numero de secuencia a transmitir
        xm_seq1_0                       ; Almacena el numero de secuencia a transmitir
        xm_seq1_1                       ; Almacena el numero de secuencia a transmitir

ENDC


;VARIABLES EN BANCO DOS (bank 2)
CBLOCK 0X120
        ;VARIABLES PARA CREAR RESPALDO DE LAS CARACTERISITCAS DEL SOCKET
        ;CUANDO SE ATIENTE A UNA NUEVA PETICIÓN ( QUE NO SEA LA APERTURA
        ;DE UN NUEVO SOCKET)
        Bck_Counter
        Bck_Chksum0
        Bck_Chksum1
        Bck_sourceParam0
        Bck_sourceParam1
        Bck_sourceParam2
        Bck_sourceParam3
        Bck_sourceParam4
        Bck_sourceParam5
        Bck_Source_IP0
        Bck_Source_IP1
        Bck_Source_IP2
```

```
Bck_Source_IP3
Bck_tcp_dest_port0
Bck_tcp_dest_port1
Bck_tcp_source_port0
Bck_tcp_source_port1
Bck_xm_seq0_0
Bck_xm_seq0_1
Bck_xm_seq1_0
Bck_xm_seq1_1
Bck_xm_ack0_0
Bck_xm_ack0_1
Bck_xm_ack1_0
Bck_xm_ack1_1
Bck_data_flags0
Bck_data_flags1


;VARIALBES PARA EL MANEJO DE LA ESCRITURA DE REGISTROS MODBUS
TCP_count_bytes                 ;Contador de bytes para la escritura de registros MODBUS
TCP_quantity_register0  ;Cantidad de registros a escribir (LSB)
TCP_quantity_register1  ;Cantidad de registros a escribir (MSB)
TCP_Byte_to_Write       ;Byte a escribir en la memoria MODBUS del esclavo
Data_SCI_Rcv                    ;Se almacena el dato recivido via SCI
Count_Data_SCI                  ;Se lleva en conteo de los datos recividos via SCI
CounterExt                      ;Usada en los lazos para llevar en conteo
INSD2                           ;4 bits mas significativos del hexa recivido
INSD1                           ;4 bits menos significativos del hexa recivido

;VAIABLE QUE DEFINE EL PUERTO DEL QUE PROCEDE UNA PETICIÓN
Socket_Number                   ;Define el numero de puerto que desea establecer el socket

;VARIABLES PARA LA LECTURA/ESCRITURA A LA EEPROM DEL PIC16F877A
VALOR_EEPROM                    ;Valor a gravar en la EEPROM
ADDR_EEPROM                     ;Dirección en donde se almacenara el dato

ENDC


;*****************************************************************
;*****************************************************************        *
; DESCRIPCIÓN:          Reset, llamado a la función Initialize                  *       *
;*****************************************************************        *
        ORG                     0X00;
                                *
;*****************************************************************
        GOTO            Initialize


;*****************************************************************
;*****************************************************************        *
; DESCRIPCIÓN: Manejo de interrupciones                                                 *       *
;*****************************************************************        *
        ORG                     0X04;
                                *
;*****************************************************************

        ;********************* Se salva el contexto **********************
        movwf   W_copy                  ;Se guarda una copia del registro W
        swapf   STATUS,W        ;Se hace una copia de STATUS
        clrf    STATUS
        movwf   STATUS_copy
        movf    PCLATH,W        ;Se hace una copia de PCLATH
        movwf   PCLATH_copy
        clrf    PCLATH

        ;Se manejan 2 tipos de interrupciones: La primera generada por el RTL8019AS y se debe
        ;a la recepción de un paquete sin errores. Y la segunda por la llegada de un paquete
        ;desde el puerto RS232 enviado por el medidor de energia.

        ;Prueba si la interrupción se debe a la llegada de un paquete desde el RTL8019AS
        btfss           INTCON,T0IF
        goto            OTHER_INT ;Si no es paquete, ir a interrupcion desde el puerto RS232
                ;*****************************************************************************
                ;* LA INTERRUPCIÓN SE GENERO POR LA LLEGADA DE UN PAQUETE DESDE EL MEDIO ETHERNET *
                ;*****************************************************************************
                ; Se configura el TMR0 de nuevo         y se limpia la bandera
                bcf             INTCON,T0IF
                movlw   0XFF
                movwf   TMR0

                ;Se revisa si existen una petición en proceso
                btfss   Request_MODBUS_present,0
                goto    NO_BCK_VARS
                        ;Se hace una copia de respaldo de las variables
                        bsf             PCLATH,4
                        bsf             PCLATH,3        ;PAGINA 3 DE PROGRAMA
                        call    save_value_vars
                        bcf             PCLATH,4
                        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                NO_BCK_VARS
```

```
			;Se llama a la función para obtención de la trama ETHERNET
			call		ReceiveFrame

			;Si se recive una petición a tratar en este paquete no se
			;restauran las variabes
			btfsc		Request_MODBUS_present,2
			goto		REQUEST_IN_CURRENT_PCK
					;Se revisa de nuevo si existe un petición en proceso
					;Con el fin de restarurar la variables
					btfss		Request_MODBUS_present,0
					goto		NO_RESTORE_VARS
							;Se devuelve el valor original a las variables
							bsf				PCLATH,4
							bsf				PCLATH,3			;PAGINA 3 DE PROGRAMA
							call		restore_value_vars
							bcf				PCLATH,4
							bcf				PCLATH,3			;PAGINA 0 DE PROGRAMA
					NO_RESTORE_VARS
			REQUEST_IN_CURRENT_PCK

			goto		EXIT_INT


		;Prueba si la interrupción fue generada por el medidor (entrada de datos via SCI)
		OTHER_INT
		btfss				PIR1,RCIF
		goto				EXIT_INT
			;**********************************************************************
			;* LA INTERRUPCIÓN SE GENERO POR LA LLEGADA DE UN DATO VIA SCI (RS232)  *
			;**********************************************************************
			;La bandera RCIF es limpiada por hardware
			;Se obtiene el dato y se almacena en Data_SCI_Rcv
			call		RS232_Rx
			BANK2
			movwf		Data_SCI_Rcv

			;Se incrementa la variable Count_Data_SCI
			incf		Count_Data_SCI,F
		EXIT_INT

		;******************** Se retorna al cotexto original *******************
		BANK0
		movf		PCLATH_copy,W
		movwf		PCLATH
		swapf		STATUS_copy,W
		movwf		STATUS
		swapf		W_copy,F
		swapf		W_copy,W

		retfie ; Fin de la función Int_Management


;*********************************************************************
;*****************************************************************			*
; NOMBRE:			Initialize										*
		*
; DESCRIPCIÓN:		Asignamos la MAC y de la tarjeta		*		*
;						Se inicializa el RTL8019AS y el PIC16F887		*		*
;						El PIC queda en espera de una interrupción		*		*
;						El RTL8019AS empieza la recepción de paquetes	*		*
;*****************************************************************			*
Initialize;
					*
;*********************************************************************
;Basado en "IEEE 802.3 Part 3:Carrier sense multiple acces with collision Detection
;(CSMA/CD) access method and physical Layer specification", Ver 4.2.7.5


		;Carga la MAC local desde la memoria EEPROM DEL PIC
		bsf				PCLATH,4
		bsf				PCLATH,3			;PAGINA 3 DE PROGRAMA
		call		LOAD_MAC_FROM_EEPROM
		bcf				PCLATH,4
		bcf				PCLATH,3			;PAGINA 0 DE PROGRAMA

		;Carga la dirección IP local desde la memoria EEPROM DEL PIC
		bsf				PCLATH,4
		bsf				PCLATH,3			;PAGINA 3 DE PROGRAMA
		call		LOAD_IP_FROM_EEPROM
		bcf				PCLATH,4
		bcf				PCLATH,3			;PAGINA 0 DE PROGRAMA


		;Asignacion del estado del socket TCP actual
		movlw		LISTEN
		BANK1
		movwf		html_socket
		clrf		Error_flag
		clrf		Error_Code
		clrf		MB_Header_error
```

```
		clrf		MB_PDU_error
		clrf		Exception_Code
		clrf		Request_MODBUS_present
		BANK0

		;Inicializo el PIC16F877
		call InitializePIC
		;Inicializo el RTL8019AS
		call InitializeRTL

		;Saco al NIC de LOOP BACK y se configura en modo NORMAL
		movlw		TCR
		movwf		Register_Address
		movlw		0x00
		movwf		Register_Value
		call		outport

;///////////////////////////////////
		;SE IMPRIME EL VALOR DE LA IP
		movf		Host_IP0,W
		call		RS232_Tx

		movf		Host_IP1,W
		call		RS232_Tx

		movf		Host_IP2,W
		call		RS232_Tx

		movf		Host_IP3,W
		call		RS232_Tx

;/////////////////////////////////

		;El PIC16F877 se queda en espera de las interrupciones
		;Interrupcion del RTL8019AS o desde el medidor de energia

MAIN_LOOP

			;Se verifica si esta presionado el boton RESET IP
			;Que indica el re-establecimineto de la IP
			BANK0
			btfsc		INT_BUS,0X00			;Verifica el estado del boton RESET IP
			goto		NOT_RESET_PRESENT  ;No se ha presionado el boton
					;El boton de RESET IP se ha presionado
					bsf			PCLATH,0x04
					bsf			PCLATH,0x03			;PAGINA 3 DE PROGRAMA
					call		Reset_IP
					bcf			PCLATH,0x04
					bcf			PCLATH,0x03			;PAGINA 0 DE PROGRAMA
			NOT_RESET_PRESENT

			;se comprueba la existencia de una petición MODBUS desde
			;el medio ETHERNET
			BANK1
			btfss		Request_MODBUS_present,0
			goto		EXIT_MAIN_LOOP

					bcf			PCLATH,4
					bsf			PCLATH,3		;PAGINA 1 DE PROGRAMA
					call		treatment_to_Request_MODBUS
					bcf			PCLATH,4
					bcf			PCLATH,3		;PAGINA 0 DE PROGRAMA

			EXIT_MAIN_LOOP

			;Revisa constantemente la existencia de un OVERRUN en la recepción de datos
			;via RS232/SCI
			BANK0
			btfss		RCSTA,OERR
			goto		NO_OVERRRUN_ACTIVE
					;Existio un overrun y se debe limpiar la bandera
					bcf		RCSTA,CREN
					bsf		RCSTA,CREN
			NO_OVERRRUN_ACTIVE

		goto MAIN_LOOP

		;Retardo de 10ms
		movlw		0x0A
		movwf		Mili_Second
		call		delay_ms
;Fin de la función Initialize


;*************************************************************************
;**********************************************************************		*
;NOMBRE:			InitializePIC									*
		*
;DESCRIPCIÓN:		Configura el PIC16F877 para empezar a operar		*		*
;ENTRADAS:				ADDR_BUS  Es el puerto asignado para manejar las *		*
;									direcciones a los registro del RTL		*		*
```

```
;                                       DATA_BUS  Es el puerto asignado para manejar los *          *
;                                                            datos enviados y recividos desde el RTL          *          *
;                                       CTRL_BUS  Es el puerto asignado para manejar las *          *
;                                                            lineas de lectura y escritura al RTL     *          *
;                                       INT_BUS             Es el puerto asignado para manejar la  *         *
;                                                            interrupción generada por el RTL              *          *
;                                       RESET               Pin de control para el reseteo del RTL  *         *
;                                       IOWB                Pin de control para escritura al RTL     *         *
;                                       IORB                Pin de control para lectura al RTL                 *          *
;SALIDAS:            Ninguna
                     *          *
;**********************************************************************          *
InitializePIC;
                                *
;**************************************************************************
        BANK1                                           ;Selecciółna banco 1

        ;Se cofiguran los puertos A y E como digitales (NO analogicos)
        movlw    b'00000111'          ;Puerto A y E son digitales
        movwf    ADCON1

        ;Se configuran los puertos como entradas ó salidas
        clrf     ADDR_BUS             ;Bus de direcciones se establece como salidas (Puerto B)
        movlw    0xFF
        movwf    DATA_BUS             ;Bus de datos se establece como entrada (Puerto D)
        movlw    b'11111101'
        movwf    INT_BUS                        ;Bus de interrupción se establece como entrada (Puerto A)
        movlw    b'10000000'          ;Se establecen los pines de reseteo, lectura y
                                                            ;escritura como salidas
        movwf    CTRL_BUS


        ;Configuracion de USART en TX
        movlw    b'00100100'
        movwf    TXSTA                ;TX en On, modo asíncrono con 8 bits y alta velocidad

        ;Configuracion de USART relacion de BAUDIOS
        movlw    .129                 ;Cristal de 20MHz
        movwf    SPBRG                ;9600 baudios con Fosc=20MHz

        ;Configuracion de USART en RX
        BANK0                                           ;Selecciółna el banco 0
        movlw    b'10010000'          ;Se configura el RX y USART en On
        movwf    RCSTA

        ;Se configura el TMR1 como timer y se limpia la bandera
        bcf               PIR1,TMR1IF
        clrf     T1CON

        ;Se configura el TMR0 como contador
        BANK1
        movlw    b'00101000'  ;Se configura el TMR0 para que incremente con reloj externo
        movwf    OPTION_REG           ;Se configura para que reaccione a flancos de subida del reloj
        BANK0
        movlw    0xFF                 ;Asigna FF al TMR0
        movwf    TMR0                 ;Al recibir un flanco de subida se activara la interrupción

        ;Se habilitan las interrupciones
        BANK1
        bsf               PIE1,RCIE   ;Se habilitan las interrupciones por la recepción de
                                                ;datos via SCI (rs232)
        movlw    b'11100000'
        movwf    INTCON               ;Habilita interrupción del TMR0 y
                                                ;las interrucciones GLOBALES

        ;Se establece como activas altas o activas bajas las señales de control
        BANK0
        bsf               INT_BUS,0x01          ;Activacion de LED
        bcf               CTRL_BUS,RESET        ; Activa alta, se establece baja para completar el
                                                            ;ciclo de reseteo
        bsf               CTRL_BUS,IOWB         ;Activa baja
        bsf               CTRL_BUS,IORB         ;Activa baja
        clrf     ADDR_BUS             ;Establece una dirección por defecto de cero

        return; Final de la función InitializePIC


;**************************************************************************
;**********************************************************************          *
;NOMBRE:             InitializeRTL                                                                    *
        *
;DESCRIPCIÓN:        Configura el RTL8019AS para empezar a operar            *          *
;ENTRADAS:                     HOST_MAC0-5         Dirección fisica (MAC, 6 bytes)          *          *
;                                       RCV_BUF_START       Comienzo del buffer de recepción      *          *
;                                       XMT_BUF_START       Comienzo del buffer de transmisión    *          *
;SALIDAS:            Ninguna
                     *          *
;**********************************************************************          *
InitializeRTL;
                                *
;**************************************************************************
```

```
BANK0                                   ;Selecciona el banco 0

movlw   CR                              ;Registro de Comando
movwf   Register_Address
movlw   0x21                    ;Pagina 0, Abortar DMA, Detener NIC
movwf   Register_Value
call    outport

movlw   0x19                    ;Retardo de 25 ms
movwf   Mili_Second
call    delay_ms

movlw   NIC_RESET               ;Registro de Reseteo de la NIC
movwf   Register_Address
movlw   0xFF                    ;Se resetea al escribir 0xFF
movwf   Register_Value
call    outport

movlw   0x19                    ;Retardo de 25 ms
movwf   Mili_Second
call    delay_ms

movlw   DCR                             ;Registro de configuración de datos
movwf   Register_Address
movlw   0x48                    ;Modo doble 16-bit, Operación Normal,
movwf   Register_Value          ;Envia comando de paquete no ejecutado
call    outport                 ;8 bytes de "FIFO threshold"

movlw   RBCR0                   ;Registro de conteo de bytes remoto 0
movwf   Register_Address
movlw   0x00                    ;Se establece a cero
movwf   Register_Value
call    outport

movlw   RBCR1                   ;Registro de contador de byes remoto 1
movwf   Register_Address
movlw   0x00                    ;Se establece a cero
movwf   Register_Value
call    outport

movlw   RCR                             ;Registro de configuracion de recepción
movwf   Register_Address
movlw   0x0C                    ;Monitoreo desabilitado, Promiscuo desabilitado
movwf   Register_Value          ;No acepta multicast, Acepta broadcast
call    outport                 ;No acepta paquetes menores a 64 bytes
                                        ;No acepta paquetes con errores

movlw   TPSR                    ;Registro de comienzo de pagina de transmisión
movwf   Register_Address
movlw   RCV_BUF_START           ;Se le asigna un valor de 0x40
movwf   Register_Value          ;Establece la dirección de la pagina de inicio
call    outport                 ;de el paquete a ser transmitido

movlw   TCR                             ;Registro de  configuracion de transmisión
movwf   Register_Address
movlw   0x02                    ;Abilita offset de colisión, operacion en lazo interno,
movwf   Register_Value
call    outport

movlw   BNDRY                   ;Registro de borde
movwf   Register_Address
movlw   RCV_BUF_START           ;Se le asigna un valor de 0x40
movwf   Register_Value
call    outport                 ;Indica la dirección de la ultima pagina del buffer
                                        ;de recepcion leida por el host

movlw   PSTART                  ;Registro de comienzo de pagina
movwf   Register_Address
movlw   RCV_BUF_START           ;Se le asigna un valor de 0x40
movwf   Register_Value
call    outport                 ;Establece en la dirección de inicio de pagina
                                        ;de el buffer de anillo de recepción

movlw   PSTOP                   ;Regisro de final de pagina
movwf   Register_Address
movlw   XMT_BUF_START           ;Se le asigna un valor de 0x54
movwf   Register_Value          ;
call    outport                 ;Establece la dirección de fial de pagina
                                        ;de el buffer de anillo de recepción

movlw   ISR                             ;Regitro de estados de interrupciones
movwf   Register_Address
movlw   0xFF                    ;Limpia los registros de estado de interrupción
movwf   Register_Value
call    outport

movlw   IMR                             ;Registro de mascara de  interrupciones
movwf   Register_Address
movlw   0x01                    ;Registro para configurar las mascaras de las interrupciones
movwf   Register_Value
call    outport
```

```
        movlw   CR                                              ;Registro de comando
        movwf   Register_Address
        movlw   0x61                                    ;Pagina 1, Abortar DMA, Detener RTL
        movwf   Register_Value
        call    outport

        movlw   PAR0                                    ;Registro de dirección fisica 0
        movwf   Register_Address
        movf    Host_Addr0,W
        movwf   Register_Value
        call    outport

        movlw   PAR1                                    ;Registro de dirección fisica 1
        movwf   Register_Address
        movf    Host_Addr1,W
        movwf   Register_Value
        call    outport

        movlw   PAR2                                    ;Registro de dirección fisica 2
        movwf   Register_Address
        movf    Host_Addr2,W
        movwf   Register_Value
        call    outport

        movlw   PAR3                                    ;Registro de dirección fisica 3
        movwf   Register_Address
        movf    Host_Addr3,W
        movwf   Register_Value
        call    outport

        movlw   PAR4                                    ;Registro de dirección fisica 4
        movwf   Register_Address
        movf    Host_Addr4,W
        movwf   Register_Value
        call    outport

        movlw   PAR5                                    ;Registro de dirección fisica 5
        movwf   Register_Address
        movf    Host_Addr5,W
        movwf   Register_Value
        call    outport

        movlw   CURR                                    ;Registro de pagina actual
        movwf   Register_Address
        movlw   RCV_BUF_START           ;Se le asigna 0x40
        movwf   Register_Value
        call    outport                                 ;Apunta a la dirección de la primera pagina
                                                                ;del primer buffer de recepción a ser usada
                                                                ;para una recepción de paquete

        movlw   CR                                              ;Registro de comando
        movwf   Register_Address
        movlw   0x22                                    ;Coloca el RTL en modo inicio
        movwf   Register_Value
        call    outport

        return; Fin de la función InitializeRTL


; Trata el paquete que ha llegado
;*****************************************************************************
;*****************************************************************  *
;NOMBRE:            ReceiveFrame (Segun Modelo TCP/IP)                                      *       *
;CAPA:                      1 -> ACCESO A RED
        *       *
;DESCRIPCIÓN:       El RTL8019AS ha generado una interrupción por       *       *
;                           recepción un paquete sin errores                                        *       *
;ENTRADAS:                  Adquiere el paquete desde el buffer de anillo del   *       *
;                           RTL8019AS
                           *       *
;SALIDAS:           Ninguna
                   *       *
;*****************************************************************  *
ReceiveFrame
;Segun "IEEE 802.3 Part 3", 2002, Ver "4.2.9 Frame Reception"

  ;Se inicia comprobando si existe un paquete en buffer de anillo del RTL
  TEST_PCK_IN_RING
        BANK0
        ;Se cambia a la pagina 1 del NIC
        movlw   CR
        movwf   Register_Address
        movlw   0x62
        movwf   Register_Value
        call    outport

        ;Se obtiene el valor de CURRENT
        movlw   CURR
        movwf   Register_Address
        call    inport
        movf    Register_Value,0
        movwf   In
```

```
;Se cambia a la pagina 0 del NIC
movlw       CR
movwf       Register_Address
movlw       0x22
movwf       Register_Value
call        outport

;Se limpia el registro de interrupciones
movlw       ISR
movwf       Register_Address
movlw       0XFF
movwf       Register_Value
call        outport

;Se obtiene el valor de BOUNDARY
movlw       BNDRY
movwf       Register_Address
call        inport
movf        Register_Value,0
movwf       Out

;Se Compara CURRENT con BOUNDARY
;De ser iguales indica que ya no existen paquetes a tratar
;en el buffer del RTL8019AS
movf        In,W
subwf       Out,W
btfsc       STATUS,Z
goto        EXIT_POLL  ;Salir si son iguales

            ;////////////Desencapsulación de Datos Recividos//////////////////
            ;/CAPA: 1 -> ACCESO A RED  (Segun Modelo TCP/IP)                                                    /
            ;/DESCRIPCIÓN:          Se obtienen los diferetes campos del paquete que /
            ;/                                                   se guardo en el buffer de anillo del RTL       /
            ;/                                                   los campos importantes se almacenan en variables /
            ;/////////////////////////////////////////////////////////////
            ;Segun "IEEE 802.3 Part 3", 2002, Ver "4.2.9 Frame Reception"

            ;Cabecera de paquetes en buffer del RTL

  ;Hay datos presentes, por lo tanto se deben tratar
            ;Se configura la lectura remota DMA desde BOUNDARY (out).
            ;Se lee la cabecera del paquete asignada en el anillo( 4 bytes )
            movlw       READ                                    ;Se haran lecturas remotas
            movwf       Read_Write              ;Configuro para lectura
            clrf        Addr0                                   ;Las lecturas se haran desde el registro
            movf        Out,0                                   ;BOUNDARY ( out)
            movwf       Addr1
            call        remote_dma_setup;Se configura el canal remoto DMA

            ;Cabecera para cada paquete almacenado en el buffer de anillo del RTL
            ;Esta basado en DP8390 "Network Interface Controller: An Introductory Guide",
            ;National Semiconductor, Aplication Note 475, 1993
            ;+------------+------------+------------+------------+----
            ;|   STATUS   | NEXT PAGE |  LENGTH L                | LENGTH L  |
            ;|  (8 bytes) |  (8 bytes) |  (8 bytes) |  (8 bytes) | ...
            ;+------------+------------+------------+------------+----
            ;Lectura desde puerto remoto DMA (Leemos STATUS)
            movlw       NIC_DATA
            movwf       Register_Address
            call        inport

;Lectura desde puerto remoto DMA (Leemos NEXT PAGE)
            call        inport
            movf        Register_Value,W
            movwf       Next_Ptr_Pck

;Lectura desde puerto remoto DMA (Leemos LENGTH L)
            call        inport

;Lectura desde puerto remoto DMA (Leemos LENGTH H)
            call        inport

            ;Cabecera Ethernet (Segun "IEEE Standars 802.3 Part 3",2002, ver 3.1.1 )
            ;+------------------+
            ;|     Preambulo    |                          ;Previamente suprimido por el RTL
            ;|    ( 7 Bytes )                |
            ;+------------------+
            ;|                                SFD                          |                          ;Previamente suprimido por el RTL
            ;|   ( 1 Bytes )                 |
            ;+------------------+
            ;| Dirección Destino |
            ;|   ( 6 Bytes )                 |
            ;+------------------+
            ;| Dirección Fuente              |
            ;|   ( 6 Bytes )                 |
            ;+------------------+
            ;|   Longitud/Tipo               |
            ;|   ( 2 Bytes )                 |
            ;+------------------+
            ;|  MAC Client Data  |                          ;Incluye el PAD
            ;|    ( N Bytes)                 |
```

```
                ;+------------------+
                ;|Chequeo de secuen- |
                ;|cia de trama (CRC) |                ;Previamente suprimido por el RTL
                ;|   ( 4 Bytes )          |
                ;+------------------+

                ;El campo de Preambulo y SFD han sido previamente suprimidos
                ;por el RTL8019AS y no estan almacenados en el buffer de anillo
                ;El campo Chequeo de secuencia de trama tambien ha sido suprimido
                ;y previamente analizado por el RTL8019AS

;Lectura desde puerto remoto DMA (Lectura de  Dirección Destino 0)
                ;Lectura de destinationParam0
                call    inport

;Lectura desde puerto remoto DMA (Lectura de Dirección Destino 1)
                ;Lectura de destinationParam1
                call    inport

;Lectura desde puerto remoto DMA (Lectura de Dirección Destino 2)
                ;Lectura de destinationParam2
                call    inport

;Lectura desde puerto remoto DMA (Lectura de Dirección Destino 3)
                ;Lectura de destinationParam3
                call    inport

;Lectura desde puerto remoto DMA (Lectura de Dirección Destino 4)
                ;Lectura de destinationParam4
                call    inport

;Lectura desde puerto remoto DMA (Lectura de Dirección Destino 5)
                ;Lectura de destinationParam5
                call    inport

;Lectura desde puerto remoto DMA (Lectura de Dirección Fuente 0)
                call    inport
                movf    Register_Value,W
                movwf   sourceParam0

;Lectura desde puerto remoto DMA (Lectura de Dirección Fuente 1)
                call    inport
                movf    Register_Value,W
                movwf   sourceParam1

;Lectura desde puerto remoto DMA (Lectura de Dirección Fuente 2)
                call    inport
                movf    Register_Value,W
                movwf   sourceParam2

;Lectura desde puerto remoto DMA (Lectura de Dirección Fuente 3)
                call    inport
                movf    Register_Value,W
                movwf   sourceParam3

;Lectura desde puerto remoto DMA (Lectura de Dirección Fuente 4)
                call    inport
                movf    Register_Value,W
                movwf   sourceParam4

;Lectura desde puerto remoto DMA (Lectura de Dirección Fuente 5)
                call    inport
                movf    Register_Value,W
                movwf   sourceParam5

                ;Lectura desde puerto remoto DMA (Lectura de TYPE/LENGTH, MSB)
                call    inport
                movf    Register_Value,W
                movwf   lengthOrTypeParam1

                ;Lectura desde puerto remoto DMA (Lectura de TYPE/LENGTH, LSB)
                call    inport
                movf    Register_Value,W
                movwf   lengthOrTypeParam0

                ;Lectura de los datos desde el puerto remoto DMA
                ;Los datos se leen directamente desde el buffer de anillo del RTL
                ;La subcapa de control MAC decide a que capa envia el control de los datos

                ;//////////////////Subcapa de control MAC //////////////////////
                ;/CAPA:                 1 -> ACCESO A RED (Segun Modelo TCP/IP)                                 /
                ;/DESCRIPCIÓN:          Decide a que capa va destinado el paquete                       /
                ;/                                                       y envia el control de los datos a dicha capa    /
                ;/                                                       Existen 2 opciones:
                        /
                ;/                                                       ARP-> Dirigido a la capa 2 (ENLACE DE DATOS)    /
                ;/                                                       IP -> Dirigido a la capa 3 (RED)                                        /
                ;////////////////////////////////////////////////////////////
                ;Ver "IEEE 802.3 Part 3", 2002, Ver "2.2.2 Model used for service spefication"
                ;Verificar si ha sido una petición ARP
                movlw   ARP0
                subwf   lengthOrTypeParam0,W
```

```
btfss       STATUS,Z
goto        SWITCH_IP
movlw       ARP1
subwf       lengthOrTypeParam1,W
btfss       STATUS,Z
goto        SWITCH_IP
            ;/////////Protocolo de resolución de direcciones(ARP)////////////
            ;/CAPA:                1 -> ACCESO A RED (Segun Modelo TCP/IP)                    /
            ;/DESCRIPCIÓN:         Protocolo para la resolucion de direcciones        /
            ;/                                   Maneja las peticiones broadcas para la obtener    /
            ;/                                   las MAC de las maquinas en la red
/
            ;/ENTRADAS:                    NICA_DATA
                        /
            ;/                                              Datos almacenados en el anillo del RTL          /
            ;/SALIDAS:             Responde a la petición ARP con TransmitFrameARP /
            ;///////////////////////////////////////////////////////////////////

            ;Formato de la trama ARP
            ;Segun RFC 826 "Protocolo de Resolución de direcciones", 1982

            ;+-------------------------+------------------------+
            ;|  Espacio para dirección | Espacio para Direccion    |
            ;|      de Hardware       |      de protocolo      |
            ;|                (Ethernet = 1)                |     (IP = 0x0800)                    |
            ;|      ( 16 bits )       |      ( 16 bits )       |
            ;+------------+------------+------------------------+
            ;| Longitud de| Longitud de|   Codigo de operacion     |
            ;| direccion  | direccion  |(1:Peticion 2:Respuesta)|
            ;| de Hardware|de protocolo|                                                          |
            ;| (MAC=0x06) |(IPV4 =0X04)|                                                          |
            ;|  (8 bits)  |  (8 bits)  |         16 bits       |
            ;+------------+------------+------------------------+
            ;|    Dirección MAC del emisor   32 bits           |
            ;+--------------------------------------------------+
            ;|    Dirección IP del emisor                  32 bits           |
            ;+--------------------------------------------------+
            ;|    Dirección MAC del receptor 32 bits           |
            ;+--------------------------------------------------+
            ;|    Dirección IP del receptor                32 bits           |
            ;+--------------------------------------------------+

            ;Se descarta el Espacio para dirección (2 bytes), Dirección
            ;de protocolo (2 bytes), Longitud de dirección de hardware (1 byte),
            ;longitud de dirección de protocolo (1 byte), Codigo de operación (2 bytes)
            movlw       NIC_DATA
            movwf       Register_Address        ;Se direcciona al buffer de anillo del RTL
            movlw       0x08                                    ;Se descartan 8 bytes del buffer
            movwf       Counter                                 ;Se le asigna 8 al contador
            LOOP_FOR1
                        call        inport
                        decfsz      Counter,F
            goto        LOOP_FOR1                               ;Si no se ha descartado 8 bytes, regresa

            ;Se descarta la dirección del hardware fuente, puesto que ya
            ;se tiene (se obtuvo desde la cabecera Ethernet)
            movlw       0x06
            movwf       Counter                                 ;Se le asigna 6 al contador
            LOOP_FOR2
                        call        inport
                        decfsz      Counter,F
            goto        LOOP_FOR2                               ;Si no se ha descartado 6 bytes, regresa

            ;Obtenemos la dirección IP del emisor del paquete
            call        inport
            movf        Register_Value,W
            movwf       Source_IP0

            call        inport
            movf        Register_Value,W
            movwf       Source_IP1

            call        inport
            movf        Register_Value,W
            movwf       Source_IP2

            call        inport
            movf        Register_Value,W
            movwf       Source_IP3

            ;Se descarta la  dirección del hardware destino
            ;El RTL la revisa previamente ( Acepta broadcast y direccionadas
            ;especificamente a esta tarjeta)
            movlw       0x06
            movwf       Counter

            LOOP_FOR3
                        call        inport
                        decfsz      Counter,F
            goto        LOOP_FOR3

            ;Se obtiene la dirección IP destino
```

```
                ;Es la IP local de la tarjeta
                call    inport
                movf    Register_Value,W
                movwf   Dest_IP0

                call    inport
                movf    Register_Value,W
                movwf   Dest_IP1

                call    inport
                movf    Register_Value,W
                movwf   Dest_IP2

                call    inport
                movf    Register_Value,W
                movwf   Dest_IP3

                ;Se compara la dirección IP destino del paquete con la IP local
                ;Para confirmar que la petición ARP está dirigida a esta tarjeta
                movf    Dest_IP0,W
                subwf   Host_IP0,W
                btfss   STATUS,Z
                goto    EXIT_ARP

                movf    Dest_IP1,W
                subwf   Host_IP1,W
                btfss   STATUS,Z
                goto    EXIT_ARP

                movf    Dest_IP2,W
                subwf   Host_IP2,W
                btfss   STATUS,Z
                goto    EXIT_ARP

                movf    Dest_IP3,W
                subwf   Host_IP3,W
                btfss   STATUS,Z
                goto    EXIT_ARP

                ;Se responde a la petición ARP
                call    TransmitFrameARP

EXIT_ARP        ;Salir de ARP
goto    EXIT_SWITCH

;Verificar si ha sido una petición IP
SWITCH_IP
movlw   IP0
subwf   lengthOrTypeParam0,W
btfss   STATUS,Z
goto    EXIT_SWITCH

movlw   IP1
subwf   lengthOrTypeParam1,W
btfss   STATUS,Z
goto    EXIT_SWITCH


                ;////////////////////Protocolo Internet (IP)//////////////////
                ;/CAPA:                  2 -> INTERNET (Segun Modelo TCP/IP)                                     /
                ;/DESCRIPCIÓN:           Protocolo de internet, maneja dos tipos de                      /
                ;/                                              peticiones: ICMP y TCP
                /
                ;////////////////////////////////////////////////////////////

                ;Formato de la trama IP (Segun RFC 791 "Protocolo Internet", 1981)
                ;    0                   1                   2                   3
                ;    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ;   |Versión|  IHL  | Tipo Servicio |         Longitud Total        |
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ;   |        Identificación         |Flags|       Posición          |
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ;   |Tiempo de Vida |   Protocolo   | Suma de Control de Cabecera   |
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ;   |                    Dirección de Origen                        |
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ;   |                    Dirección de Destino                       |
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ;   |                  Opciones                     |    Relleno    |
                ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                ;////////////////Desencapsulación de cabecera IP//////////////
                ;/CAPA:                  2 -> INTERNET (Segun Modelo TCP/IP)                                     /
                ;/DESCRIPCIÓN:           Obtienen los datos en cada campo de la cabecera /
                ;/                                               IP del paquete almacenado en el RTL8019AS       /
                ;////////////////////////////////////////////////////////////

                ;Versión (4 bits)
                ;Longitud de cabecera de Internet (IHL) (4 bits)
                movlw   NIC_DATA
                movwf   Register_Address
```

```
call        inport
movf        Register_Value,W

;Se obtiene la longitud de las opciones
andlw       0x0F
movwf       hdr_len                 ;Se obtiene la longitud de la cabecera total
bcf                 STATUS,C
rlf                 hdr_len,F       ;Se rota para obtener la longitud en bytes
rlf                 hdr_len,F
movf        hdr_len,W   ;Se guarda la longitud de cabecera en Opt_Len
movwf       Opt_Len
movlw       0x14                    ;Se guarda 20 (decimal) en W (tamaño mínimo cabecera)
subwf       Opt_Len,F   ;Resta la cabecera base de la longitud total
                                                        ;para obtener los bytes de opciones en la cabera

;Descartar el tipo de servicio (1 byte)
call        inport

;Longitud Total del paquete IP (2 bytes)
call        inport
movf        Register_Value,W
movwf       IP_Length1
call        inport
movf        Register_Value,W
movwf       IP_Length0

;Se obtiene la identificación identificación
call        inport
movf        Register_Value,W
movwf       Identification1

call        inport
movf        Register_Value,W
movwf       Identification0

;Fragmento (Banderas y posición)
call        inport
movf        Register_Value,W
movwf       Fragment1

call        inport
movf        Register_Value,W
movwf       Fragment0

;Se descarta el tiempo de vida (Los paquetes estan aquí)
call        inport

;Protocolo IP
call        inport
movf        Register_Value,W
movwf       IP_Protocol

;Suma de Control de Cabecera
call        inport
movf        Register_Value,W
movwf       Chksum1

call        inport
movf        Register_Value,W
movwf       Chksum0

;Dirección IP de paquete fuente
call        inport
movf        Register_Value,W
movwf       Source_IP0

call        inport
movf        Register_Value,W
movwf       Source_IP1

call        inport
movf        Register_Value,W
movwf       Source_IP2

call        inport
movf        Register_Value,W
movwf       Source_IP3

;Dirección IP de destino

call        inport
movf        Register_Value,W
movwf       Dest_IP0

call        inport
movf        Register_Value,W
movwf       Dest_IP1

call        inport
movf        Register_Value,W
movwf       Dest_IP2
```

```
call        inport
movf        Register_Value,W
movwf       Dest_IP3

;Se compara la dirección IP destino del paquete con la IP local
;Para confirmar que la petición ICMP está dirigida a esta tarjeta
movf        Dest_IP0,W
subwf       Host_IP0,W
btfss       STATUS,Z
goto        EXIT_SWITCH_IP

movf        Dest_IP1,W
subwf       Host_IP1,W
btfss       STATUS,Z
goto        EXIT_SWITCH_IP

movf        Dest_IP2,W
subwf       Host_IP2,W
btfss       STATUS,Z
goto        EXIT_SWITCH_IP

movf        Dest_IP3,W
subwf       Host_IP3,W
btfss       STATUS,Z
goto        EXIT_SWITCH_IP

;Se descartan las  opciones
movf        Opt_Len,W
andlw       0xFF
btfsc       STATUS,Z
goto        EXIT_OPT    ;Salir si no existen opciones

            movf        Opt_Len,W;Descartar opciones
            movwf       Counter
            LOOP_FOR4
                        call        inport
                        decfsz      Counter,F
            goto LOOP_FOR4

EXIT_OPT

;////////////////// Control de la capa de red//////////////////
;/CAPA:                 2-> INTERNET (Segun Modelo TCP/IP)                  /
;/DESCRIPCIÓN:          Direcciona el control a la capa correspondiente     /
;/                                  segun el protocolo en la cabecera IP    /
;/////////////////////////////////////////////////////////////////

;Verificar si el paquete es de tipo ICMP
;Petición de PING
movlw       ICMP
subwf       IP_Protocol,W
btfss       STATUS,Z
goto        SWITCH_TCP

            ;//////Protocolo de Internet de control de mesajeria (ICMP)//////
            ;/CAPA:                 2 -> INTERNET (Segun Modelo TCP/IP)
/
            ;/DESCRIPCIÓN:          Este protocolo maneja la respuesta al ping      /
            ;/                                          ICMP de tipo ECHO
                        /
            ;/////////////////////////////////////////////////////////////////

            ;Cabecera de protocolo ICMP (Internet Control Message Protocol)
            ;Basado en RFC 792 "Protocolo de Internet para el control de mensajes", 1981

            ;           Mensaje de Eco o de Respuesta de Eco ("Echo or Echo Reply Message")
            ;   0                   1                   2                   3
            ;   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
            ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            ;   |     Tipo      |    Código     |        Suma de Control        |
            ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            ;   |       Identificador           |     Número de Secuencia       |
            ;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            ;   |     Datos ...
            ;   +-+-+-+-+-

            ;///////////////Desencapsulación de cabecera ICMP///////////////
            ;/CAPA:                             2 -> INTERNET (Segun Modelo TCP/IP)
/
            ;/DESCRIPCIÓN:          Maneja la petición ICMP (responde a Eco - ping) /
            ;/////////////////////////////////////////////////////////////////

            ;Se obtiene el tipo
            ;8 para mensaje de eco (Echo Message)
            ;0 para mensaje de respuesta de eco (Echo Reply Message)
            movlw       NIC_DATA
            movwf       Register_Address
            call        inport
            movf        Register_Value,W
            movwf       ICMP_Type

            ;Solo se responde al tipo ECO
```

```
                ;Se verifica si la petición es de tipo ECO
                movlw   ECHO
                subwf   ICMP_Type,W
                btfss   STATUS,Z
                goto    EXIT_ICMP  ; Si no es Eco, salir

                  ;Obtenemos el código
                        movlw   NIC_DATA
                        movwf   Register_Address
                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Code

                        ; Obtenemos la suma de control de cabecera (checksum)
                        movlw   NIC_DATA
                        movwf   Register_Address
                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Chksum1

                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Chksum0

                  ;Obtenemos el Identificador
                        movlw   NIC_DATA
                        movwf   Register_Address
                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Identifier1

                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Identifier0

                    ;Recibir la secuencia
                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Sequence1

                        call    inport
                        movf    Register_Value,W
                        movwf   ICMP_Sequence0

                    ;Responde a la petición de ping
                        call    TransmitFrameICMP
                EXIT_ICMP
        goto    EXIT_SWITCH_IP

        ;Verificar si el paquete es de tipo TCP
        SWITCH_TCP
        movlw   TCP
        subwf   IP_Protocol,W
        btfss   STATUS,Z
        goto    SWITCH_UDP

                ;////////////Protocolo de control de transporte (TCP)////////////
                ;/CAPA:                          3-> TRANSPORTE (Segun Modelo TCP/IP)
/
                ;/DESCRIPCIÓN:       Protocolo de control de transporte                                      /
                ;/                                  Establece una comunicación orienteada a la
/
                ;/                                  conexión (usa sockets para la comunicación)
/
                ;////////////////////////////////////////////////////////

                ; Formato de la cabecera de TCP
                ; Segun RFC 793 "Protocolo de control de Transporte", 1981
                ;
                ;  0               1               2               3
                ;  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; |    Puerto de origen         |      Puerto de destino       |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; |                 Número de secuencia                  |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; |               Número de acuse de recibo              |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; | Posic |          |U|A|P|R|S|F|                          |
                ; | de los| Reservado |R|C|S|S|Y|I|          Ventana           |
                ; | datos |          |G|K|H|T|N|N|                          |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; |      Suma de control        |       Puntero urgente       |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; |                Opciones                |    Relleno    |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                ; |                       Datos                          |
                ; +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                ;////////////////Desencapsulación de cabecera TCP////////////////
                ;/CAPA:                          3 -> TRANSPORTE (Segun Modelo TCP/IP)                   /
                ;/DESCRIPCIÓN:       Obtiene los diferentes campos de la cabecera TCP/
```

```
;/ENTRADAS:                    NICA_DATA
                               /
;/                                                              Datos almacenados en el anillo del RTL
/

;/SALIDAS:                     Ninguna
/
;////////////////////////////////////////////////////////////
BANK1
;Se obtiene el puerto origen de la petición TCP
movlw       NIC_DATA
movwf       Register_Address1

call        inport1
movf        Register_Value1,W
movwf       tcp_source_port1
call        inport1
movf        Register_Value1,W
movwf       tcp_source_port0

;Se obtiene el puerto destino de la petición TCP
call        inport1
movf        Register_Value1,W
movwf       tcp_dest_port1
call        inport1
movf        Register_Value1,W
movwf       tcp_dest_port0

;Se obtiene el número de secuencia
call        inport1
movf        Register_Value1,W
movwf       seq0_1
call        inport1
movf        Register_Value1,W
movwf       seq0_0
call        inport1
movf        Register_Value1,W
movwf       seq1_1
call        inport1
movf        Register_Value1,W
movwf       seq1_0

;Se obtiene el número de acuse de recibo
call        inport1
movf        Register_Value1,W
movwf       ack0_1
call        inport1
movf        Register_Value1,W
movwf       ack0_0
call        inport1
movf        Register_Value1,W
movwf       ack1_1
call        inport1
movf        Register_Value1,W
movwf       ack1_0

;Se obtiene la la posición de los datos
call        inport1
movf        Register_Value1,W
andlw       0xF0                        ;Se filtra la posición de los datos
movwf       offset                      ;Se almacena la posición en "offset"
bcf                 STATUS,C            ;Se limpia el acarreo
rrf                 offset,F            ;Se obtiene la posición en bytes
bcf                 STATUS,C
rrf                 offset,F
movf        offset,W            ;Se almacena en W
movwf       tcp_options         ;Se guarda en tcp_options
movlw       0x14                        ;20 en decimal (tamaño minimo cabecera)
subwf       tcp_options,F       ;Se resta el tamaño de cabecera minima
                                                                    ;y se obtiene la cantidad de
bytes de opcion

;Se obtienen las banderas
call        inport1
movf        Register_Value1,W
andlw       0x3F                ;Obtiene unicamente las banderas
movwf       flags

;Si se ha recivido una petición de sincronización, limpiar max_seg
movlw       TCP_SYN
andwf       flags,W
btfsc       STATUS,Z
goto        EXIT_SYN
            clrf        max_seg0
            clrf        max_seg1
EXIT_SYN

;Se obtiene la ventana
call        inport1
movf        Register_Value1,W
movwf       window1
call        inport1
movf        Register_Value1,W
```

```
movwf       window0

;Se descarta la suma de control de cabecera
call        inport1
call        inport1

;Se descarta el puntero urgente
call        inport1
call        inport1

;Se manejas las opciones si existen
movlw       0XFF
andwf       tcp_options,W
btfsc       STATUS,Z
goto        EXIT_OPTION

            ;Se limpia en contador de uso general
            clrf        Counter1

            ;Se obtiene el tipo de opción
            movlw       NIC_DATA
            movwf       Register_Address1
            call        inport1
            movf        Register_Value1,W
            movwf       kind

            ;Se identifica el tipo de opción
  movlw     0X02
            subwf       kind,W
            btfss       STATUS,Z
            goto        OTHER_CASE

                        ;Se maneja el tipo de opción 2
                        ;+--------+--------+---------+--------+
                        ;|00000010|00000100|  tam max seg     |
                        ;+--------+--------+---------+--------+
                        ;  Tipo=2   Long=4

                        ;Se descarta la longitud
                        call        inport1

                ;Si esta activada la bandera SYN captura max_seg
                        movlw       TCP_SYN
                        andwf       flags,W
                        btfsc       STATUS,Z
                        goto        ELSE_OPT
                                    ;Captura el valor de max_seg0 y max_seg1
                                    call        inport1
                                    movf        Register_Value1,W
                                    movwf       max_seg1
                                    call        inport1
                                    movf        Register_Value1,W
                                    movwf       max_seg0
                                    goto        END_OPT
                        ELSE_OPT
                                    ;Se descarta max_seg
                                    call        inport1
                                    call        inport1
                        END_OPT

                        ;Sumamos 3 al contador del lazo
                        movlw       0x03
                        addwf       Counter1,F

            OTHER_CASE
                        ;Se le suma 1 al contador de lazo
                        movlw       0x01
                        addwf       Counter1,F

                        ;Verifica que todas las opciones haigan sido
                        ;descartadas del paquete entrante
                        movf        Counter1,W
                        subwf       tcp_options,W
                        btfsc       STATUS,Z
                        goto        EXIT_OPTION
                        call        inport1
            goto        OTHER_CASE
EXIT_OPTION


;////////////// Control de la capa de transporte/////////////////
;/CAPA:                 3 -> TRANSPORTE (Segun Modelo TCP/IP)                                   /
;/DESCRIPCIÓN:          Direcciona el control a la capa correspondiente         /
;/                                                       segun el protocolo en la cabecera IP
/

;////////////////////////////////////////////////////////////

;Se obtiene la longitud de los datos en la trama TCP
BANK1
movf        offset,W
BANK0
movwf       numero2_0
```

```
                clrf    numero2_1
                movf    hdr_len,W
                movwf   numero1_0
                clrf    numero1_1
                call    sumar
                movf    resultado0,W
                movwf   numero2_0
                movf    resultado1,W
                movwf   numero2_1

                movf    IP_Length0,W
                movwf   numero1_0
                movf    IP_Length1,W
                movwf   numero1_1
                call    restar
                movf    resultado0,W
                BANK1
                movwf   tcp_data_Len0
                BANK0
                movf    resultado1,W
                BANK1
                movwf   tcp_data_Len1

                ;Puerto 502 en maestro (0X01F6)
                ;Se verifica si el puerto correponde al puerto 502
                movlw   0x01
                subwf   tcp_dest_port1,W
                btfss   STATUS,Z
                goto    RQ_PORT_HTML

                movlw   0xF6                                    ;Puerto 502 en decimal
                subwf   tcp_dest_port0,W
                btfss   STATUS,Z
                goto    RQ_PORT_HTML

                        ;Establece que se ha recivido una petición MODBUS
                        BANK2
                        movlw   PORT_MODBUS
                        movwf   Socket_Number

                        ;Función para el tratamiento de los sockets
                        BANK1
                        call Socket_Mgnt
                        goto EXIT_PORT_MGNT

                ;Se identifica el puerto y se dirige el control del programa a la
                ;aplicación respectiva ( capa 7 del modelo OSI)
                ;MODBUS ocupa el puerto 503 ( en el servidor )
                ;HTML ocupa el puerto 80
                RQ_PORT_HTML
                movlw   0x00                                    ;Puerto 80 en decimal
                subwf   tcp_dest_port1,W
                btfss   STATUS,Z
                goto    RQ_PORT_CONFIG

                movlw   0x50
                subwf   tcp_dest_port0,W
                btfss   STATUS,Z
                goto    RQ_PORT_CONFIG
                        ;Por ahora no se manejan peticiones HTML
;               ;Establece que se ha recivido una petición HTML
;               BANK2
;               movlw   PORT_HTML
;               movwf   Socket_Number

;               BANK1
;               call Socket_Mgnt
;               goto EXIT_PORT_MGNT


                RQ_PORT_CONFIG
                movlw   0x04                                    ;Puerto 1026 en decimal (no asignado a un
protocolo)
                subwf   tcp_dest_port1,W
                btfss   STATUS,Z
                goto    EXIT_PORT_MGNT

                movlw   0x02
                subwf   tcp_dest_port0,W
                btfss   STATUS,Z
                goto    EXIT_PORT_MGNT

                        ;Establece que se ha recivido una petición para CONFIGURACIÓN
                        BANK2
                        movlw   PORT_CONFIG
                        movwf   Socket_Number

                        BANK1
                        call Socket_Mgnt
                        goto EXIT_PORT_MGNT

                EXIT_PORT_MGNT
                BANK0
```

```
                    goto            EXIT_SWITCH_IP
            ;Se verifica si la petición es de tipo  UDP
            SWITCH_UDP
            movlw           UDP
            subwf           IP_Protocol,W
            btfss           STATUS,Z
            goto            EXIT_SWITCH_IP
                            ;Aqui se debe colocar el codigo para manejar el
                            ;protocolo UDP
                            BANK0
            EXIT_SWITCH_IP
        EXIT_SWITCH

        ;Se revisa si se continuara con el tramamiento de una
        ;petición MODBUS fuera de la interrupción
        BANK0
        btfsc           Request_MODBUS_present,2
        goto            EXIT_IF_MODBUS_REQUEST

                ;Se establece la pagina 0 del RTL
                movlw           CR
                movwf           Register_Address
                movlw           0x22
                movwf           Register_Value
                call            outport

                ;Se establece la nueva dirección del paquete entrante
                movlw           BNDRY
                movwf           Register_Address
                movf            Next_Ptr_Pck,W
                movwf           Register_Value
                call            outport

        goto            TEST_PCK_IN_RING
        EXIT_IF_MODBUS_REQUEST

    EXIT_POLL
    return


;**********************************************************************
;******************************************************************          *
;NOMBRE:                RS232_Tx
        *               *
;DESCRIPCIÓN:           Función para transmisión de datos por el puerto              *               *
;                                       RS232
                                        *               *
;ENTRADAS:                      W                       Contiene el byte a transmitir                       *               *
;SALIDAS:               Ninguna
                        *               *
;******************************************************************          *
RS232_Tx;
                                *               *
;**********************************************************************
    BANK0                                                       ;Cambia al banco 0
    movwf       TXREG                                   ;Almacena el byte a transmitir en TXREG

    BANK1                                                       ;Cambia al banco 1
    TX_WAIT
        btfss           TXSTA,TRMT                      ;Byte transmitido?
        goto            TX_WAIT                         ;No, esperar
    BANK0                                                       ;Cambia al banco 0

    return;Fin de la función RS232_Tx



;**********************************************************************
;******************************************************************          *
;NOMBRE:                RS232_Rx
        *               *
;DESCRIPCIÓN:           Función para recepción de datos por el puerto                *               *
;                                       RS232
                                        *               *
;ENTRADAS:                      Ninguna
                                *               *
;SALIDAS:               W                       Contiene el byte recivido                           *               *
;******************************************************************          *
RS232_Rx;
                                *               *
;**********************************************************************
    movf        RCREG,W
    return;Fin de la función RS232_Tx



;**********************************************************************
;******************************************************************          *
;NOMBRE:                outport
                        *               *
```

```
;CAPA:                   1 ->ACCESO A RED (Segun Modelo TCP/IP)                                  *       *
;DESCRIPCIÓN:    Coloca un byte de datos en la dirección epecificada  *       *
;                               Utiliza variables del banco 0                                                   *
        *
;ENTRADAS:               Register_Address
        *       *
;                               Register_Value
                        *       *
;SALIDAS:        Ninguna
                        *       *
;*****************************************************************       *
outport;
                        *       *
;*****************************************************************
        BANK0
        movf    Register_Address,W      ;Se guarda la dirección a ser escrita en el registro W
        movwf   ADDR_BUS                ;Se coloca la dirección en en el bus de dirección (puerto B)
        bcf             CTRL_BUS,IOWB           ;Activar la peticion de escritura

        BANK1
        clrf    DATA_BUS                ;Se direcciona el puerto de datos como salida

        BANK0
        movf    Register_Value,W        ;Se guarda el dato a ser escrito en el registro W
        movwf   DATA_BUS                ;Se coloca el dato en el bus de datos (puerto D)

        bsf             CTRL_BUS,IOWB           ;Desactivar la peticion de escritura

        BANK1
        movlw   0xFF                            ;Se direcciona el bus de datos como entrada (puerto D)
        movwf   DATA_BUS

        BANK0
        movlw   0x01                            ;Incremena en uno el contador de bytes encapsulados
        addwf   FrameCount,F

        return; Fin de función outport



;*****************************************************************
;*****************************************************************       *
;NOMBRE:         outport1
        *       *
;CAPA:                   1 ->ACCESO A RED (Segun Modelo TCP/IP)                                  *       *
;DESCRIPCIÓN:    Coloca un byte de datos en la dirección epecificada  *       *
;                               Utiliza variables del banco 1                                                   *
        *
;ENTRADAS:               Register_Address1
        *       *
;                               Register_Value1
                        *       *
;SALIDAS:        Ninguna
                        *       *
;*****************************************************************       *
outport1;
                        *       *
;*****************************************************************
        BANK1
        movf    Register_Address1,W             ;Se guarda la dirección a ser escrita en el registro W
        BANK0
        movwf   ADDR_BUS                        ;Se coloca la dirección en en el bus de dirección (puerto B)
        bcf             CTRL_BUS,IOWB                   ;Activar la peticion de escritura

        BANK1
        clrf    DATA_BUS                        ;Se direcciona el puerto de datos como salida

        movf    Register_Value1,W               ;Se guarda el dato a ser escrito en el registro W
        BANK0
        movwf   DATA_BUS                        ;Se coloca el dato en el bus de datos (puerto D)

        bsf             CTRL_BUS,IOWB                   ;Desactivar la peticion de escritura

        movlw   0x01                                    ;Incremena en uno el contador de bytes encapsulados
        addwf   FrameCount,F

        BANK1
        movlw   0xFF
        movwf   DATA_BUS                        ;Se direcciona el bus de datos como entrada (puerto D)

        return; Fin de función outport1



;*****************************************************************
;*****************************************************************       *
;NOMBRE:         inport
                        *       *
;CAPA:                   1 ->ACCESO A RED (Segun Modelo TCP/IP)                                  *       *
;DESCRIPCIÓN:    Obtiene un byte de datos en la direccion                                 *       *
;                               especificada por Register_Address                                       *       *
```

```
;                                       Utiliza variables del banco 0                                                   *
                *
;ENTRADAS:              Register_Address        Dirección a leer en el RTL              *       *
;SALIDAS:       Register_Value                  Valor leido desde el RTL                *       *
;*****************************************************************       *
inport;
                                        *       *
;*****************************************************************
        BANK0
        movf    Register_Address,W      ;Almacena la dirección en W
        movwf   ADDR_BUS                ;Coloca la dirección en el bus de dirección (puerto D)
        bcf             CTRL_BUS,IORB                   ;Se activa la petición de lectura

        movf    DATA_BUS,W                              ;Se lee el dato desde el bus de datos (puerto D)
        movwf   Register_Value          ;Se almacena el dato en Register_Value

        bsf             CTRL_BUS,IORB                   ;Se desaciva la petición de lectura
        return


;*****************************************************************
;*****************************************************************       *
;NOMBRE:        inport1
                *       *
;CAPA:                  1 ->ACCESO A RED (Segun Modelo TCP/IP)                                  *       *
;DESCRIPCIÓN:   Obtiene un byte de datos en la direccion                 *       *
;                                       especificada por Register_Address                                       *       *
;                                       Utiliza variables del banco 1                                                   *
                *
;ENTRADAS:              Register_Address1       Dirección a leer en el RTL              *       *
;SALIDAS:       Register_Value1                 Valor leido desde el RTL                *       *
;*****************************************************************       *
inport1;
                                        *       *
;*****************************************************************
        BANK1
        movf    Register_Address1,W     ;Almacena la dirección en W
        BANK0
        movwf   ADDR_BUS                ;Coloca la dirección en el bus de dirección (puerto D)
        bcf             CTRL_BUS,IORB                   ;Se activa la petición de lectura
        movf    DATA_BUS,W                              ;Se lee el dato desde el bus de datos (puerto D)
        BANK1
        movwf   Register_Value1         ;Se almacena el dato en Register_Value
        BANK0
        bsf             CTRL_BUS,IORB                   ;Se desaciva la petición de lectura
        BANK1

        return


;*****************************************************************
;*****************************************************************       *
;NOMBRE:        delay_ms
        *       *
;CAPA:                  1 ->ACCESO A RED (Segun Modelo TCP/IP)                                  *       *
;DESCRIPCIÓN:   Genera un retardo en mili segundos hasta de 0xFF   *               *
;ENTRADAS:              Mili_Second             Retardo en milisegundos                                 *       *
;SALIDAS:       Ninguna
                *       *
;*****************************************************************       *
delay_ms;
                                        *       *
;*****************************************************************
DELAY_mSECONDS
;Realiza 5082+2=5084 ciclos de instrucción
;equivalente a 5084*200ns=1ms

        movlw   0x07                                    ;1 ciclo
        movwf   Delay_Ext               ;1 ciclo

;Realiza (1+1+720+1(NOP)+1+2)*7 = (726)*7=5082
DELAY_E
        movlw   0xF0                                    ;1 ciclo (240 en dec)
        movwf   Delay_Int               ;1 ciclo

;Realiza 240*3=720 ciclos de intrucción
DELAY_I
        decfsz  Delay_Int,F             ;1 ciclo
        goto    DELAY_I                                 ;2 ciclos
                                                                        ;1 ciclo NOP

        decfsz  Delay_Ext,F             ;1 ciclo
        goto    DELAY_E                                 ;2 ciclos
;Se logro un retardo de 1 ms

        decfsz  Mili_Second,F
        goto    DELAY_mSECONDS
        return



;*****************************************************************
```

```
;*********************************************************************	*
;NOMBRE:			remote_dma_setup									*
		*
;CAPA:					1 ->ACCESO A RED (Segun Modelo TCP/IP)			*	*
;DESCRIPCIÓN:		Configura la lectura/escritura remota al RTL8019AS	*	*
;ENTRADAS:				Read_Write	Indica la operación a realizar		*	*
;								Addr0			Dirección remota de inicio LSB	*	*
;								Addr1			Dirección remota de inicio MSB	*	*
;SALIDAS:			Ninguna
				*		*
;*********************************************************************	*
remote_dma_setup;
						*
;*************************************************************************
		;Abortar/completar DMA Remoto
		movlw		CR
		movwf		Register_Address
		movlw		0x22
		movwf		Register_Value
		call		outport

		;Establecemos numero de bytes a leer/escribir LSB
		movlw		RBCR0
		movwf		Register_Address
		movlw		0xFF
		movwf		Register_Value
		call		outport

		;Establecemos numero de bytes a leer/escribir LSB
		movlw		RBCR1
		movwf		Register_Address
		movlw		0xFF
		movwf		Register_Value
		call		outport

		;Dirección remota de inicio de lectura/escritura LSB
		movlw		RSAR0
		movwf		Register_Address
		movf		Addr0,0
		movwf		Register_Value
		call		outport

		;Dirección remota de inicio de lectura/escritura LSB
		movlw		RSAR1
		movwf		Register_Address
		movf		Addr1,0
		movwf		Register_Value
		call		outport

		;Si la operación es escritura almacena 0x12 en CR
		;Si la operación es lectura almacena 0x0A en CR
		movf		Read_Write,W
		andlw		WRITE
		btfsc		STATUS,Z
		goto		SKIP_READ

		;Escritura
		movlw		CR
		movwf		Register_Address
		movlw		0x12
		movwf		Register_Value
		call		outport

		goto		EXIT_DMA

		;Lectura
SKIP_READ
		movlw		CR
		movwf		Register_Address
		movlw		0x0A
		movwf		Register_Value
		call		outport

EXIT_DMA

		return

;*************************************************************************
;*********************************************************************	*
;NOMBRE:			TransmitFrameARP									*
		*
;CAPA:					1 ->ACCESO A RED (Segun Modelo TCP/IP)			*	*
;DESCRIPCIÓN:		Encapsula la trama de respuesta al ARP			*	*
;								Envia su MAC a quien genero la petición ARP	*	*
;ENTRADAS:				sourceParam			MAC del que emite la peticion ARP	*	*
;								Source_IP			IP del que emite la petición ARP	*	*
;SALIDAS:			Envia el paquete respuesta usando TransmitLinkMgmt	*	*
;*********************************************************************	*
TransmitFrameARP;
						*
;*************************************************************************
```

```
;///////////////Encapsulación de datos a transmitir///////////////
;/CAPA:                  1 -> ACCESO A RED                                /
;/DESCRIPCIÓN:           Realiza la encapsulación de los datos en el      /
;/                       buffer de transmisión del RTL                    /
;//////////////////////////////////////////////////////////////////
;Carga dirección MAC fuente y destino en cabecera ethernet
call        load_ethernet_header

;Tipo de paquete ARP (0x0806)
;Forma parte de la cabecera ethernet
movlw       NIC_DATA
movwf       Register_Address

movlw       ARP1
movwf       Register_Value
call        outport
movlw       ARP0
movwf       Register_Value
call        outport

;Tipo de hardware (0x0001 = ethernet)
;Comienzo de cabecera ARP
movlw       0x00
movwf       Register_Value
call        outport
movlw       0x01
movwf       Register_Value
call        outport

;Tipo de protocol (0x0800 = IP)
movlw       0x08
movwf       Register_Value
call        outport
movlw       0x00
movwf       Register_Value
call        outport

;Longitud de dirección de hardware (6 bytes)
movlw       0x06
movwf       Register_Value
call        outport

;Longitud de dirección de protocolo (4 bytes)
movlw       0x04
movwf       Register_Value
call        outport

;Opcode (ARP reply)
movlw       0x00
movwf       Register_Value
call        outport
movlw       0x02
movwf       Register_Value
call        outport

;Dirección fuente (Dirección MAC Local)
movf        Host_Addr0,W
movwf       Register_Value
call        outport

movf        Host_Addr1,W
movwf       Register_Value
call        outport

movf        Host_Addr2,W
movwf       Register_Value
call        outport

movf        Host_Addr3,W
movwf       Register_Value
call        outport

movf        Host_Addr4,W
movwf       Register_Value
call        outport

movf        Host_Addr5,W
movwf       Register_Value
call        outport

;Dirección IP local
movf        Host_IP0,W
movwf       Register_Value
call        outport

movf        Host_IP1,W
movwf       Register_Value
call        outport

movf        Host_IP2,W
movwf       Register_Value
call        outport
```

```
                movf    Host_IP3,W
                movwf   Register_Value
                call    outport

                ;Dirección destino (origen de la petición ARP)
                movf    sourceParam0,W
                movwf   Register_Value
                call    outport

                movf    sourceParam1,W
                movwf   Register_Value
                call    outport

                movf    sourceParam2,W
                movwf   Register_Value
                call    outport

                movf    sourceParam3,W
                movwf   Register_Value
                call    outport

                movf    sourceParam4,W
                movwf   Register_Value
                call    outport

                movf    sourceParam5,W
                movwf   Register_Value
                call    outport

                ;Direccion IP destino (origen de la petición ARP)
                movf    Source_IP0,W
                movwf   Register_Value
                call    outport

                movf    Source_IP1,W
                movwf   Register_Value
                call    outport

                movf    Source_IP2,W
                movwf   Register_Value
                call    outport

                movf    Source_IP3,W
                movwf   Register_Value
                call    outport

                ;Se agrega un relleno de ceros si es necesario
                call    ComputePad

                ;Paquete ensamblado y listo para ser enviado
                movlw   0x00                    ;Longitud del paquete a enviar
                movwf   Len_pack1   ;Se asigna el valor minimo de 60
                movf    FrameCount,W;60+4(CRC) = 64 (Tamaño minimo de paquete)
                movwf   Len_pack0   ;Los 4 bytes de CRC son agregados por el RTL
                call    TransmitLinkMgmt

                return


;*****************************************************************************
;*****************************************************************  *
;NOMBRE:        ComputePad
                *       *
;CAPA:                   1 ->ACCESO A RED (Segun Modelo TCP/IP)                                  *       *
;DESCRIPCIÓN:   Calcula el relleno para completar el tamaño minimo   *       *
;                               del paquete a transmitir                                                        *
                *
;ENTRADAS:              FrameCount Numero de bytes encapsulados en la trama*    *
;SALIDAS:       Ninguna
                *       *
;*****************************************************************  *
ComputePad;
                                                                        *
;*****************************************************************************
                movlw   0x3C                            ;0x3C = 60 en decimal (tamaño mínimo de trama)
                movwf   Counter
                movf    FrameCount,W            ;Contador de bytes en trama a transmitir
                subwf   Counter,F               ;Counter - FrameCount

                btfss   STATUS,C                ;Si C=1 -> Counter > FrameCount
                goto    EXIT_PAD                ;Si C=0 -> Counter < FrameCount

                ;Si Counter > FrameCount se agrega el relleno para completar el
                ;tamaño minimo de paquete
                LOOP_FOR5
                        movlw   NIC_DATA
                        movwf   Register_Address
                        movlw   0x00
                        movwf   Register_Value
                        call    outport
                        decfsz  Counter,F
```

```
        goto        LOOP_FOR5

        ;Establesco a 60 los bytes a transmitir (tamaño minimo de paquete)
        movlw       0x3C
        movwf       FrameCount

        EXIT_PAD
        return


;*******************************************************************
;****************************************************************          *
; NOMBRE:               load_ethernet_header                                                   *          *
;CAPA:                          1 ->ACCESO A RED (Segun Modelo TCP/IP)                    *          *
; DESCRIPCIÓN:          Se carga la cabecera ethernet, menos el campo       *          *
;                                               de Longitud/tipo
        *           *
;ENTRADA:  SourceParam                          Es la dirección MAC del emisor             *          *
;                               Host_Addr                       Es la dirección MAC local          *          *
;SALIDA:    Ninguna
        *           *
;****************************************************************          *
load_ethernet_header;
        *
;*******************************************************************
        ;Configuro el canal remoto DMA para escritura
        movlw       WRITE                           ;Escritura de datos
        movwf       Read_Write
        movlw       XMT_BUF_START           ;Dirección de inicio del buffer de transmision
        movwf       Addr1
        movlw       0x00
        movwf       Addr0
        call        remote_dma_setup;Configura el RTL para escritura remota

        ;Se inicializa a cero el contador de bytes de la trama a transmitir
        movlw       0x00
        movwf       FrameCount

        ;Escribir las direccion destino (origen de la petición ARP)
        ;en la cabecera ethernet
        movlw       NIC_DATA
        movwf       Register_Address
        movf        sourceParam0,W
        movwf       Register_Value
        call        outport

        movf        sourceParam1,W
        movwf       Register_Value
        call        outport

        movf        sourceParam2,W
        movwf       Register_Value
        call        outport

        movf        sourceParam3,W
        movwf       Register_Value
        call        outport

        movf        sourceParam4,W
        movwf       Register_Value
        call        outport

        movf        sourceParam5,W
        movwf       Register_Value
        call        outport

        ;Se escribe la dirección fuente (dirección MAC Local)
        movf        Host_Addr0,W
        movwf       Register_Value
        call        outport

        movf        Host_Addr1,W
        movwf       Register_Value
        call        outport

        movf        Host_Addr2,W
        movwf       Register_Value
        call        outport

        movf        Host_Addr3,W
        movwf       Register_Value
        call        outport

        movf        Host_Addr4,W
        movwf       Register_Value
        call        outport

        movf        Host_Addr5,W
        movwf       Register_Value
        call        outport

        return; Fin de función load_ethernet_header
```

```
;*********************************************************************
;*****************************************************************          *
; NOMBRE:                 TransmitLinkMgmt                                                                   *          *
;CAPA:                               1 ->ACCESO A RED (Segun Modelo TCP/IP)                         *          *
; DESCRIPCIÓN:            Envia a la capa 1 el paquete previamente          *          *
;                                    ensamblado en en buffer de transmision del RTL      *          *
;ENTRADAS:                           Len_pack0   Bytes  a transmitir LSB              *          *
;                                                Len_pack1   Bytes  a transmitir MSB           *          *
;SALIDAS:                 Ninguna
          *              *
;*****************************************************************          *
TransmitLinkMgmt;
                          *
;*********************************************************************
          ;Registro de comando (escritura a los registros del RTL)
          movlw      CR
          movwf      Register_Address
          movlw      0x22
          movwf      Register_Value
          call       outport

          ;Registro de contador de bytes a transmitir LSB
          movlw      TBCR0
          movwf      Register_Address
          movf       Len_pack0,W
          movwf      Register_Value
          call       outport

          ;Registro de contador de bytes a transmitir MSB
          movlw      TBCR1
          movwf      Register_Address
          movf       Len_pack1,W
          movwf      Register_Value
          call       outport

          ;Registro de inicio de pagina a transmitir
          movlw      TPSR
          movwf      Register_Address
          movlw      XMT_BUF_START
          movwf      Register_Value
          call       outport

          ;Registro de comando (habilita transmisión de paquete)
          movlw      CR
          movwf      Register_Address
          movlw      0x26
          movwf      Register_Value
          call       outport

          return;Fin de función TransmitLinkMgmt


;*****************************************************************
; CASO UDP (NO IMPLEMENTADO)                                                                          *
;*****************************************************************
case_UDP
          return


;*************************************************************************
;*********************************************************************          *
;NOMBRE:                  TransmitFrameICMP                                                                      *
          *
;CAPA:                               2 ->INTERNET (Segun Modelo TCP/IP)                         *          *
;DESCRIPCIÓN:             Encapsula la trama de respuesta al ICMP de Eco          *          *
;*********************************************************************          *
TransmitFrameICMP;
                                    *
;*************************************************************************

          ;///////////Encapsulacion de Datos a Transmitir//////////////////
          ;/CAPA: 3 -> INTERNET
           /
          ;/DESCRIPCIÓN:          Realiza la encapsulación de los datos en el          /
          ;/                                buffer de transmisión del RTL                                   /
          ;/////////////////////////////////////////////////////////////////
          ;Carga dirección MAC fuente y destino en cabecera ethernet
          call       load_ethernet_header

          ;Tamaño de la cabecera IP
          movlw      0x1C                          ;28 decimal
          movwf      IP_packet_length0
          clrf       IP_packet_length1

          ;Protocolo en la trama IP
          movlw      ICMP
          movwf      IP_send_protocol

          ;Se carga la cabecera IP
          call       load_IP_header
```

```
		;Se inicializa el chksum a cero;
		clrf		Chksum0
		clrf		Chksum1

		;Se carga la cabecera ICMP

		;Se carga el tipo = 0 (replica de eco)
		;0 para mensaje de respuesta de eco
		;El checksum de estos valores no es necesario ya que es cero
		movlw		0x00
		movwf		Out_Word1

		;Código ICMP
		movf		ICMP_Code,W
		movwf		Out_Word0
		call		outportw_cksm

		;Se empieza a calcular el checksum (se agrega el identificador)
		movf		ICMP_Identifier0,W
		movwf		Out_Word0
		movf		ICMP_Identifier1,W
		movwf		Out_Word1

		call		checksum

		;Se continua con el checksum (se agrega el numero de secuencia)
		movf		ICMP_Sequence0,W
		movwf		Out_Word0
		movf		ICMP_Sequence1,W
		movwf		Out_Word1

		call		checksum

		;Se obtiene el complemento de la variable Chksum
		comf		Chksum0,F
		comf		Chksum1,F
		movlw		NIC_DATA
		movwf		Register_Address

		;Se carga el checksum en la cabecera ICMP
		movf		Chksum1,W
		movwf		Register_Value
		call		outport

		movf		Chksum0,W
		movwf		Register_Value
		call		outport

		;Se carga el  identificador en la cabecra ICMP
		movf		ICMP_Identifier1,W
		movwf		Register_Value
		call		outport

		movf		ICMP_Identifier0,W
		movwf		Register_Value
		call		outport

		;Se carga el número de secuencia en la cabecera ICMP
		movf		ICMP_Sequence1,W
		movwf		Register_Value
		call		outport

		movf		ICMP_Sequence0,W
		movwf		Register_Value
		call		outport

		;Se agrega un relleno de ceros si es necesario
		call		ComputePad

		;Se transmite la trama en el buffer del RTL
		;Se transmite el tamaño minimo de trama (60 bytes)
		movlw		0x00
		movwf		Len_pack1
		movlw		0x3C			; 60 en decimal
		movwf		Len_pack0
		call		TransmitLinkMgmt

		return

;*********************************************************************
;****************************************************************		*
;NOMBRE:		load_IP_header								*
		*
;CAPA:				3 -> INTERNET
		*	*
;DESCRIPCIÓN:		Carga la cabecera IP del paquete a transmitir		*		*
;****************************************************************		*
load_IP_header;
				*
;*********************************************************************
```

```
;                Formato de cabecera IP (Segun RFC 791 "Protocolo Internet", 1981)
;
;   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;   |Version|  IHL  |Type of Service|          Total Length         |
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;   |        Identification         |Flags|      Fragment Offset    |
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;   | Time to Live |    Protocol    |         Header Checksum       |
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;   |                    Source Address                             |
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;   |                  Destination Address                          |
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;   |                    Options                    |    Padding    |
;   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

;Se carga el tipo de protocolo correspondiente a la cabecera Ethernet
;Este campo aún no forma parte de la cabecera IP
movlw       NIC_DATA
movwf       Register_Address

movlw       IP1
movwf       Register_Value
call        outport
movlw       IP0
movwf       Register_Value
call        outport

;Se limpia la suma de control de cabecera
clrf        Chksum0
clrf        Chksum1

;Se carga en la cabecera el tipo de servicio
movlw       0x00
movwf       Out_Word0
;Se carga en la cabecera la versión y el tamaño de cabecera
movlw       0x45
movwf       Out_Word1
call        outportw_cksm
;Se calcula la suma de control de cabecera de estos campos
call        checksum

;Se carga en la cabecera la longitud total del datagrama
;Incluye cabecera y datos
movf        IP_packet_length0,W
movwf       Out_Word0
movf        IP_packet_length1,W
movwf       Out_Word1
call        outportw_cksm
;Se calcula la suma de control de cabecera de estos campos
call        checksum

;Se carga el identificador (el mismo que la trama de petición)
movf        Identification0,W
movwf       Out_Word0
movf        Identification1,W
movwf       Out_Word1
call        outportw_cksm
;Se calcula la suma de control de cabecera de estos campos
call        checksum

;Se cargan las banderas y la posición
movlw       0x00
movwf       Out_Word0
movlw       0x40                    ;No se acepta fragmentación
movwf       Out_Word1
call        outportw_cksm
;Se calcula la suma de control de cabecera de estos campos
call        checksum

;Se carga en la cabecera el protocolo del siguiente nivel
movf        IP_send_protocol,W
movwf       Out_Word0
;Se carga en la cabecera el tiempo de vida
movlw       0xFF
movwf       Out_Word1
call        outportw_cksm
;Se calcula la suma de control de cabecera de estos campos
call        checksum

;Se calcula la suman de control de cabecera para la IP local
;IP0 e IP1
movf        Host_IP1,W
movwf       Out_Word0
movf        Host_IP0,W
movwf       Out_Word1
call        checksum

;Se calcula la suman de control de cabecera para la IP local
;IP2 e IP3
movf        Host_IP3,W
```

```
        movwf   Out_Word0
        movf    Host_IP2,W
        movwf   Out_Word1
        call    checksum

        ;Se calcula la suman de control de cabecera para la IP origen
        ;IP0 e IP1
        movf    Source_IP1,W
        movwf   Out_Word0
        movf    Source_IP0,W
        movwf   Out_Word1
        call    checksum

        ;Se calcula la suman de control de cabecera para la IP origen
        ;IP2 e IP3
        movf    Source_IP3,W
        movwf   Out_Word0
        movf    Source_IP2,W
        movwf   Out_Word1
        call    checksum

        ;Se obtiene el complemento de la suma de todos 16 bits contenidos
        ;en la cabecera IP
        comf    Chksum0,F
        comf    Chksum1,F

        ;Se carga la suma de control en la cabecera IP
        movf    Chksum1,W
        movwf   Register_Value
        call    outport
        movf    Chksum0,W
        movwf   Register_Value
        call    outport

        ;Se carga la IP local en la cabecera IP
        ;IP0 e IP1
        movlw   NIC_DATA
        movwf   Register_Address
        movf    Host_IP0,W
        movwf   Register_Value
        call    outport
        movf    Host_IP1,W
        movwf   Register_Value
        call    outport

        ;Se carga la IP local en la cabecera IP
        ;IP2 e IP3
        movf    Host_IP2,W
        movwf   Register_Value
        call    outport
        movf    Host_IP3,W
        movwf   Register_Value
        call    outport

        ;Se carga la IP origen en la cabecera IP
        ;IP0 e IP1
        movlw   NIC_DATA
        movwf   Register_Address
        movf    Source_IP0,W
        movwf   Register_Value
        call    outport
        movf    Source_IP1,W
        movwf   Register_Value
        call    outport

        ;Se carga la IP origen en la cabecera IP
        ;IP2 e IP3
        movf    Source_IP2,W
        movwf   Register_Value
        call    outport
        movf    Source_IP3,W
        movwf   Register_Value
        call    outport

        return; Fin de la función load_IP_header

;*********************************************************************
;*****************************************************************              *
;NOMBRE:          load_tcp_header                                                              *
        *
;CAPA:                        3 -> TRANSPORTE
        *         *
;DESCRIPCIÓN:     Carga la cabecera TCP en el buffer de anillo del*      *
;                                             RTL
                                              *          *
;*****************************************************************              *
load_tcp_header;
                  *
;*********************************************************************

        ;Inicializo el checksum al valor inicial
```

```
BANK1
movf        tcp_Chksum1,W
BANK0
movwf       Chksum1
BANK1
movf        tcp_Chksum0,W
BANK0
movwf       Chksum0

;Se le suma la IP local al checksum
movf        Host_IP1,W
movwf       Out_Word0
movf        Host_IP0,W
movwf       Out_Word1
call        checksum

movf        Host_IP3,W
movwf       Out_Word0
movf        Host_IP2,W
movwf       Out_Word1
call        checksum

;Se le suma la IP destino al checksum
movf        Source_IP1,W
movwf       Out_Word0
movf        Source_IP0,W
movwf       Out_Word1
call        checksum

movf        Source_IP3,W
movwf       Out_Word0
movf        Source_IP2,W
movwf       Out_Word1
call        checksum

;Se le suma el protocolo al checksum
movlw       TCP
movwf       Out_Word0
movlw       0x00
movwf       Out_Word1
call        checksum

;Se le suma la  longitud de cabecera al checksum
BANK1
movf        tcp_length0,W
BANK0
movwf       Out_Word0
BANK1
movf        tcp_length1,W
BANK0
movwf       Out_Word1
call        checksum

;Se le suma el pueto origen al checksum
;Se carga el puerto destino al buffer del RTL
BANK1
movlw       NIC_DATA
movwf       Register_Address1
movf        tcp_dest_port0,W
BANK0
movwf       Out_Word0
BANK1
movf        tcp_dest_port1,W
BANK0
movwf       Out_Word1
call        checksum
BANK1

call        outportw_cksm1

;Se le suma el puerto destino al checksum
;Se carga el puerto destino al buffer del RTL
movf        tcp_source_port0,W
BANK0
movwf       Out_Word0
BANK1
movf        tcp_source_port1,W
BANK0
movwf       Out_Word1
call        checksum
BANK1

call        outportw_cksm1

;Se le suma el numero de secuencia al checksum
;Se carga el numero de secuencia al buffer del RTL
movf        xm_seq0_0,W
BANK0
movwf       Out_Word0
BANK1
movf        xm_seq0_1,W
BANK0
```

```
movwf       Out_Word1
call        checksum
BANK1

call        outportw_cksm1

movf        xm_seq1_0,W
BANK0
movwf       Out_Word0
BANK1
movf        xm_seq1_1,W
BANK0
movwf       Out_Word1
call        checksum
BANK1

call        outportw_cksm1

;Se le suma el numero de acuse al checksum
;Se carga el numero de acuse al buffer del RTL
movf        xm_ack0_0,W
BANK0
movwf       Out_Word0
BANK1
movf        xm_ack0_1,W
BANK0
movwf       Out_Word1
call        checksum
BANK1

call        outportw_cksm1

movf        xm_ack1_0,W
BANK0
movwf       Out_Word0
BANK1

movf        xm_ack1_1,W
BANK0
movwf       Out_Word1
call        checksum
BANK1
call        outportw_cksm1

;Se le suman la banderas del paquete TCP al checksum
;Se cargan las banderas en la cabecera TCP
movf        data_flags0,W
BANK0
movwf       Out_Word0
BANK1
movf        data_flags1,W
BANK0
movwf       Out_Word1
call        checksum
BANK1
call        outportw_cksm1

;Se le suma el tamño de la ventana al checksum
;Se carga el tamaño de ventana a la cabecera TCP
movlw       0xB4
BANK0
movwf       Out_Word0
BANK1
movlw       0x05
BANK0
movwf       Out_Word1
call        checksum
BANK1

call        outportw_cksm1

;Se carga el checksum a la cabecera TCP
BANK0
;Se calcula el complemento
comf        Chksum0,F
comf        Chksum1,F
;Se carga el valor de la suma de control (checksum)
movf        Chksum0,W
movwf       Out_Word0
movf        Chksum1,W
movwf       Out_Word1
BANK1
call        outportw_cksm1

;Se carga el puntero urgente a la cabecera TCP
movlw       0x00
BANK0
movwf       Out_Word0
BANK1
movlw       0x00
BANK0
movwf       Out_Word1
```

```
		BANK1
		call		outportw_cksm1

		return


;*******************************************************************
;***************************************************************		*
;NOMBRE:			Socket_Mgnt									*
		*
;CAPA:					3 -> TRANSPORTE
		*		*
;DESCRIPCIÓN:		Establece, realiza transacciones y finaliza la		*		*
;								conexión del socket
		*		*
;***************************************************************		*
Socket_Mgnt;
						*
;*******************************************************************

		;Se verifica el estado del socket
		BANK1
		;El primer estado es LISTEN e indica que el socket no ha sido
		;inicializado
		movlw	LISTEN
		subwf	html_socket,W
		btfss	STATUS,Z
		goto	CASE_SYNRCVD			; Si no esta en LISTEN verificar SYNRCVD

			;El estado del socket es LISTEN, por lo cual se procede a verificar
			;si es una petición para establecer el socket

			;Si aun se esta procesando una petición anterior, no se puede iniciar
			;una nueva transacción
			btfsc		Request_MODBUS_present,0
			goto		EXIT_SOCKET ;Aun se esta procesando la petición anterior

			;Se verifica si la bandera TCP_SYN esta activada en el paquete entrante
			movlw		TCP_SYN
			andwf		flags,W
			btfsc		STATUS,Z
			goto		EXIT_SOCKET ;Si no esta activada, no es una peticion para
										;inicializar el socket

			;Se verifica si la bandera TCP_ACK esta activada en el paquete entrante
			;Si esta activada no debe aceptarse, ya que el socket no ha sido establecido
			movlw		TCP_ACK
			andwf		flags,W
			btfss		STATUS,Z
			goto		EXIT_SOCKET ;TCP_ACK activada y el paquete no debe ser tratado


			;Se verifica si la bandera TCP_RST esta activada en el paquete entrante
			;Si esta activada no debe aceptarse, ya que el socket no ha sido establecido
			movlw		TCP_RST
			andwf		flags,W
			btfss		STATUS,Z
			goto		EXIT_SOCKET ; TCP_RST activada y el paquete no debe ser tratado

			;Se verifica si la bandera TCP_RST esta activada en el paquete entrante
			;Si esta activada no debe aceptarse, ya que el socket no ha sido establecido
			movlw		TCP_FIN
			andwf		flags,W
			btfss		STATUS,Z
			goto		EXIT_SOCKET ;TCP_FIN activada y el paquete no debe ser tratado


			;Se han verificado todas las banderas y el paquete es una petición para
			;iniciar un socket, ser respondera a dicha petición con un [SYN,ACK]
			bsf				PCLATH,4
			bcf				PCLATH,3			;PAGINA 2 DE PROGRAMA
			call		tcp_listen
			bcf				PCLATH,4
			bcf				PCLATH,3			;PAGINA 0 DE PROGRAMA

			goto		EXIT_SOCKET


		;Se verifica si el  estado del socket es SYNRCVD ( En proceso de establecimiento )
		CASE_SYNRCVD
		movlw	SYN_RCVD
		subwf	html_socket,W
		btfss	STATUS,Z
		goto	CASE_ESTAB ;Probar el otro estado del socket

			;El estado del socket es SYNRCVD y espera una respuesta de tipo ACK para
			;finalizar con el establecimiento del socket

			;Verifica si el paquete proviene del mismo puerto
			movf		port1,W
			subwf		tcp_source_port1,W
```

```
                btfss       STATUS,Z
                goto        EXIT_SOCKET

                movf        port0,W
                subwf       tcp_source_port0,W
                btfss       STATUS,Z
                goto        EXIT_SOCKET

                ;Verificar si el paquete venga desde la misma direccion IP
                movf        ip0,W
                BANK0
                subwf       Source_IP0,W
                btfss       STATUS,Z
                goto        EXIT_SOCKET

                BANK1
                movf        ip1,W
                BANK0
                subwf       Source_IP1,W
                btfss       STATUS,Z
                goto        EXIT_SOCKET

                BANK1
                movf        ip2,W
                BANK0
                subwf       Source_IP2,W
                btfss       STATUS,Z
                goto        EXIT_SOCKET

                BANK1
                movf        ip3,W
                BANK0
                subwf       Source_IP3,W
                btfss       STATUS,Z
                goto        EXIT_SOCKET
                BANK1

                ;Se esta esperando un ACK sin un RST
                ;Si la bandera RST esta activada no se atiende este paquete
                movf        TCP_RST,W
                subwf       flags,W
                btfsc       STATUS,Z
                goto        STAR_RST

                ;Se espera que la bandera ACK este activada para continuar
                ;con el establecimiento del socket
                movlw       TCP_ACK
                subwf       flags,W
                btfss       STATUS,Z
                goto        EXIT_SOCKET

                ;Se esta esperando un ACK sin un SYN
                ;Si la bandera SYN esta activada no se atiende este paquete
                movf        TCP_SYN,W
                subwf       flags,W
                btfsc       STATUS,Z
                goto        EXIT_SOCKET

                ;Se esta esperando un ACK sin un FIN
                ;Si la bandera FIN esta activada no se atiende este paquete
                movf        TCP_FIN,W
                subwf       flags,W
                btfsc       STATUS,Z
                goto        EXIT_SOCKET

                ;Se ha verificado que este paquete es un ACK al SYN enviado
                ;y por lo tanto se configurara el establecimiento del socket

                bcf                     PCLATH,4
                bsf                     PCLATH,3            ;PAGINA 1 DE PROGRAMA
                call        tcp_syn_rcvd
                bcf                     PCLATH,4
                bcf                     PCLATH,3            ;PAGINA 0 DE PROGRAMA

                goto        EXIT_SOCKET

;Se verifica si el estado del socket es ESTAB ( Socket establecido )
CASE_ESTAB
movlw       ESTAB
subwf       html_socket,W
btfss       STATUS,Z
goto        CASE_FIN ;Verificar si el estado es FIN

                ;//////////////////////Administrador TCP//////////////////////
                ;/CAPA:                 3 -> TRANSPORTE
                /
                ;/DESCRIPCIÓN:          Dentro del socket  administra la aceptación de          /
                ;/                                          las peticiones y  si es necesario envia a la            /
                ;/                                          capa superior el control del paquete                    /
                ;//////////////////////////////////////////////////////////

                ;Se verifica que el numero de secuencia del paquete
```

```
;entrante coincida con el ACK del ultimo paquete enviado
BANK1
movf        seq0_1,W
subwf       xm_ack0_1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

movf        seq0_0,W
subwf       xm_ack0_0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

movf        seq1_1,W
subwf       xm_ack1_1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

movf        seq1_0,W
subwf       xm_ack1_0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Se verifica que el socket se este conectando
;al mismo puerto
BANK1
movf        port1,W
subwf       tcp_source_port1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

movf        port0,W
subwf       tcp_source_port0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Se verifica que provenga de la misma dirección IP
BANK1
movf        ip0,W
BANK0
subwf       Source_IP0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

BANK1
movf        ip1,W
BANK0
subwf       Source_IP1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

BANK1
movf        ip2,W
BANK0
subwf       Source_IP2,W
btfss       STATUS,Z
goto        EXIT_SOCKET

BANK1
movf        ip3,W
BANK0
subwf       Source_IP3,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;**********Se verifican las banderas del paquete **********

;Nos aseguramos que un RST no fue enviado
BANK1
movlw       TCP_RST
andwf       flags,W
btfss       STATUS,Z
goto        STAR_RST

;Un FIN puede ser recivido, pero indica la finalización
;del socket actual
movlw       TCP_FIN
andwf       flags,W
btfss       STATUS,Z
goto        STAR_FIN ;Proceder con la finalización del socket

;En ESTAB los datos vienen con esta bandra activada
movlw       TCP_ACK
andwf       flags,W
btfsc       STATUS,Z
goto        EXIT_SOCKET             ; Si no esta activada, salir

;No se deben recivir paquetes con la bandera SYN (solo maneja
;un socket a la vez)
movlw       TCP_SYN
andwf       flags,W
btfss       STATUS,Z
goto        EXIT_SOCKET
```

```
;Verifica que el paquete contega datos
;Si no tiene, es un ACK a un paquete enviado anteriormente
movlw   0xFF
andwf   tcp_data_Len0,W
btfss   STATUS,Z
goto    GO_TCP_ESTAB

movlw   0xFF
andwf   tcp_data_Len1,W
btfss   STATUS,Z
goto    GO_TCP_ESTAB

goto    EXIT_SOCKET ;El paquete es un ACK

GO_TCP_ESTAB
        ;Si existe una petición en proceso, no se debe aceptar
        ;esta nueva petición
        BANK0
        btfsc   Request_MODBUS_present,0
        goto    EXIT_SOCKET ;Aun se esta procesando la petición anterior

        ;Se verifica el tipo de petición entranate
        BANK2
        movlw   PORT_MODBUS
        subwf   Socket_Number,W
        btfss   STATUS,Z
        goto    TEST_REQUEST_HTML

                ;Se aceptan las peticiones MODBUS
                BANK0
                ;El paquete posee datos y deben ser tratados
                ;Se activa la bandera por PETICIÓN MODBUS
                ;Se establece la existencia de una petición MODBUS
                bsf             Request_MODBUS_present,0
                ;Se establece que en la corriente interrupción existio
                ;una petición MODBUS
                bsf             Request_MODBUS_present,2

                ;********** SE DESABILITA LA INTERRUPCION DEL TMR0 **********
                ;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE EHTERNET
                BANK1
                bcf             INTCON,T0IE
                ;********** SE DESABILITAN LAS INTERRUPCIONES VIA SCI **********
                bcf             PIE1,RCIE

                goto EXIT_SOCKET

        ;Se aceptan las peticiones HTML
        TEST_REQUEST_HTML
        movlw   PORT_HTML
        subwf   Socket_Number,W
        btfss   STATUS,Z
        goto    TEST_REQUEST_CONFIG

                ;Aqui se coloca el codigo para manejar las peticiones HTML

                goto EXIT_SOCKET

        TEST_REQUEST_CONFIG
        movlw   PORT_CONFIG
        subwf   Socket_Number,W
        btfss   STATUS,Z
        goto    EXIT_SOCKET

                ;Se verifica si se desea modificar la IP del disposivo
                BANK1
                movf    tcp_data_Len1,W
                sublw   0x00
                btfss   STATUS,Z
                goto    EXIT_SOCKET

                movf    tcp_data_Len0,W
                sublw   0x0A
                btfss   STATUS,Z
                goto    EXIT_SOCKET

                        ;Se activa la bandera que indica la presencia de una nueva
                        ;IP en la EEPROM del PIC
                        BANK0
                        bsf             Request_MODBUS_present,3

                        bsf             PCLATH,4
                        bsf             PCLATH,3        ;PAGINA 3 DE PROGRAMA
                        call    Change_IP_device
                        bcf             PCLATH,4
                        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA

                        goto EXIT_SOCKET


;Se ha recivido una petición de cierre obligatorio del socket
```

```
;Esto es debido a un problema en las transacciones realizadas
STAR_RST
        BANK1
        movlw   LISTEN
        movwf   html_socket
        goto EXIT_SOCKET

;Se ha recivido una petición para finalización del socket
;Se inicial el proceso para cerrarlo

STAR_FIN

;Se define un nuevo numero de indentificación
BANK0
movf    Identification0,W
movwf   numero1_0
movf    Identification1,W
movwf   numero1_1
movlw   0x01
movwf   numero2_0
movlw   0x00
movwf   numero2_1

call    sumar

movf    resultado0,W
movwf   Identification0
movf    resultado1,W
movwf   Identification1


;Se activan las banderas ACK y FIN como respueta a la
;petición de finalizar el socket
BANK1
movlw   0x00
movwf   data_flags0
bsf             data_flags0,4           ;Se habilita la bandera TCP_ACK
bsf             data_flags0,0           ;Se habilita la bandera TCP_FIN

;Se envia un paquete de FIN, ACK para hacer constar al que
;origino la petición que nos llego el paquete
bcf             PCLATH,4
bsf             PCLATH,3                ;PAGINA 1 DE PROGRAMA
call    socket_control
bcf             PCLATH,4
bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA

;Se verifica si existe una petición en proceso
btfss   Request_MODBUS_present,0
goto    NO_BCK_SOCKET
        ;Se deben reestablecer los valores adecuados para el socket
        ;ya que han sido cambiados por realizar una transacción
        bsf             PCLATH,4
        bsf             PCLATH,3                ;PAGINA 3 DE PROGRAMA
        call    save_value_vars
        bcf             PCLATH,4
        bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA
NO_BCK_SOCKET

;Se cambia es estado del  socket a FIN
;Se espera  un paquete ACK para continuar con la finalización
;del socket
BANK1
movlw   FIN
movwf   html_socket
goto    EXIT_SOCKET

;Se ha inicializado el proceso para finalización del socket y se
;ha recivido un ACK. Se respondera con otro ACK para finalizar la comunicaicón

CASE_FIN
movlw   FIN
subwf   html_socket,W
btfss   STATUS,Z
goto    EXIT_SOCKET

        ;Se verifica que el numero de secuencia del paquete
        ;entrante coincida con el ACK del ultimo paquete enviado
        movf    seq0_1,W
        subwf   xm_ack0_1,W
        btfss   STATUS,Z
        goto    EXIT_SOCKET

        movf    seq0_0,W
        subwf   xm_ack0_0,W
        btfss   STATUS,Z
        goto    EXIT_SOCKET

        movf    seq1_1,W
        subwf   xm_ack1_1,W
        btfss   STATUS,Z
        goto    EXIT_SOCKET
```

```
movf        seq1_0,W
subwf       xm_ack1_0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Se verifica que el socket provenga del mismo puerto
movf        port1,W
subwf       tcp_source_port1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

movf        port0,W
subwf       tcp_source_port0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Se verifica que provenga de la misma IP
movf        ip0,W
BANK0
subwf       Source_IP0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

BANK1
movf        ip1,W
BANK0
subwf       Source_IP1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

BANK1
movf        ip2,W
BANK0
subwf       Source_IP2,W
btfss       STATUS,Z
goto        EXIT_SOCKET

BANK1
movf        ip3,W
BANK0
subwf       Source_IP3,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Nos aseguramos que un RST no fue enviado
BANK1
movlw       TCP_RST
andwf       flags,W
btfss       STATUS,Z
goto        STAR_RST

;Se esta esperando por un ACK y terminar con el socket
movlw       TCP_ACK
andwf       flags,W
btfsc       STATUS,Z
goto        EXIT_SOCKET             ; Si no esta activada, salir

;No se deben recivir paquetes con la bandera SYN
;Solo se maneja un socket a la vez
movlw       TCP_SYN
andwf       flags,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Un FIN no puede ser recivido
movlw       TCP_FIN
andwf       flags,W
btfss       STATUS,Z
goto        EXIT_SOCKET ;Proceder con la finalización del socket

;Verifica que el paquete contega datos
;Si no tiene, es un ACK
movlw       0xFF
andwf       tcp_data_Len0,W
btfss       STATUS,Z
goto        EXIT_SOCKET

movlw       0xFF
andwf       tcp_data_Len1,W
btfss       STATUS,Z
goto        EXIT_SOCKET

;Se contesta con un ACK para finalizar el socket
movlw       0x00
movwf       data_flags0
bsf                 data_flags0,4           ;TCP_ACK

bcf                 PCLATH,4
bsf                 PCLATH,3                ;PAGINA 1 DE PROGRAMA
call        socket_control
bcf                 PCLATH,4
```

```
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA

                movlw   LISTEN
                movwf   html_socket

                ;Ahora se verifica si existio una petición para el cambio de IP o MAC

                ;Solo se aceptan cambios de IP cuando no existe una transacción
                ;MODBUS en proceso
                btfsc   Request_MODBUS_present,0
                goto    EXIT_SOCKET

                btfsc   Request_MODBUS_present,2
                goto    EXIT_SOCKET

                ;Verifica si ha existido una petición para cambio de IP
                btfss           Request_MODBUS_present,3
                goto            TEST_RQ_NEW_MAC
                        ;Existe una nueva IP en la EEPROM del PIC y debe actualizarse
                        bsf             PCLATH,4
                        bsf             PCLATH,3        ;PAGINA 3 DE PROGRAMA
                        call    LOAD_IP_FROM_EEPROM
                        bcf             PCLATH,4
                        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                        ;Se limpia la bandera por recepción de nueva IP
                        BANK0
                        bcf             Request_MODBUS_present,3

                TEST_RQ_NEW_MAC
                ;Verifica si ha existido una petición para cambio de MAC
                btfss           Request_MODBUS_present,4
                goto            EXIT_SOCKET
                        ;Existe una nueva MAC en la EEPROM del PIC y debe actualizarse
                        bsf             PCLATH,4
                        bsf             PCLATH,3        ;PAGINA 3 DE PROGRAMA
                        call    LOAD_MAC_FROM_EEPROM
                        bcf             PCLATH,4
                        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                        ;Se limpia la bandera por recepción de nueva MAC
                        BANK0
                        bcf             Request_MODBUS_present,4

        EXIT_SOCKET

        BANK1
        return


;*****************************************************************
;***************************************************************  *
;NOMBRE:         sumar
        *       *
;DESCRIPCIÓN:    Suma dos palabras                                                                       *       *
;ENTRADAS:               numero1_0, numero1_1, numero2_0, numero2_1      *       *
;SALIDAS:        resultado1, resultado2                                                                  *       *
;***************************************************************  *
;
sumar;
                                                *
;*****************************************************************
        ;Se inicializa variable para conocer si se ha desbordado la suma
        clrf    CarryOut ;Un valor de uno indica un desbordamiento

        movf    numero1_0,W
        addwf   numero2_0,W
        movwf   resultado0
        btfss   STATUS,C
        goto    continuarSumar
        movlw   0x01
        addwf   numero1_1,F
        btfss   STATUS,C
        goto    continuarSumar
        movlw   0x01
        movwf   CarryOut        ;Se dio un desbordamiento


continuarSumar
        movf    numero1_1,W
        addwf   numero2_1,W
        movwf   resultado1
        btfss   STATUS,C
        goto    exitsum
        movlw   0x01
        movwf   CarryOut        ;Se dio un desbordamiento

exitsum
        return


;*****************************************************************
;***************************************************************  *
;
```

```
;NOMBRE:                 outportw_cksm                                                                   *
                *
;CAPA:                           1 ->ACCESO A RED (Segun Modelo TCP/IP)                  *       *
;DESCRIPCIÓN:            Carga una palabra en el buffer del RTL          *       *
;                                                Se utiliza en el banco cero                             *       *
;ENTRADAS:                       Out_Word0 y Out_Word1                                                   *       *
;SALIDAS:                ninguna
                *       *
;*****************************************************************       *
'
outportw_cksm;
                                        *
;*****************************************************************
'
                movlw   NIC_DATA
                movwf   Register_Address
                movf    Out_Word1,w
                movwf   Register_Value
                call    outport

                movf    Out_Word0,w
                movwf   Register_Value
                call    outport

                return


;*****************************************************************
'*****************************************************************       *
;NOMBRE:                 outportw_cksm1                                                                  *
                *
;CAPA:                           1 ->ACCESO A RED (Segun Modelo TCP/IP)                  *       *
;DESCRIPCIÓN:            Carga una palabra en el buffer del RTL          *       *
;                                                Se utiliza en el banco 1                                *       *
;ENTRADAS:                       Out_Word0 y Out_Word1                                                   *       *
;SALIDAS:                ninguna
                *       *
;*****************************************************************       *
'
outportw_cksm1;
                                        *
;*****************************************************************
'
                BANK1
                movlw   NIC_DATA
                movwf   Register_Address1

                BANK0
                movf    Out_Word1,W

                BANK1
                movwf   Register_Value1
                call    outport1

                BANK0
                movf    Out_Word0,W
                BANK1
                movwf   Register_Value1
                call    outport1

                return


;*****************************************************************
'*****************************************************************       *
;NOMBRE:                 checksum                                                                        *
                *
;CAPA:                           1, 2 Y 3 (Segun Modelo TCP/IP)                                          *       *
;DESCRIPCIÓN:            Realiza la sumatoria de cada palabra del paquete*       *
;ENTRADAS:                       Out_Word0 y Out_Word1                                                   *       *
;SALIDAS:                Chksum0, Chksum1                                                                *       *
;*****************************************************************       *
'
checksum;
                                        *
;*****************************************************************
'
                ;Suma los datos con el checksum actual
                BANK0
                movf    Chksum0,W
                movwf   numero1_0
                movf    Chksum1,W
                movwf   numero1_1
                movf    Out_Word0,W
                movwf   numero2_0
                movf    Out_Word1,W
                movwf   numero2_1

                call    sumar

                movf    resultado0,W
                movwf   Chksum0
                movf    resultado1,W
                movwf   Chksum1
```

```
                ;Si el nuevo valor del checksum en menor que el checksum
                ;anterior, se incrementa el nuevo checksum en uno
                movf    CarryOut,W
                sublw   0x01
                btfss   STATUS,Z
                goto    EXIT_OUTPORTW

                ;Incrementa el checksum nuevo en uno
                movf    Chksum0,W
                movwf   numero1_0
                movf    Chksum1,W
                movwf   numero1_1
                movlw   0x01
                movwf   numero2_0
                movlw   0x00
                movwf   numero2_1

                call    sumar

                movf    resultado0,W
                movwf   Chksum0
                movf    resultado1,W
                movwf   Chksum1

EXIT_OUTPORTW
                return


;*********************************************************************
;*********************************************************************        *
;NOMBRE:                 restar
                *       *
;DESCRIPCIÓN:            Realiza la resta de dos palabras                                        *       *
;ENTRADAS:                       numero1_0, numero1_1, numero2_0, numero2_1                      *       *
;SALIDAS:                resultado0, resultado1                                                                  *       *
;*********************************************************************        *
restar;
                                                        *
;*********************************************************************
                ;Se inicializa variable para conocer si se ha desbordado la resta
                BANK0
                clrf    CarryOut ;Un valor de uno indica un desbordamiento

                movf    numero2_0,W
                subwf   numero1_0,W
                movwf   resultado0
                btfsc   STATUS,C                ;Si existe un desbordamiento se suma 1
                goto    continuarRestar         ;al MSB del numero 2
                movlw   0x01                            ;Se le suma 1 al MSB del numero2
                addwf   numero2_1,F
                btfsc   STATUS,C
                goto    continuarRestar
                movlw   0x01
                movwf   CarryOut                ;ha existido un desbordamiento


continuarRestar
                movf    numero2_1,W
                subwf   numero1_1,W
                movwf   resultado1
                btfsc   STATUS,C
                goto    exitsub
                movlw   0x01
                movwf   CarryOut                ;ha existido un desbordamiento

                exitsub
                return



;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;$   CAMBIO A LA PAGINA 1 DEL PIC                $$
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                ORG     0X800

;*********************************************************************
;*********************************************************************        *
;NOMBRE:                 tcp_syn_rcvd                                                                                                    *       *
;CAPA:                           3 -> TRANSPORTE
                *       *
;DESCRIPCIÓN:            Indica la recepción de un SYN, pero que aun             *       *
;                                                       necesita un ACK para finalizar el socket        *       *
;*********************************************************************        *
tcp_syn_rcvd;
                                        *
;*********************************************************************
                ;Obtiene la longitud de los datos
                BANK1
                movf    offset,W
                BANK0
                movwf   numero2_0
```

```
clrf        numero2_1
movf        hdr_len,W
movwf       numero1_0
clrf        numero1_1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        sumar
bcf                 PCLATH,4
bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

movf        resultado0,W
movwf       numero2_0
movf        resultado1,W
movwf       numero2_1

movf        IP_Length0,W
movwf       numero1_0
movf        IP_Length1,W
movwf       numero1_1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        restar
bcf                 PCLATH,4
bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

movf        resultado0,W
BANK1
movwf       tcp_data_Len0
BANK0
movf        resultado1,W
BANK1
movwf       tcp_data_Len1


;Actualiza el valor del numero de acuse sumandole
;los datos recividos
movf        tcp_data_Len0,W
BANK0
movwf       numero1_0
BANK1
movf        tcp_data_Len1,W
BANK0
movwf       numero1_1
BANK1
movf        xm_ack1_0,W
BANK0
movwf       numero2_0
BANK1
movf        xm_ack1_1,W
BANK0
movwf       numero2_1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        sumar
bcf                 PCLATH,4
bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

movf        resultado0,W
BANK1
movwf       xm_ack1_0
BANK0
movf        resultado1,W
BANK1
movwf       xm_ack1_1

;Se le suma uno a la segunda palabra del ACK si existio
;un desborde en la suma de la palabra LSB
BANK0
movf        CarryOut,W
sublw       0x01
btfss       STATUS,Z
goto        NOT_CARRY_OUT2      ;numero1>numero2 (no hay acarreo)

            ;Existio un desborde, por lo cual se sumara a la palabra ACK MSB
            BANK1
            movf        xm_ack0_1,W
            BANK0
            movwf       numero1_1
            BANK1
            movf        xm_ack0_0,W
            BANK0
            movwf       numero1_0
            movlw       0x00
            movwf       numero2_1
            movlw       0x01
            movwf       numero2_0

            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
```

```
		call		sumar
		bcf				PCLATH,4
		bsf				PCLATH,3			;PAGINA 1 DE PROGRAMA

		movf		resultado0,W
		BANK1
		movwf		xm_ack0_0
		BANK0
		movf		resultado1,W
		BANK1
		movwf		xm_ack1_1

	NOT_CARRY_OUT2

	;El socket ya se establecio y se cambia es estado de la
	;variable socket a ESTAB, estando ahora en espera de paquetes
	;con datos
	movlw		ESTAB
	BANK1
	movwf		html_socket

	return


;*********************************************************************
;*****************************************************************		*
;NOMBRE:		socket_control								*
	*
;CAPA:				3 - TRANSPORTE
	*		*
;DESCRIPCIÓN:	Administra los mensajes para inicial mantener	*		*
;					y finalizar el socket					*
	*
;*****************************************************************		*
socket_control;
				*
;*********************************************************************

	BANK1
	;Se actualiza el numero de acuse sumandole los datos recividos
	movf		tcp_data_Len0,W
	BANK0
	movwf		numero1_0
	BANK1
	movf		tcp_data_Len1,W
	BANK0
	movwf		numero1_1
	BANK1
	movf		xm_ack1_0,W
	BANK0
	movwf		numero2_0
	BANK1
	movf		xm_ack1_1,W
	BANK0
	movwf		numero2_1

	bcf				PCLATH,4
	bcf				PCLATH,3
	call		sumar
	bcf				PCLATH,4
	bsf				PCLATH,3

	movf		resultado0,W
	BANK1
	movwf		xm_ack1_0
	BANK0
	movf		resultado1,W
	BANK1
	movwf		xm_ack1_1

	;Se le suma uno a la segunda palabra del ACK si existio
	;un desborde en la suma de la palabra LSB
	BANK0
	movf		CarryOut,W
	sublw		0x01
	btfss		STATUS,Z
	goto		NOT_CARRY_OUT3		;numero1>numero2 (no hay acarreo)

		;Existio un desborde, por lo cual se sumara a la palabra ACK MSB
		BANK1
		movf		xm_ack0_1,W
		BANK0
		movwf		numero1_1
		BANK1
		movf		xm_ack0_0,W
		BANK0
		movwf		numero1_0
		movlw		0x00
		movwf		numero2_1
		movlw		0x01
		movwf		numero2_0
```

```
                bcf             PCLATH,4
                bcf             PCLATH,3
                call    sumar
                bcf             PCLATH,4
                bsf             PCLATH,3

                movf    resultado0,W
                BANK1
                movwf   xm_ack0_0
                BANK0
                movf    resultado1,W
                BANK1
                movwf   xm_ack1_1

        NOT_CARRY_OUT3

        ;Si se recivio un FIN se le suma uno  al ACK
        BANK1
        movlw   TCP_FIN
        andwf   flags,W
        btfsc   STATUS,Z
        goto    EXIT_SUM_SYN

                ;Se le suma uno al numero de acuse de recivo
                movf    xm_ack1_1,W
                BANK0
                movwf   numero1_1
                BANK1
                movf    xm_ack1_0,W
                BANK0
                movwf   numero1_0
                movlw   0x00
                movwf   numero2_1
                movlw   0x01
                movwf   numero2_0

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    sumar
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

                movf    resultado0,W
                BANK1
                movwf   xm_ack1_0
                BANK0
                movf    resultado1,W
                BANK1
                movwf   xm_ack1_1

        ;Se le suma uno a la segunda palabra del ACK si existio
        ;un desborde en la suma de la palabra LSB
        BANK0
        movf    CarryOut,W
        sublw   0x01
        btfss   STATUS,Z
        goto    NOT_CARRY_OUT4  ;numero1>numero2 (no hay acarreo)

                ;Existio un desborde, por lo cual se sumara a la palabra ACK MSB
                BANK1
                movf    xm_ack0_1,W
                BANK0
                movwf   numero1_1
                BANK1
                movf    xm_ack0_0,W
                BANK0
                movwf   numero1_0
                movlw   0x00
                movwf   numero2_1
                movlw   0x01
                movwf   numero2_0

                bcf             PCLATH,4    ;se cambia a la pagina 0
                bcf             PCLATH,3
                call    sumar
                bcf             PCLATH,4    ;se cambia a la pagina 1
                bsf             PCLATH,3

                movf    resultado0,W
                BANK1
                movwf   xm_ack0_0
                BANK0
                movf    resultado1,W
                BANK1
                movwf   xm_ack1_1
        NOT_CARRY_OUT4
EXIT_SUM_SYN


        ;Se carga la cabecera Ethernet
        BANK0
```

```
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    load_ethernet_header
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA


        ;Se establece la longitud del paquete IP
        ;20 para la cabecera IP
;20 para la cabecera TCP
;8 para las opciones
        clrf    IP_packet_length1
        movlw   0X28
        movwf   IP_packet_length0
        movlw   TCP
        movwf   IP_send_protocol

        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    load_IP_header
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

        ;Se establece es tamaño de la cabecera TCP
        ;Las banderas han sido definidas antes del llamado
        ;a esta función
        BANK1
        movlw   0x50
        movwf   data_flags1

        ;Se establece el valor del checksum inicial
        movlw   0x00
        movwf   tcp_Chksum1
        movlw   0x00
        movwf   tcp_Chksum0

        ;Se establece la longitud de la cabecera TCP
        clrf    tcp_length1
        movlw   0X14                    ;20 en decimal
        movwf   tcp_length0

        ;Se carga la cabecera TCP (en el banco 0)
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    load_tcp_header
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA


        ;FALTA LLAMADO AL PAD**********************************************************************************************************************************

        ;Se establecen los bytes a transmitir 60
        BANK0
        clrf    Len_pack1
        movlw   0X3C
        movwf   Len_pack0

        ;Se envian los datos almacenados en el buffer de anillo
        ;del RTL8019AS
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    TransmitLinkMgmt
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

        BANK1
        ;Si se envio un FIN se le suma uno al numero de secuencia
        movlw   TCP_FIN
        andwf   data_flags0,W
        btfsc   STATUS,Z
        goto    EXIT_ESTAB

                movf    xm_seq1_1,W
                BANK0
                movwf   numero1_1
                BANK1
                movf    xm_seq1_0,W
                BANK0
                movwf   numero1_0

                movlw   0x00
                movwf   numero2_1
                movlw   0x01
                movwf   numero2_0

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    sumar
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

                movf    resultado0,W
```

```
			BANK1
			movwf	xm_seq1_0
			BANK0
			movf	resultado1,W
			BANK1
			movwf	xm_seq1_1

		;Se le suma uno a la segunda palabra del numero de secuencia
		;si existio un desborde en la suma de la palabra LSB
		BANK0
		movf		CarryOut,W
		sublw		0x01
		btfss		STATUS,Z
		goto		NOT_CARRY_OUT5		;numero1>numero2 (no hay acarreo)

			;Existio un desborde, por lo cual se sumara a la palabra ACK MSB
			BANK1
			movf	xm_seq0_1,W
			BANK0
			movwf	numero1_1
			BANK1
			movf	xm_seq0_0,W
			BANK0
			movwf	numero1_0
			movlw	0x00
			movwf	numero2_1
			movlw	0x01
			movwf	numero2_0

			bcf			PCLATH,4	;se cambia a la pagina 0
			bcf			PCLATH,3
			call	sumar
			bcf			PCLATH,4	;se cambia a la pagina 1
			bsf			PCLATH,3

			movf	resultado0,W
			BANK1
			movwf	xm_seq0_0
			BANK0
			movf	resultado1,W
			BANK1
			movwf	xm_seq1_1

		NOT_CARRY_OUT5
EXIT_ESTAB

		BANK1
		return


;*******************************************************************
;*****************************************************************		*
;NOMBRE:		Generate_CRC								*
		*
;CAPA:				4 - APLICACIÓN
		*		*
;DESCRIPCIÓN:		Calcula el CRC		para el medio serial		*		*
;*****************************************************************		*
Generate_CRC;
				*
;*******************************************************************
		BANK1
		;Se aplica Xor con el primer byte de dato y el byte
		;menos significativo del CRC, el resultado se almacena
		;el CRC_LSB
		movf		Data_Modbus_Serial,W
		XorWF		CRC_LSB,F

		;Se inicializa el condator de lazos
		movlw		0x00
		movwf		Counter1

START_SHIFT

			movf	Counter1,W
			sublw	0x08
			btfsc	STATUS,Z
			goto	EXIT_SHIFT_BYTE

			;Se incrementa el contador de desplazamientos hacia la derecha
			incf	Counter1,F

			;Se establece el bit C de STATUS a cero (para el desplazamiento)
			bcf			STATUS,C

			;Desplazamiento del byte mas significativo del CRC
			RRF			CRC_MSB,F

			;El bit menos significativo de CRC_MSB ha quedado almacenado
			;en el bit C del registro STATUS
```

```
                    ;Desplazamiento del byte menos significativo del CRC
                    RRF                 CRC_LSB,F

                    ;Ahora el bit menos significativo de CRC_LSB ha quedado en
                    ;el bit C del registro STATUS, este bit nos servira para
                    ;el siguiente proceso

                    btfss       STATUS,C
                    goto        START_SHIFT

                    ;Se hara un Xor con la palabra 0xA001
                    movlw       0x01
                    XorWF       CRC_LSB,F

                    movlw       0xA0
                    XorWF       CRC_MSB,F

                    goto        START_SHIFT

EXIT_SHIFT_BYTE
        BANK1
return


;**********************************************************************
;*****************************************************************        *
;NOMBRE:            Request_treatment                                                                          *        *
;CAPA:                      4 -> APLICACIÓN
        *           *
;DESCRIPCIÓN:       Trata el paquete MODBUS entrante                                      *         *
;*****************************************************************        *
treatment_to_Request_MODBUS;                                                                                            *
;**********************************************************************

        ;//////////////////////MODBUS_PDU_Checking//////////////////////
        ;/CAPA:                     4 -> APLICACIÓN
        /
        ;/DESCRIPCIÓN:      Revisa la trama MODBUS entrante                           /
        ;////////////////////////////////////////////////////////////////
        ;Revisión del la cabecera MODBUS (MBAP header)

        ;|----------------------------------|
        ;|  Identificador de transación    | 2 bytes
        ;|----------------------------------|
        ;|  Identificador de protocolo     | 2 bytes
        ;|----------------------------------|
        ;|                     Longitud                    | 2 bytes
        ;|----------------------------------|
        ;|  Identificador de unidad            | 1 bytes
        ;|----------------------------------|

        ;Se obtiene el identificador de transacción
        ;Este valor se coloca nuevamente en la trama de respuesta para identificar
        ;la transacción
        BANK1
        movlw       NIC_DATA
        movwf       Register_Address1
        bcf                 PCLATH,4
        bcf                 PCLATH,3            ;PAGI NA 0 DE PROGRAMA
        call        inport1
        bcf                 PCLATH,4
        bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
        movf        Register_Value1,W
        movwf       TCP_transaction_id1     ;se almacena el identificador de transacción MSB

        bcf                 PCLATH,4
        bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
        call        inport1
        bcf                 PCLATH,4
        bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
        movf        Register_Value1,W
        movwf       TCP_transaction_id0     ;se almacena el identificador de transacción LSB

        ;Se obtiene el identificador de protocolo
        ;Unicamente se aceptan paquetes MODBUS (identificador = 0)
        bcf                 PCLATH,4
        bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
        call        inport1
        bcf                 PCLATH,4
        bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
        movf        Register_Value1,W
        movwf       TCP_protocol_id1 ;Se almacena el identificador de protocolo MSB

        bcf                 PCLATH,4
        bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
        call        inport1
        bcf                 PCLATH,4
        bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
        movf        Register_Value1,W
        movwf       TCP_protocol_id0 ;Se almacena el identificador de protocolo LSB
```

```
                ;Si el protocolo es diferente a MODBUS, se rechaza
                ;Se enviara un ACK de respuesta pero no se tratara la petición
                movlw   0xFF
                andwf   TCP_protocol_id0,W
                btfss   STATUS,Z
                goto    ERROR_HEADER_MODBUS

                movlw   0xFF
                andwf   TCP_protocol_id1,W
                btfss   STATUS,Z
                goto    ERROR_HEADER_MODBUS

;Se obtiene la longitud (la longitud partes desde el identificador de
;unidad e incluye todos los bytes restantes del paquete MODBUS)
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    inport1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
movf    Register_Value1,W
movwf   TCP_length_data1 ;Se almacena la longitud del paquete MSB

bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    inport1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
movf    Register_Value1,W
movwf   TCP_length_data0 ;Se almacena la longitud del paquete LSB

                ;La maxima longitud aceptada es 254 bytes (0xFE)
                ;de ser mayor, existe un error en la trama
                ;MAX LONG = 256 - 2 (CRC) = 254 bytes
                BANK0
                movlw   0x00
                movwf   numero1_1
                movlw   0xFE
                movwf   numero1_0
                BANK1
                movf    TCP_length_data1,W
                BANK0
                movwf   numero2_1
                BANK1
                movf    TCP_length_data0,W
                BANK0
                movwf   numero2_0

                ;Se resta numero1 - numero2,
                ;Si numero1 < numero2 entonces CarryOut=1
                bcf                     PCLATH,4
                bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA

                call    restar
                bcf                     PCLATH,4
                bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA


                ;Se establece si hubo un acarreo(numero1<numero2)
                movf    CarryOut,W
                sublw   0x01
                btfss   STATUS,Z
                goto    LENGTH_CORRECT      ;numero1>numero2 (no hay acarreo)
                goto    ERROR_HEADER_MODBUS

LENGTH_CORRECT

BANK1
;Se obtiene el identificador de unidad
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    inport1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
movf    Register_Value1,W
movwf   TCP_Unit_identifier

                ;El identificador de unidad maximo es 247 (0xF7)
                movlw   0x00
                BANK0
                movwf   numero2_1
                BANK1
                movf    TCP_Unit_identifier,W
                BANK0
                movwf   numero2_0
                movlw   0x00
                movwf   numero1_1
                movlw   0xF7
                movwf   numero1_0

                bcf                     PCLATH,4
```

```
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA

                call    restar
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA


                ;Se establece si hubo un acarreo(numero1<numero2)
                movf    CarryOut,W
                sublw   0x01
                btfss   STATUS,Z
                goto    UNIT_IDENTIFIER_CORRECT         ;numero1>numero2 (no hay acarreo)
                goto    ERROR_HEADER_MODBUS

UNIT_IDENTIFIER_CORRECT
goto    CONTINUE_PDU_ANALYSIS

;Si existio un error en la cabecera MODBUS solo se
;contesta con un ACK del paquete recivido
ERROR_HEADER_MODBUS
                BANK1
                movlw   0x01
                movwf   MB_Header_error         ;bandera por error en cabecera
                goto    RESPONSE_WITH_ACK;responder con un ACK

CONTINUE_PDU_ANALYSIS
; Se analiza el MODBUS PDU
; Se manejara DOS codigos de función.
;               - LECTURA DE REGISTROS DE ENTRADA  (0x04)
;               - ESCRITURA DE MULTIPLES REGISTROS  (0X16)
;|-------------------------|
;|   Codigo de Función           | 1 bytes
;|-------------------------|
;....(depende del codigo de función)

;Se obtiene el codigo de función
BANK1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    inport1
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

movf    Register_Value1,W
movwf   TCP_function_code

;Segun el codigo de función se envia al proceso respectivo

;--- PROCESO PARA ESCRITURA DE MULTIPLES REGISTROS (0X16) ---
movf    TCP_function_code,W
sublw   0x10
btfss   STATUS,Z
goto    CONTINUE_FUNTION_CODE_TEST

                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA
                goto    TREATMENT_WRITE_MULTIPLE_REGISTER
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

CONTINUE_FUNTION_CODE_TEST
;--- PROCESO PARA LESTURA DE REGISTROS DE ENTRADA ---
movf    TCP_function_code,W
sublw   0x04
btfss   STATUS,Z
goto    FUNCTION_CODE_INVALID

                goto    TREATMENT_READ_INPUT_REGISTER

FUNCTION_CODE_INVALID

                ;Si no es ninguno de los codigos anteriores, se ha
                ;recivido un codigo de función no soportado

                ;Se activa la bandera por error en el PDU
                movlw   0x01
                movwf   MB_PDU_error

                ;Se establece el tipo de excepción ocurrido
                movlw   EXCEPTION_01            ;Error en el codigo de función
                movwf   Exception_Code

                ;Siempre se responde con un ACK, pero esta vez tambien se
                ;incluye un paquete por error en la cabecera MODBUS (PDU)
                goto    RESPONSE_WITH_ACK


;/////TRATAMIENTO PARA LECTURA DE REGISTROS DE ENTRADA//////////
;/CAPA:                 4 -> APLICACIÓN
/
;/DESCRIPCIÓN:          Maneja las peticiones por lectura de registros          /
```

```
;/                                                    de entrada
                         /
;//////////////////////////////////////////////////////////////////
TREATMENT_READ_INPUT_REGISTER
            ;Datos recividos en una lectura de registros de entrada
            ;|--------------------------|
            ;|  Dirección de inicio     | 2 bytes
            ;|--------------------------|
            ;|  Cantidad de registros   | 2 bytes
            ;|--------------------------|

            ;Se obtiene la dirección de inicio
            ;No se restringe un rango debido a que ésta es tarea del esclavo serial
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

            call        inport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA


            movf        Register_Value1,W
            movwf       TCP_starting_address1

            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

            call        inport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA


            movf        Register_Value1,W
            movwf       TCP_starting_address0

            ;Se obtiene la cantidad de entradas
            ;La maxima cantidad de bytes permitidos serialmente en una trama
            ;es de 256 bytes (Se analiza el tamaño de la trama de respuesta)
            ;1 byte             Dirección del esclavo
            ;1 byte             Codigo de función
            ;1 byte             Cantidad de bytes a transmitir
            ;251 bytes   Datos transmitidos (125 registros/7D)
            ;2 bytes     CRC
            ;TOTAL = 256 bytes
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

            call        inport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA


            movf        Register_Value1,W
            movwf       TCP_quantity_inputs1

            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

            call        inport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA


            movf        Register_Value1,W
            movwf       TCP_quantity_inputs0

                        ;La cantidad de registros a leer no debe ser mayor a 125/7D
                        ;Se resta numero1 - numero2
                        ;Si numero1< numero2 entonces CarryOut se activa
                        movf        TCP_quantity_inputs1,W
                        BANK0
                        movwf       numero2_1
                        BANK1
                        movf        TCP_quantity_inputs0,W
                        BANK0
                        movwf       numero2_0
                        BANK1
                        movlw       0x00
                        BANK0
                        movwf       numero1_1
                        BANK1
                        movlw       0x7D
                        BANK0
                        movwf       numero1_0

                        ;Se resta numero1-numero2
                        bcf                 PCLATH,4
                        bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

                        call        restar
                        bcf                 PCLATH,4
```

```
            bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA


            ;Se establece si hubo un acarreo(numero1<numero2)
            movf    CarryOut,W
            sublw   0x01
            btfss   STATUS,Z
            goto    QUANTITY_INPUT_CORRECT      ;numero1>numero2 (no hay acarreo)

                    BANK1
                    ;Se activa la bandera por error en el PDU
                    movlw   0x01
                    movwf   MB_PDU_error

                    ;Se establece el tipo de excepción ocurrido
                    movlw   EXCEPTION_03        ;Error en la cantidad de datos
                    movwf   Exception_Code

                    goto    RESPONSE_WITH_ACK

QUANTITY_INPUT_CORRECT

;********** SE CONSTESTA CON UN ACK AL PAQUETE RECIVIDO *********
;Ha finalizado el tratamiento a la trama que genero la
;petición MODBUS
RESPONSE_WITH_ACK

;Se activan las banderas correspondientes a un ACK
BANK1
movlw   0x00
movwf   data_flags0
bsf             data_flags0,4           ;Se habilita la bandera TCP_ACK

;Se procede a enviar un ACK del paquete recivido
call    socket_control

;********** SE CIERRA EL PAQUETE EN TRATAMIENTO **********

;Petición MODBUS aceptada
;Ahora se procede a cerrar el tratamiento al paquete que
;genero esta petición
bcf             Request_MODBUS_present,2

;Se establece la pagina 0 en el RTL8019AS
BANK0
movlw   CR
movwf   Register_Address
movlw   0x22
movwf   Register_Value
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA

call    outport
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA


movlw   BNDRY
movwf   Register_Address
movf    Next_Ptr_Pck,W
movwf   Register_Value
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA

call    outport
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA



;********** SE INICIALIZAN LAS VARIABLES GENERALES **********
;Se inicializa a cero la variable para datos recividos
BANK1
movlw   0x00
movwf   length_data_modbus

;Se define la cantidad de bytes que esperamos recivir del medidor
;Se define como 2 * (Cantidad de Palabras a Leer)
movf    TCP_quantity_inputs0,W
movwf   TCP_quantity_inputs0_ORG
addwf   TCP_quantity_inputs0,F

;Se inicializa el checksum a cero
BANK0
clrf    Chksum0
clrf    Chksum1

;**********MANEJO DE ERRORES EN CABECERAS **********
;Si existe un eror en la cabecera MODBUS (sin tomar en cuenta el PDU),
;entonces no se envian mas paquetes
BANK1
movlw   0x01
```

```
andwf       MB_Header_error,W
btfss       STATUS,Z
goto        EXIT_TREATMENT_MODBUS ;Salir sin enviar mas paquetes

;Si existio un error en la recepción del PDU (a partir del codigo de función),
;se envia un mensaje de error
movlw       0x01
andwf       MB_PDU_error,W
btfss       STATUS,Z
goto        ERROR_IN_SLAVE                  ;Enviar paquete de notificación de error


;La trama MODBUS esta bien formada y se procede al tratamiento de la petición

;//////////////////MODBUS_Service_Processing//////////////////
;/CAPA:                  4 -> APLICACIÓN
/
;/MEDIO:                 Ethernet
;/DESCRIPCIÓN:           Revisa la trama MODBUS entrante                                 /
;////////////////////////////////////////////////////////////

;Servicio de analisis de petición
;Aqui se decide si la petición se envia a la aplicación de usuario
;o se hara tratamiento local, en este caso siempre se hara tratamiento
;local

;///////////////Proceso del servicio Local//////////////////
;/CAPA:                          4 - > APLICACIÓN
/
;/MEDIO:                 Ethernet
/
;/DESCRIPCIÓN:           La petición modbus se manejara localmente, lo       /
;/                                       significa que no se hara el llamado a la        /
;/                                       aplicación de usuario
/
;////////////////////////////////////////////////////////////

;SE TRANSFIERE EL CONTROL AL MEDIO SERIAL PARA EN TRATAMIENTO DE LA
;PETICIÓN MODBUS

;////////////////////request_to_slaveMODBUS//////////////////
;/CAPA:                  4 -> APLICACIÓN
/
;/MEDIO:                 Serial
            /
;/DESCRIPCIÓN:           Envia una petición al esclavo MODBUS                    /
;////////////////////////////////////////////////////////////

; Emisión peticiòn hacia el esclavo MODBUS via serial (SCI)
;|-------------------------|
;|   Dirección de esclavo   | 1 bytes
;|-------------------------|
;|    Codigo de función                | 1 bytes
;|-------------------------|
;|    Dirección de inicio     | 2 bytes
;|-------------------------|
;|                Cantidad                         | 2 bytes
;|-------------------------|
;|        CRC                                      | 2 bytes
;|-------------------------|

;******** SE INICIA EL PROCESO DE TRANSMISIÓN DE LA TRAMA *********
;Se inicializa el CRC con 0xFFFF
BANK1
movlw       0XFF
movwf       CRC_LSB
movwf       CRC_MSB

;Se envia el IDENTIFICADOR DE UNIDAD (via serial SCI)
;tambien llamada dirección de esclavo
BANK1
movf        TCP_Unit_identifier,W
bcf                 PCLATH,4
bcf                 PCLATH,3            ;Pagina 0 del programa
BANK0
call        RS232_Tx
BANK1
bcf                 PCLATH,4
bsf                 PCLATH,3            ;Pagina 1 del programa
;Se calcula el CRC del dato enviado
movf        TCP_Unit_identifier,W
movwf       Data_Modbus_Serial
call        Generate_CRC


;Se envia el CODIGO DE FUNCIÓN (via serial SCI)
movf        TCP_function_code,W
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
```

```
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
;Se calcula el CRC del dato enviado
movf        TCP_function_code,W
movwf       Data_Modbus_Serial
call        Generate_CRC


;Se envia la DIRECCIÓN DE INICIO (via serial SCI) MSB
movf        TCP_starting_address1,W
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
;Se calcula el CRC del dato enviado
movf        TCP_starting_address1,W
movwf       Data_Modbus_Serial
call        Generate_CRC


;Se envia la DIRECCIÓN DE INICIO (via serial SCI) LSB
movf        TCP_starting_address0,W
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
;Se calcula el CRC del dato enviado
movf        TCP_starting_address0,W
movwf       Data_Modbus_Serial
call        Generate_CRC


;Se envia la CANTIDAD DE ENTRADAS (via serial SCI) MSB
movlw       0x00
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
;Se calcula el CRC del dato enviado
movlw       0x00
movwf       Data_Modbus_Serial
call        Generate_CRC


;Se envia la CANTIDAD DE ENTRADAS (via serial SCI) LSB
movf        TCP_quantity_inputs0_ORG,W
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA
;Se calcula el CRC del dato enviado
movf        TCP_quantity_inputs0_ORG,W
movwf       Data_Modbus_Serial
call        Generate_CRC


;Se ha completado la transmisión de la trama basica y se
;procede a transmitir el CRC, primero el LSB y luego el MSB
movf        CRC_LSB,W
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA

movf        CRC_MSB,W
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
BANK0
call        RS232_Tx
BANK1
bcf                     PCLATH,4
bsf                     PCLATH,3                ;PAGINA 1 DE PROGRAMA

;Se ha completado la transmisión de datos hacia el esclavo MODBUS
;*****************INICIALIZA TIMER A 3.5 CARACTERES *****************
;Se inicializa un TIMER de 3.5 caracteres (tiempo entre paquetes SCI)
BANK0
```

```
bcf                     PIR1,TMR1IF                     ;limpia la bandera TMR1IF
movlw   TIMER3.5_L      ;Se carga el registro del TMR para 3.5 caracteres
movwf   TMR1L
movlw   TIMER3.5_H
movwf   TMR1H
bsf                     T1CON,TMR1ON    ;TMR1IF en ON

;************** SE HABILITA LA  INTERRUPCION DEL TMR0 **************
;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE EHTERNET
BANK1
bsf                     INTCON,T0IE
;************* SE HABILITAN LAS INTERRUPCIONES VIA SCI **************
bsf                     PIE1,RCIE

        ;**** ESPERA FINALIZACIÓN DE TIEMPO DE 3.5 CARACTERES ******
        ;Se espera a que finalize un tiempo de 3.5 caracteres
        ;Este tiempo es el minimo entre tramas
        BANK0
        WAIT_t3.5
                btfss   PIR1,TMR1IF
                goto    WAIT_t3.5

        ;Se apaga el timer1
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

;************* SE DESABILITAN LAS INTERRUPCIONES VIA SCI **************
BANK1
bcf                     PIE1,RCIE

;//////////Espera por la contestación del esclavo MODBUS///////////////

;/CAPA:                 4 -> APLICACION
                        /
;/MEDIO:                Serial
                        /
;/DESCRIPCIÓN:          Espera a que el esclavo MODBUS responda a la petición   /
;/                      enviada. Inicia el reloj para el tiempo de espera       /
;////////////////////////////////////////////////////////////////

;Se inicializa el CRC con 0xFFFF
;con el fin de calcular el CRC de la trama recivida serialmente
BANK1
movlw   0XFF
movwf   CRC_LSB
movwf   CRC_MSB

;Se activa el tiempo de espera para la respuesta del esclavo
;Esto es requerido para no esperara indefinidamente por la respuesta

;Se establece a cero el contador del lazo externo
BANK2
clrf    CounterExt

RESPONSE_TIME_OUT2
        ;Se inicializa a cero el contador de rebasos del TIMER1
        BANK0
        clrf    Counter

        RESPONSE_TIME_OUT1
                ;Se inicializa el timer 1 (13.107ms por cada rebasamiento)
                BANK0
                bcf             PIR1,TMR1IF             ;limpia la bandera TMR1IF
                movlw   0X00
                movwf   TMR1L
                movlw   0X00
                movwf   TMR1H
                bsf             T1CON,TMR1ON    ;TMR1IF en ON

                ;Se espera hasta que se de el rebaso del TIMER1 (de FFFF a 0000 )
                ;o se reciva el primer byte serial desde el medidor
                BANK0
                WAIT_OVERFLOW_TIMER1
                        btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                        goto    FIRST_BYTE_HEADER
                        btfss   PIR1,TMR1IF             ;Revisa si el TIMER1 se ha desbordado
                        goto    WAIT_OVERFLOW_TIMER1

                ;Se apaga el timer1
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                ;Se ha producido un rebaso y por lo tanto se debe incrementar el contador
                incf    Counter,F

                ;Se revisa si el numero de rebasos del timer 1 ha sido completado
                movf    Counter,W
                sublw   MAX_OVERFLOW_TIMER1_1
                btfss   STATUS,Z
                goto    RESPONSE_TIME_OUT1

;Se ha finalizado el contador de rebasos internos
;Ahora se procede al lazo externo
;Se incrementa el contador externo
```

```
                BANK2
                incf        CounterExt,F

                ;Se revisa si el numero de rebasos se ha superado
                movf        CounterExt,W
                sublw       MAX_OVERFLOW_TIMER1_2
                btfss       STATUS,Z
                goto        RESPONSE_TIME_OUT2

                            ;Se ha terminado la espera para la respuesta del esclavo
                            ;Se establece el checksum de los datos a cero
                            BANK0
                            clrf        Chksum0
                            clrf        Chksum1

                            ;Se establece el codigo de exepción
                            BANK1
                            movlw       EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                            movwf       Exception_Code

                            ;Se inicializa a cero la variable para datos recividos
                            movlw       0x00
                            movwf       length_data_modbus

                            goto        ERROR_IN_SLAVE                  ;Se ha superado el tiempo de espera

;//////////////////////Recepción de la contestación//////////////////////
;/CAPA:                     4 -> APLICACION
                            /
;/MEDIO:                    Serial
                            /
;/DESCRIPCIÓN:              Empieza la recepción de datos desde el dispositivo          /
;/                                       esclavo  serial MODBUS
                            /
;////////////////////////////////////////////////////////////////////

;Ha llegado el primer byte serial, que forma parte de la cabecera MODBUS serial
;|----------------------|
;| Dirección del esclavo |   1 byte
;|----------------------|
;|   Codigo de Función   |   1 byte
;|----------------------|
;|  Contador de bytes     |          1 byte
;|----------------------|
;|                          DATOS                  |  0 hasta 251 bytes
;|----------------------|
;|                          CRC                                    |          2 bytes
;|----------------------|

;Recepción de la cabecera MODBUS serial desde el esclavo (MEDIDOR DE ENERGIA)
FIRST_BYTE_HEADER

;************** SE DESABILITA LA INTERRUPCION DEL TMR0 **************
;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE ETHERNET
BANK1
bcf                 INTCON,T0IE

;---------- Se captura la dirección del esclavo ---------
BANK0
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        RS232_Rx
bcf                 PCLATH,4
bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
BANK1
movwf       sci_data0           ;Se almacena el dato obtenido serialmente

;Si fue enviado por el esclavo esperado se acepta, de lo contrario
;se ignora y se continua con la espera (notar que el timer1 no ha sido detenido)
movf        TCP_Unit_identifier,W
subwf       sci_data0,W
btfsc       STATUS,Z
goto        SLAVE_CORRECT       ;Si no es del esclavo esperado se ignora

            ;************** SE HABILITA LA INTERRUPCION DEL TMR0 **************
            ;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE ETHERNET
            bsf             INTCON,T0IE
            BANK0
            goto        WAIT_OVERFLOW_TIMER1        ;Si no es del esclavo esperado se ignora

SLAVE_CORRECT

;Se apaga el timer1
BANK0
bcf                 T1CON,TMR1ON        ;TMR1IF en OFF

;************** SE RELLENAN LAS CABECERAS CON CEROS **********************
;Configuramos para escritura remota por el canal DMA al buffer del RTL
;Configuro el canal remoto DMA para escritura
BANK0
movlw       WRITE                       ;Escritura de datos
movwf       Read_Write
```

```
movlw   XMT_BUF_START   ;Dirección de inicio del buffer de transmision
movwf   Addr1
movlw   0x00
movwf   Addr0

bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    remote_dma_setup;Configura el RTL para escritura remota
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;Se inicializa a cero el contador de bytes de la trama a transmitir
movlw   0x00
movwf   FrameCount

;Inicializo el contador de bytes a escribir a 63
;14 bytes para cabecera Ethernet
;20 bytes para cabercera Internet
;20 bytes para cabecera TCP
;7      bytes para cabecera MODBUS
;2  bytes PDU modbus (codigo de función y longitud de bytes)
;       tambien puede significar codigo de error y tipo de error
movlw   0x3F            ;63 en decimal
movwf   Header_Eth_IP_TCP

;Se inicializa el contador de lazo a cero
movlw   0x00            ;54 en decimal
movwf   Counter

;Se crea un lazo para rellenar con ceros las cabeceras ETHERNET,
;IP y TCP(MODBUS)
LOOP_FILL_HEADER
        movlw   NIC_DATA
        movwf   Register_Address
        movlw   0x00
        movwf   Register_Value

        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

        incf    Counter,F
        movf    Counter,W
        subwf   Header_Eth_IP_TCP,W
        btfss   STATUS,Z
        goto    LOOP_FILL_HEADER

;Se calcula el CRC del dato recivido
BANK1
movf    sci_data0,W
movwf   Data_Modbus_Serial
call    Generate_CRC

;El dato recivido es del esclavo que se esperaba, por lo tanto se inicia
;el proceso para la recepción de la trama

;Se deben iniciar el timer1 a t1.5 (espera 1.5 caracteres antes de finalizar la
;recepción del paquete MODBUS del esclavo)
call    Timer1_To_1.5

;------- Ahora se espera recivir el codigo de función o de error ---------
WAIT_RCV_FUNC_CODE
        btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
        goto    FUNC_CODE_RCV
        btfss   PIR1,TMR1IF         ;Revisa si el TIMER1 se ha desbordado
        goto    WAIT_RCV_FUNC_CODE

        ;El tiempo de 1.5 caracteres se ha superado
        ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

        BANK1
        movlw   EXCEPTION_04    ;Error por falla en el dispositivo esclavo
        movwf   Exception_Code
        goto    ERROR_IN_SLAVE

;Se ha recivido el byte de codigo de función o el codigo de error
FUNC_CODE_RCV
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

        ;Se captura el codigo de función o de error
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bcf             PCLATH,4
```

```
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
                BANK1
                movwf   sci_data0

                ;Se calcula el CRC del dato recivido
                movwf   Data_Modbus_Serial
                call    Generate_CRC

                ;Se compara con el codigo de función esperado
                movf    TCP_function_code,W
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    RCV_BYTE_COUNT

                ;Se compara con el codigo de error esperado
                movlw   0x84
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    RCV_EXCEPTION_CODE

                        ;Si no es ninguno de estos codigo, existe un error
                        movlw   EXCEPTION_04    ;Error por codigo de función no esperado
                        movwf   Exception_Code
                        goto    ERROR_IN_SLAVE

;--------- SE HA RECIVIDO UNA EXCEPCIÓN AL PAQUETE ENVIADO ----------
;Se espera recivir el codigo de exepción
RCV_EXCEPTION_CODE
;Se deben iniciar el timer1 a 1.5 caracteres
call    Timer1_To_1.5

WAIT_RCV_EXCEPTION_CODE
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    EXEPTION_CODE_RCV
                btfss   PIR1,TMR1IF     ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_EXCEPTION_CODE

                ;El tiempo de 1.5 caracteres se ha superado
                ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04    ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE

;Se ha recivido el byte de codigo de excepción
EXEPTION_CODE_RCV
                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                ;Se captura el codigo excepcion
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
                BANK1
                movwf   Exception_Code

                ;----------  Se captura el CRC ----------
                ;Se deben iniciar el timer1 a 1.5 caracteres
                call    Timer1_To_1.5

                WAIT_RCV_CRC_ERROR
                        btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                        goto    RCV_CRC_ERROR1
                        btfss   PIR1,TMR1IF     ;Revisa si el TIMER1 se ha desbordado
                        goto    WAIT_RCV_CRC_ERROR

                        goto    ERROR_IN_SLAVE

                RCV_CRC_ERROR1
                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                ;Se captura el CRC LSB
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

                ;Se deben iniciar el timer1 a 1.5 caracteres
                call    Timer1_To_1.5

                WAIT_RCV_CRC_ERROR2
```

```
                btfsc   PIR1,RCIF    ;Revisa si ha llegado un byte serial
                goto    RCV_CRC_ERROR2
                btfss   PIR1,TMR1IF             ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_CRC_ERROR2

                goto    ERROR_IN_SLAVE

        RCV_CRC_ERROR2
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON            ;TMR1IF en OFF

        ;Se captura el CRC MSB
        bcf             PCLATH,4
        bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bcf             PCLATH,4
        bsf             PCLATH,3                ;PAGINA 1 DE PROGRAMA

        goto    ERROR_IN_SLAVE

;Ahora se espera  recivir el contador de  bytes
;Es la cantidad de bytes en los datos
RCV_BYTE_COUNT
call    Timer1_To_1.5

;---------- Ahora se espera recivir el contador de bytes ---------
WAIT_RCV_BYTE_COUNT
        btfsc   PIR1,RCIF    ;Revisa si ha llegado un byte serial
        goto    COUNT_BYTE_RCV
        btfss   PIR1,TMR1IF             ;Revisa si el TIMER1 se ha desbordado
        goto    WAIT_RCV_BYTE_COUNT

        ;El tiempo de 1.5 caracteres se ha superado
        ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON            ;TMR1IF en OFF

        BANK1
        movlw   EXCEPTION_04            ;Error por falla en el dispositivo esclavo
        movwf   Exception_Code
        goto    ERROR_IN_SLAVE

;Se ha recivido el byte de codigo de función
COUNT_BYTE_RCV
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON            ;TMR1IF en OFF

        ;Se captura el contador de bytes
        bcf             PCLATH,4
        bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bcf             PCLATH,4
        bsf             PCLATH,3                ;PAGINA 1 DE PROGRAMA
        BANK1
        movwf   sci_data0

        ;Se calcula el CRC del dato recivido
        movwf   Data_Modbus_Serial
        call    Generate_CRC

        ;Se compara con la cantidad de datos esperada
        movf    TCP_quantity_inputs0,W
        subwf   sci_data0,W
        btfsc   STATUS,Z
        goto    CONT_WITH_DATA_RCV

                movlw   EXCEPTION_04            ;Error por codigo de función no esperado
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE

;---------- Ahora se procede a recivir los datos ------------
;La cabecera MODBUS serial ha sido revisada y es correcta
;El primer dato que se recive forma parte de una palabra con
;el contador de bytes de la cabecera MODBUS respuesta y para
;propositos de checksum se obtiene individualmente

;Se deben iniciar el timer a t1.5 (esperar 1.5 caracteres)
CONT_WITH_DATA_RCV

;Se deben iniciar el timer a t1.5 (esperar 1.5 caracteres)
call    Timer1_To_1.5

;Espera por la llegada del primer byte de datos
WAIT_FIST_DATA
        btfsc   PIR1,RCIF    ;Revisa si ha llegado un byte serial
        goto    FIRST_DATA_RCV
        btfss   PIR1,TMR1IF             ;Revisa si el TIMER1 se ha desbordado
        goto    WAIT_FIST_DATA
```

```
                ;Se ha desbordado t1.5, lo cual se debe a un error en el esclavo
                ;MODBUS (medidor)
                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON        ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE

FIRST_DATA_RCV
;Se apaga el timer1
bcf             T1CON,TMR1ON        ;TMR1IF en OFF

;Se captura el primer dato recivido
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    RS232_Rx
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
BANK1
movwf   sci_data0           ;Se almacena el dato

;Se calcula el CRC del dato recivido
movwf   Data_Modbus_Serial
call    Generate_CRC

;Se deben iniciar el timer a t1.5 (esperar 1.5 caracteres)
call    Timer1_To_1.5

;Ahora se carga el primer byte recivido en el buffer de anillo del RTL
BANK1
movf    sci_data0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Se calcula el checksum de el primer byte en conjunto con la cantidad de
;bytes de respuesta (que es la cantidad de registros a leer por dos )
movf    sci_data0,W
BANK0
movwf   Out_Word0
BANK1
movf    TCP_quantity_inputs0,W
BANK0
movwf   Out_Word1

bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    checksum
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Se incrementa el contador de datos recividos
BANK1
incf    length_data_modbus,F

;La captura del primer dato serial desde el esclavo MODBUS
;se hace independiente con el proposito de calcular adecuadamente
;el checksum del paquete TCP

;Ahora empieza la descarga de los demas bytes de datos
LOAD_MODBUS_DATA
                ;Si llega un dato antes de t=1.5 caracteres se aceptan
                ;como parte de la trama, de lo contrario son parte de otra
                ;trama o ha existido un error
                BANK0
                Rx_Data1
                        btfsc           PIR1,RCIF           ;Llego un nuevo dato SCI?
                        goto            DATA_ARRIVE                 ;Ha llegado un nuevo dato
                        btfsc           PIR1,TMR1IF                 ;Expiro el timer t1.5?
                        goto            ERROR_INCOMPLETE_DATA;ha recivido una cantidad impar de datos
                        goto            Rx_Data1            ;Ninguno de los anteriores

                ;Ha llegado un nuevo dato que forma parte de la misma trama
                DATA_ARRIVE
                bcf             PCLATH,4
                bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx            ;Se obtiene el dato
                bcf             PCLATH,4
                bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
                BANK1
                movwf   sci_data1

                ;Se calcula el CRC del dato recivido
                movwf   Data_Modbus_Serial
                call    Generate_CRC
```

```
;Se apaga el timer1
BANK0
bcf             T1CON,TMR1ON        ;TMR1IF en OFF

;Se deben iniciar el timer a t1.5 (esperar 1.5 caracteres)
call    Timer1_To_1.5

;Ahora se cargan en el buffer de anillo del RTL
;Se carga el primer byte de la palabra MSB
BANK1
movf    sci_data1,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Se incrementa el contador de datos recividos
BANK1
incf    length_data_modbus,F

;Antes de verificar la llegada del siguiente byte se verifica si
;han llegado todos los bytes de datos esperados (despues esperaremos
;la llegada del CRC, pero estos no conforman parte de la trama MODBUS/TCP)
movf    TCP_quantity_inputs0,W
subwf   length_data_modbus,W
btfsc   STATUS,Z
goto    FULL_DATA_RECEIVE

;Nuevamente se comprueba si ha llegado un segundo dato
;o si el tiempo de 1.5 caracteres (t1.5) ha expirado
BANK0
Rx_Data2
        btfsc           PIR1,RCIF   ;Llego un nuevo dato?
        goto            DATA_ARRIVE2;Ha llegado un nuevo dato
        btfsc           PIR1,TMR1IF         ;Expiro el timer t1.5?
        goto            ERROR_INCOMPLETE_DATA;Ha expirado t1.5 cuando aun no se han
                                                                    ;recivido todos
los datos esperados
        goto            Rx_Data2    ;Ninguno de los anteriores

DATA_ARRIVE2
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    RS232_Rx
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
BANK1
movwf   sci_data0

;Se calcula el CRC del dato recivido
movwf   Data_Modbus_Serial
call    Generate_CRC

;Se apaga el timer1
BANK0
bcf             T1CON,TMR1ON        ;TMR1IF en OFF

;Se deben iniciar el timer a t1.5 (esperar 1.5 caracteres)
call    Timer1_To_1.5

;Ahora se cargan en el buffer de anillo del RTL
;Se carga la primer palabra MSB
BANK1
movf    sci_data0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Se calcula el checksum de los datos enviados
movf    sci_data0,W
BANK0
movwf   Out_Word0
BANK1
movf    sci_data1,W
BANK0
movwf   Out_Word1

bcf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
call    checksum
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Se incrementa el contador de datos recividos
BANK1
incf    length_data_modbus,F
```

```
                goto        LOAD_MODBUS_DATA

;Ha ocurrido un error debido a la recepcion de una cantidad
;impar de datos MODBUS desde el esclavo
ERROR_INCOMPLETE_DATA

                ;Se apaga el timer1
                BANK0
                bcf                 T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se limpia la suma de control de los datos
                clrf        Chksum0
                clrf        Chksum1

                ;Se limpia la  cantidad de bytes recibidos
                ;Se limpia el checksum de los datos antes recividos
                BANK1
                clrf        length_data_modbus

                ;Se establece el codigo de excepcion
                movlw       EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                movwf       Exception_Code
                goto        ERROR_IN_SLAVE

FULL_DATA_RECEIVE

                ;El TMR1 sigue corriendo con un tiempo de 1.5 caracteres
                ;Esperamos por la llegada del CRC (LSB)
                BANK0
                Rx_CRC_LSB
                        btfsc               PIR1,RCIF           ;Llego un nuevo dato?
                        goto                ARRIVE_CRC_LSB      ;Ha llegado un nuevo dato
                        btfsc               PIR1,TMR1IF                 ;Expiro el timer t1.5?
                        goto                ERROR_INCOMPLETE_DATA       ;Ha expirado t1.5 cuando aun no se
                                                                                ;ha completado el paquete MODBUS
                        goto                Rx_CRC_LSB          ;Sigue esperando por la llegada

                ;Ha llegado el primer byte del CRC (LSB)
ARRIVE_CRC_LSB
                bcf                 PCLATH,4
                bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call        RS232_Rx
                bcf                 PCLATH,4
                bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
                BANK1
                movwf       Receive_CRC_LSB     ;Almacena el byte menos significativo del CRC

                ;Se apaga el timer1
                BANK0
                bcf                 T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se deben iniciar el timer a t1.5 (esperar 1.5 caracteres)
                call        Timer1_To_1.5

                ;Esperamos por la llegada del CRC (MSB)
                Rx_CRC_MSB
                        btfsc               PIR1,RCIF           ;Llego un nuevo dato?
                        goto                ARRIVE_CRC_MSB      ;Ha llegado un nuevo dato
                        btfsc               PIR1,TMR1IF                 ;Expiro el timer t1.5?
                        goto                ERROR_INCOMPLETE_DATA       ;Ha expirado t1.5 cuando aun no se
                                                                                ;ha completado el paquete MODBUS
                        goto                Rx_CRC_MSB          ;Se continua esperando

                ;Ha llegado el segundo byte del CRC (MSB)
ARRIVE_CRC_MSB
                bcf                 PCLATH,4
                bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call        RS232_Rx
                bcf                 PCLATH,4
                bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
                BANK1
                movwf       Receive_CRC_MSB     ;Almacena el byte mas significativo del CRC

                ;Se apaga el timer1
                BANK0
                bcf                 T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se determina si el CRC es el correcto
                BANK1
                movf        CRC_LSB,W
                subwf       Receive_CRC_LSB,W
                btfss       STATUS,Z
                goto        ERROR_INCOMPLETE_DATA

                movf        CRC_MSB,W
                subwf       Receive_CRC_MSB,W
                btfss       STATUS,Z
                goto        ERROR_INCOMPLETE_DATA

                ;********** INCORPORACIÓN DEL PAD **********

                ;La cantidad de datos es la requerida
```

```
                ;Se hace un relleno de un byte (esto debido a que los datos
                ;modbus terminan en byte impar, por lo cual es
                ;requerido agregar un byte con ceros para propositos
                ;del calculo del checksum)
                BANK1
                movlw   NIC_DATA
                movwf   Register_Address1
                movlw   0x00
                movwf   Register_Value1
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    outport1
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

                ;Se calcula el checksum de los datos enviados
                movlw   0x00
                BANK0
                movwf   Out_Word0
                BANK1
                movf    sci_data1,W
                BANK0
                movwf   Out_Word1

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    checksum
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

                goto    SEND_RESPONSE

;*************** CONSTRUIR EXCEPCIÓN MODBUS DE RESPUESTA ****************
;* CAPA:                4 -> APLICACIÓN                                    *
                        *
;* DESCRIPCIÓN          Define el tipo de excepción generada en el esclavo *
;************************************************************************
;Se define el tipo de error ocurrido y se envia una respuesta
ERROR_IN_SLAVE

                ;************** SE DESABILITA LA INTERRUPCION DEL TMR0 **************
                ;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE EHTERNET
                BANK1
                bcf             INTCON,T0IE

                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                BANK1
                ;Se activa la bandera de error
                movlw   0x01
                movwf   Error_flag

                ;Se establece el tipo de error ocurrido
                movlw   0x80
                movwf   Error_Code
                movf    TCP_function_code,W
                addwf   Error_Code,F

;************************CONSTRUIR RESPUESTA MODBUS************************
;* CAPA:                4 -> APLICACIÓN                                    *
                        *
;* DESCRIPCIÓN          Contruye el paquete de respuesta a la petición MODBUS *
;************************************************************************
SEND_RESPONSE

;Luego de la recepción erronea o correcta de la trama, se debe esperar
;un tiempo de 3.5 caracteres antes de realizar una nueva petición por el medio

;Se inicializa el timer1 para 3.5 caracteres

;****************** RETARDO DE 3.5 CARACTERES SERIAL **********************
;Previo a la transmisión de paquete MODBUS via serial se genera un
;retardo de 3.5 caracteres
;Programación del TMR1 con un tiempo de 3.5 caracteres
BANK0
bcf             PIR1,TMR1IF                             ;limpia la bandera TMR1IF
movlw   TIMER3.5_L                      ;Se carga el registro del TMR para 3.5  caracteres
movwf   TMR1L
movlw   TIMER3.5_H
movwf   TMR1H
bsf             T1CON,TMR1ON    ;TMR1IF en ON


;Ahora es importante conocer el estado del socket antes de responder
;Ya que exise la posibilidad de que ahiga sido cerrado miestras se esperaba por la
;repuesta del esclavo serial MODBUS
BANK1
movlw   ESTAB
subwf   html_socket,W
btfsc   STATUS,Z
```

```
        goto    CONT_WITH_SEND_FRAME ;Verificar si el estado es FIN
                goto    EXIT_AND_WAIT_T3.5

        CONT_WITH_SEND_FRAME

        ;Se guarda el checksum de los datos
        BANK0
        movf    Chksum0,W
        BANK1
        movwf   tcp_Chksum0
        BANK0
        movf    Chksum1,W
        BANK1
        movwf   tcp_Chksum1

        ;Ahora que se han cargado los datos en el buffer del RTL, se
        ;empiezan a cargar las cabeceras Ethernet, IP, TCP y MOBUS

        ;Se carga la cabecera Ethernet en el buffer de anillo del RTL8019AS
        BANK0
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    load_ethernet_header
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

        ;Se carga la cabecera IP en el buffer de anillo del RTL, primero se
        ;establece el tamaño minimo requerido en las cabeceras:
        ;20 bytes para cabecera IP
;20 bytes para cabecera TCP
;9  bytes para cabecera MODBUS (7 de cabecera básica)
        ;?  bytes de datos ( luego se agregan )
        BANK0
        clrf    IP_packet_length1
        movlw   0X31                    ;49
        movwf   IP_packet_length0

        ;Se le incrementa a la cantidad de datos almacenados
        movf    IP_packet_length0,W
        movwf   numero1_0
        movf    IP_packet_length1,W
        movwf   numero1_1
        BANK1
        movf    length_data_modbus,W
        BANK0
        movwf   numero2_0
        movlw   0x00
        movwf   numero2_1

        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    sumar
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

        movf    resultado0,W
        movwf   IP_packet_length0
        movf    resultado1,W
        movwf   IP_packet_length1

        ;Se define un nuevo numero de indentificación
        movf    Identification0,W
        movwf   numero1_0
        movf    Identification1,W
        movwf   numero1_1
        movlw   0x01
        movwf   numero2_0
        movlw   0x00
        movwf   numero2_1

        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    sumar
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

        movf    resultado0,W
        movwf   Identification0
        movf    resultado1,W
        movwf   Identification1

        ;Se establece el protocolo dentro de la trama IP
        movlw   TCP
        movwf   IP_send_protocol

        ;Se carga la cabecera IP
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    load_IP_header
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
```

```
;Se carga la cabecera TCP en el buffer del anillo del RTL
;Se definen las banderas en la cabecera TCP
BANK1
movlw   0x50                    ;Tamaño de la cabecera TCP (valor fijo)
movwf   data_flags1
movlw   0x00
movwf   data_flags0
bsf             data_flags0,4           ;Se habilita la bandera TCP_ACK
bsf             data_flags0,3           ;Se habilita la bandera TCP_PUSH

;Se almacena el tamaño de la cabecera TCP y MODBUS
;Primero se establece el tamaño minimo para ambas cabeceras:
;20 bytes de cabecera TCP
;9  bytes de cabecera MODBUS
;?      despues de agregan los bytes de datos
clrf    tcp_length1
movlw   0X1D                            ;29 en decimal
movwf   tcp_length0

;Se le incrementa la cantidad de datos almacenados
movf    tcp_length0,W
BANK0
movwf   numero1_0
BANK1
movf    tcp_length1,W
BANK0
movwf   numero1_1
BANK1
movf    length_data_modbus,W
BANK0
movwf   numero2_0
movlw   0x00
movwf   numero2_1

bcf             PCLATH,4
bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    sumar
bcf             PCLATH,4
bsf             PCLATH,3                ;PAGINA 1 DE PROGRAMA

movf    resultado0,W
BANK1
movwf   tcp_length0
BANK0
movf    resultado1,W
BANK1
movwf   tcp_length1

;Con el proposito de cargar el checksum en la cabecera TCP
;se calcula el checksum de la cabecera MODBUS sin cargarla
;en el buffer del RTL
movf    tcp_Chksum0,W
BANK0
movwf   Chksum0
BANK1
movf    tcp_Chksum1,W
BANK0
movwf   Chksum1

;Se agrega al checksum el numero de transacción
BANK1
movf    TCP_transaction_id0,W
BANK0
movwf   Out_Word0
BANK1
movf    TCP_transaction_id1,W
BANK0
movwf   Out_Word1

bcf             PCLATH,4
bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    checksum
bcf             PCLATH,4
bsf             PCLATH,3                ;PAGINA 1 DE PROGRAMA

;Se agrega al checksum el identificador de protocolo
BANK1
movf    TCP_protocol_id0,W
BANK0
movwf   Out_Word0
BANK1
movf    TCP_protocol_id1,W
BANK0
movwf   Out_Word1

bcf             PCLATH,4
bcf             PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    checksum
bcf             PCLATH,4
bsf             PCLATH,3                ;PAGINA 1 DE PROGRAMA
```

```
;Se agrega al checksum la longitud
;Se establece la longitud de cabecera (obligatoria)
; 1 byte para el identificador de unidad
; 1 byte para el codigo de función
; 1 byte para la cantidad de bytes
BANK1
movlw   0x03
movwf   TCP_bytes_data0
clrf    TCP_bytes_data1

;Se le suma la cantidad de datos MODBUS  a enviar
movf    TCP_bytes_data0,W
BANK0
movwf   numero1_0
BANK1
movf    TCP_bytes_data1,W
BANK0
movwf   numero1_1
BANK1
movf    length_data_modbus,W
BANK0
movwf   numero2_0
movlw   0x00
movwf   numero2_1

bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    sumar
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

movf    resultado0,W
BANK1
movwf   TCP_bytes_data0
BANK0
movf    resultado1,W
BANK1
movwf   TCP_bytes_data1

;Se le suma la cantidad de bytes de datos MODBUS
;al checksum
BANK1
movf    TCP_bytes_data0,W
BANK0
movwf   Out_Word0
BANK1
movf    TCP_bytes_data1,W
BANK0
movwf   Out_Word1

bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    checksum
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;Confirma si la respuesta sera debido a un error
BANK1
movlw   0x01
andwf   Error_flag,W
btfss   STATUS,Z
goto    LOAD_ERROR_FRAME

        ;Se agrega al checksum el codigo de funcion e
        ;identificador
        movf    TCP_function_code,W
        BANK0
        movwf   Out_Word0
        BANK1
        movf    TCP_Unit_identifier,W
        BANK0
        movwf   Out_Word1

        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    checksum
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
        goto    EXIT_LOAD_ERROR_FRAME

LOAD_ERROR_FRAME
        ;Se agrega al checksum el codigo de error y el
        ;identificador
        BANK1
        movf    Error_Code,W
        BANK0
        movwf   Out_Word0
        BANK1
        movf    TCP_Unit_identifier,W
        BANK0
        movwf   Out_Word1
```

```
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    checksum
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

                ;Tambien se agrega el codigo de excepción
                ;y un pad de cero
                BANK1
                movlw   0X00
                BANK0
                movwf   Out_Word0
                BANK1
                movf    Exception_Code,W
                BANK0
                movwf   Out_Word1

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    checksum
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

EXIT_LOAD_ERROR_FRAME

;Se guarda el checksum de los datos
BANK0
movf    Chksum0,W
BANK1
movwf   tcp_Chksum0
BANK0
movf    Chksum1,W
BANK1
movwf   tcp_Chksum1


bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    load_tcp_header
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;--------- Se empieza a cargar la cabecera MODBUS --------
;Se ensambla la cabera MODBUS
BANK1
movlw   NIC_DATA
movwf   Register_Address1

;Se carga el identificador de transacción MSB
movf    TCP_transaction_id1,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;Se carga el identificador de transacción LSB
movf    TCP_transaction_id0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;Se carga el identificador de protocolo (MODBUS =0) MSB
movf    TCP_protocol_id1,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;Se carga el identificador de protocolo (MODBUS =0) LSB
movf    TCP_protocol_id0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA

;Se carga  la longitud de los datos (MODBUS =0) MSB
movf    TCP_bytes_data1,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
```

```
;Se carga la longitud de los datos (MODBUS =0) LSB
movf        TCP_bytes_data0,W
movwf       Register_Value1
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        outport1
bcf                 PCLATH,4
bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Se carga el identificador de unidad
movf        TCP_Unit_identifier,W
movwf       Register_Value1
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        outport1
bcf                 PCLATH,4
bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

;Confirma si la respuesta sera debido a un error
movlw       0x01
andwf       Error_flag,W
btfss       STATUS,Z
goto        LOAD_ERROR_FRAME2
            ;Se carga el PDU MODBUS (de acuerdo a la petición realizada)
            ;Se carga el codigo de función
            movf        TCP_function_code,W
            movwf       Register_Value1
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        outport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

            ;Se carga el contador de bytes
            movf        TCP_quantity_inputs0,W
            movwf       Register_Value1
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        outport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
            goto        EXIT_LOAD_ERROR_FRAME2

LOAD_ERROR_FRAME2
            ;Se carga el codigo de error
            movf        Error_Code,W
            movwf       Register_Value1
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        outport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

            ;Se carga el codigo de excepción
            movf        Exception_Code,W
            movwf       Register_Value1
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        outport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

            ;Se carga un pad de cero
            movlw       0x00
            movwf       Register_Value1
            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        outport1
            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
            goto        EXIT_LOAD_ERROR_FRAME2

EXIT_LOAD_ERROR_FRAME2

;Establesco el tamaño base de cabeceras = 63
;14 bytes para cabecera Ethernet
;20 bytes para cabecera IP
;20 bytes para cabecera TCP
;9 bytes para cabecera MODBUS
;1 byte de pad para compensar la cabecera impar MODBUS
BANK0
clrf        Len_pack1
movlw       0X40                        ;64 en decimal
movwf       Len_pack0

;Se le suma la cantidad de datos MODBUS  a enviar
movf        Len_pack0,W
movwf       numero1_0
movf        Len_pack1,W
movwf       numero1_1
BANK1
```

```
movf        length_data_modbus,W
BANK0
movwf       numero2_0
movlw       0x00
movwf       numero2_1

bcf                 PCLATH,4
bcf                 PCLATH,3        ;PAGINA 0 DE PROGRAMA
call        sumar
bcf                 PCLATH,4
bsf                 PCLATH,3        ;PAGINA 1 DE PROGRAMA

movf        resultado0,W
movwf       Len_pack0
movf        resultado1,W
movwf       Len_pack1

bcf                 PCLATH,4
bcf                 PCLATH,3        ;PAGINA 0 DE PROGRAMA
call        TransmitLinkMgmt
bcf                 PCLATH,4
bsf                 PCLATH,3        ;PAGINA 1 DE PROGRAMA

;La cantidad de datos bytes enviados en esta trama
;9 bytes para cabecera TCP
BANK1
movlw       0x09                                ;9 en decimal
movwf       tcp_data_Len0
clrf        tcp_data_Len1

;Se le suma la cantidad de datos MODBUS  a enviar
movf        tcp_data_Len0,W
BANK0
movwf       numero1_0
BANK1
movf        tcp_data_Len1,W
BANK0
movwf       numero1_1
BANK1
movf        length_data_modbus,W
BANK0
movwf       numero2_0
movlw       0x00
movwf       numero2_1

bcf                 PCLATH,4
bcf                 PCLATH,3        ;PAGINA 0 DE PROGRAMA
call        sumar
bcf                 PCLATH,4
bsf                 PCLATH,3        ;PAGINA 1 DE PROGRAMA

movf        resultado0,W
BANK1
movwf       tcp_data_Len0
BANK0
movf        resultado1,W
BANK1
movwf       tcp_data_Len1

;se le suma la cantidad de datos enviados al numero
;de acuse de recivo
BANK1
movf        xm_seq1_1,W
BANK0
movwf       numero1_1
BANK1
movf        xm_seq1_0,W
BANK0
movwf       numero1_0

BANK1
movf        tcp_data_Len1,W
BANK0
movwf       numero2_1
BANK1
movf        tcp_data_Len0,W
BANK0
movwf       numero2_0

bcf                 PCLATH,4
bcf                 PCLATH,3        ;PAGINA 0 DE PROGRAMA
call        sumar
bcf                 PCLATH,4
bsf                 PCLATH,3        ;PAGINA 1 DE PROGRAMA

movf        resultado0,W
BANK1
movwf       xm_seq1_0
BANK0
movf        resultado1,W
BANK1
movwf       xm_seq1_1
```

```
;Se le suma uno a la segunda palabra del numero de secuencia
;si existio un desborde en la suma de la palabra LSB
BANK0
movf		CarryOut,W
sublw		0x01
btfss		STATUS,Z
goto		NOT_CARRY_OUT6

		;Existio un desborde al sumar el primer byte
		BANK1
		movf		xm_seq0_1,W
		BANK0
		movwf		numero1_1
		BANK1
		movf		xm_seq0_0,W
		BANK0
		movwf		numero1_0
		movlw		0x00
		movwf		numero2_1
		movlw		0x01
		movwf		numero2_0

		bcf				PCLATH,4	;se cambia a la pagina 0
		bcf				PCLATH,3
		call		sumar
		bcf				PCLATH,4	;se cambia a la pagina 1
		bsf				PCLATH,3

		movf		resultado0,W
		BANK1
		movwf		xm_seq0_0
		BANK0
		movf		resultado1,W
		BANK1
		movwf		xm_seq1_1
NOT_CARRY_OUT6

EXIT_AND_WAIT_T3.5
		;************** SE HABILITA LA  INTERRUPCION DEL TMR0 **************
		;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE ETHERNET
		BANK1
		bsf				INTCON,T0IE
		;************ SE HABILITAN LAS INTERRUPCIONES VIA SCI *************
		bsf				PIE1,RCIE
				;********** ESPERA A QUE FINALICE EL RETARDO DE 3.5 CARACTERES *********
				;Segun estandar se debe esperar a que finalice un retardo de 3.5
				;caracteres serial
				BANK0
				START_DELAY_t3.5
						btfss		PIR1,TMR1IF
						goto		START_DELAY_t3.5


EXIT_TREATMENT_MODBUS
;Se apaga el timer1
BANK0
bcf				T1CON,TMR1ON		;TMR1IF en OFF

;Se re-establecen las variables de control de errores
BANK1
clrf		MB_Header_error
clrf		MB_PDU_error
clrf		Exception_Code
clrf		Error_flag
clrf		Error_Code

;************** SE HABILITA LA  INTERRUPCION DEL TMR0 **************
;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE EHTERNET
BANK1
bsf				INTCON,T0IE
;************ SE HABILITAN LAS INTERRUPCIONES VIA SCI *************
bsf				PIE1,RCIE

;Se finaliza con el tratamiento a una petición MODBUS
BANK1
bcf				Request_MODBUS_present,0

return

;*********************************************************************
;*****************************************************************		*
;NOMBRE:		Timer1_To_1.5								*
	*
;CAPA:				4 -> APLICACION
	*		*
;DESCRIPCIÓN:		Crea un retardo de 2.5 caracteres serial		*		*
;						a 9600 baudios
		*		*
;*****************************************************************		*
Timer1_To_1.5
```

```
;*******************************************************************
'
            BANK0
            bcf             PIR1,TMR1IF                 ;limpia la bandera TMR1IF
            movlw   TIMER1.5_L          ;Se carga el registro del TMR para 3.5 caracteres
            movwf   TMR1L
            movlw   TIMER1.5_H
            movwf   TMR1H
            bsf             T1CON,TMR1ON        ;TMR1IF en ON
return


;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;$   CAMBIO A LA PAGINA 2 DEL PIC            $$
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
            ORG         0X1000


;*******************************************************************
'
;* INICIA EL TRATAMIENTO PARA ESCRITURA DE MULTIPLES REGISTROS  *
'*******************************************************************
'
TREATMENT_WRITE_MULTIPLE_REGISTER

                        BANK1
                        ;Datos recividos en una escritura de multiples registros
                        ;| ..... el byte anterior fue el codigo de función
                        ;|------------------------------|
                        ;|     Dirección de inicio                   | 2 bytes
                        ;|------------------------------|
                        ;|    Cantidad de registros  | 2 bytes (N registros ->120 Max)
                        ;|------------------------------|
                        ;|      Contador de bytes                    | 1 byte (2xN -> 240 Max)
                        ;|------------------------------|
                        ;|     Valor de los registros  | 2xN bytes
                        ;|------------------------------|

                        ;Se obtiene la DIRECCION DE INICIO
                        ;No se restringe un rango debido a que ésta es tarea del esclavo serial
                        bcf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA

                        call            inport1
                        bsf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA


                        movf            Register_Value1,W
                        movwf           TCP_starting_address1

                        bcf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA

                        call            inport1
                        bsf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA


                        movf            Register_Value1,W
                        movwf           TCP_starting_address0

                        ;Se obtiene la CANTIDAD DE REGISTROS
                        ;La maxima cantidad de registros a escribir es 120 (decimal)
                        bcf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA

                        call            inport1
                        bsf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA


                        movf            Register_Value1,W
                        BANK2
                        movwf           TCP_quantity_register1
                        BANK1

                        bcf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA

                        call            inport1
                        bsf                     PCLATH,4
                        bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA


                        movf            Register_Value1,W
                        BANK2
                        movwf           TCP_quantity_register0

                                ;La cantidad de registros a leer no debe ser mayor a 120/78
                                ;Se resta numero1 - numero2
                                ;Si numero1< numero2 entonces CarryOut se activa
                                movf            TCP_quantity_register1,W
                                BANK0
```

```
movwf   numero2_1
BANK2
movf    TCP_quantity_register0,W
BANK0
movwf   numero2_0
movlw   0x00
movwf   numero1_1
movlw   0x78
movwf   numero1_0

;Se resta numero1-numero2
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA

call    restar
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA


;Se establece si hubo un acarreo(numero1<numero2)

movf    CarryOut,W
sublw   0x01
btfss   STATUS,Z
goto    QUANTITY_INPUT_CORRECT#2;numero1>numero2 (no hay acarreo)

        BANK1
        ;Se activa la bandera por error en el PDU
        movlw   0x01
        movwf   MB_PDU_error

        ;Se establece el tipo de excepción ocurrido
        movlw   EXCEPTION_03    ;Error en la cantidad de datos
        movwf   Exception_Code

        goto    RESPONSE_WITH_ACK#2

QUANTITY_INPUT_CORRECT#2

;Se obtiene el CONTADOR DE BYTES
BANK1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA

call    inport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA


movf    Register_Value1,W
BANK2
movwf   TCP_count_bytes


;Transacción MODBUS aceptada
;Las cabeceras MODBUS estan bien formadas y se procede a tratar
;la petición

;//////////////////MODBUS_Service_Processing//////////////////
;/CAPA:          4 -> APLICACIÓN
/
;/MEDIO:         Ethernet
;/DESCRIPCIÓN:   Revisa la trama MODBUS entrante                                 /
;////////////////////////////////////////////////////////////

;Servicio de analisis de petición
;Aqui se decide si la petición se envia a la aplicación de usuario
;o se hara tratamiento local, en este caso siempre se hara tratamiento
;local

;///////////////Proceso del servicio Local//////////////////
;/CAPA:                  4 - > APLICACIÓN
/
;/MEDIO:         Ethernet
/
;/DESCRIPCIÓN:   La petición modbus se manejara localmente, lo   /
;/                               significa que no se hara el llamado a la        /
;/                               aplicación de usuario
/
;////////////////////////////////////////////////////////////

;SE TRANSFIERE EL CONTROL AL MEDIO SERIAL PARA EN TRATAMIENTO DE LA
;PETICIÓN MODBUS

;////////////////////request_to_slaveMODBUS//////////////////
;/CAPA:          4 -> APLICACIÓN
/
;/MEDIO:         Serial
        /
;/DESCRIPCIÓN:   Envia una petición al esclavo MODBUS                    /
;////////////////////////////////////////////////////////////
```

```
; Emisión petición hacia el esclavo MODBUS via serial (SCI)
;|-------------------------|
;|   Dirección de esclavo    | 1 bytes
;|-------------------------|
;|    Codigo de función                    | 1 bytes
;|-------------------------|
;|    Dirección de inicio      | 2 bytes
;|-------------------------|
;|   Cantidad de registros   | 2 bytes (N bytes)
;|-------------------------|
;|     Contador de bytes     | 1 byte (2xN bytes)
;|-------------------------|
;|  Valor de los registros    | 2xN bytes
;|-------------------------|
;|         CRC                                          | 2 bytes
;|-------------------------|

;******** SE INICIA EL PROCESO DE TRANSMISIÓN DE LA TRAMA *********

;Se inicializa el CRC con 0xFFFF
BANK1
movlw       0XFF
movwf       CRC_LSB
movwf       CRC_MSB

;Se envia el IDENTIFICADOR DE UNIDAD (via serial SCI)
;tambien llamada dirección de esclavo
movf        TCP_Unit_identifier,W
BANK0
bcf                 PCLATH,4
bcf                 PCLATH,3            ;Pagina 0 del programa
call        RS232_Tx
bsf                 PCLATH,4
bcf                 PCLATH,3            ;Pagina 2 del programa
BANK1

            ;Se calcula el CRC del dato enviado
            movf        TCP_Unit_identifier,W
            movwf       Data_Modbus_Serial

            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;Pagina 1 del programa
            call        Generate_CRC
            bsf                 PCLATH,4
            bcf                 PCLATH,3            ;Pagina 2 del programa

;Se envia el CODIGO DE FUNCIÓN (via serial SCI)
movf        TCP_function_code,W
BANK0
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        RS232_Tx
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA
BANK1

            ;Se calcula el CRC del dato enviado
            movf        TCP_function_code,W
            movwf       Data_Modbus_Serial

            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;Pagina 1 del programa
            call        Generate_CRC
            bsf                 PCLATH,4
            bcf                 PCLATH,3            ;Pagina 2 del programa

;Se envia la DIRECCIÓN DE INICIO (via serial SCI) MSB
movf        TCP_starting_address1,W
BANK0
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        RS232_Tx
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA
BANK1

            ;Se calcula el CRC del dato enviado
            movf        TCP_starting_address1,W
            movwf       Data_Modbus_Serial

            bcf                 PCLATH,4
            bsf                 PCLATH,3            ;Pagina 1 del programa
            call        Generate_CRC
            bsf                 PCLATH,4
            bcf                 PCLATH,3            ;Pagina 2 del programa

;Se envia la DIRECCIÓN DE INICIO (via serial SCI) LSB
movf        TCP_starting_address0,W
BANK0
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        RS232_Tx
```

```
bsf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA
BANK1

                ;Se calcula el CRC del dato enviado
                movf    TCP_starting_address0,W
                movwf   Data_Modbus_Serial

                bcf                     PCLATH,4
                bsf                     PCLATH,3        ;Pagina 1 del programa
                call    Generate_CRC
                bsf                     PCLATH,4
                bcf                     PCLATH,3        ;Pagina 2 del programa

;Se envia la CANTIDAD DE REGISTROS (via serial SCI) MSB
BANK2
movf    TCP_quantity_register1,W
BANK0
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    RS232_Tx
bsf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA
BANK2

                ;Se calcula el CRC del dato enviado
                movf    TCP_quantity_register1,W
                BANK1
                movwf   Data_Modbus_Serial

                bcf                     PCLATH,4
                bsf                     PCLATH,3        ;Pagina 1 del programa
                call    Generate_CRC
                bsf                     PCLATH,4
                bcf                     PCLATH,3        ;Pagina 2 del programa


;Se envia la CANTIDAD DE REGISTROS (via serial SCI) LSB
BANK2
movf    TCP_quantity_register0,W
BANK0
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    RS232_Tx
bsf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA
BANK2

                ;Se calcula el CRC del dato enviado
                movf    TCP_quantity_register0,W
                BANK1
                movwf   Data_Modbus_Serial

                bcf                     PCLATH,4
                bsf                     PCLATH,3        ;PAGINA 1 DE PROGRAMA
                call    Generate_CRC
                bsf                     PCLATH,4
                bcf                     PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se envia el CONTADOR DE BYTES
BANK2
movf    TCP_count_bytes,W
BANK0
bcf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 0 DE PROGRAMA
call    RS232_Tx
bsf                     PCLATH,4
bcf                     PCLATH,3                ;PAGINA 2 DE PROGRAMA
BANK2

                ;Se calcula el CRC del dato enviado
                movf    TCP_count_bytes,W
                BANK1
                movwf   Data_Modbus_Serial

                bcf                     PCLATH,4
                bsf                     PCLATH,3        ;PAGINA 1 DE PROGRAMA
                call    Generate_CRC
                bsf                     PCLATH,4
                bcf                     PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Ahora se empieza con la transmisión de los datos
BANK2
clrf    CounterExt

TRANSMIT_REGISTER_TO_WRITE

                ;Se obtiene un byte desde el buffer de anillo del RTL
                ;y se envia serialmente, ademas se le calcula el CRC
                BANK1
                bcf                     PCLATH,4
```

```
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA

                call    inport1
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA


                BANK2
                movwf   TCP_Byte_to_Write

                ;Se transmite el dato almacenado en W
                BANK0
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    RS232_Tx
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se calcula el CRC del dato enviado
                BANK2
                movf    TCP_Byte_to_Write,W
                BANK1
                movwf   Data_Modbus_Serial

                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
                call    Generate_CRC
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se verifica si se ha completado la transmisión de todos
                ;los registros a escribir
                BANK2
                incf    CounterExt,F
                movf    CounterExt,W
                subwf   TCP_count_bytes,W
                btfss   STATUS,Z
                goto    TRANSMIT_REGISTER_TO_WRITE

;Se ha completado la transmisión de la trama basica y se
;procede a transmitir el CRC, primero el LSB y luego el MSB
BANK1
movf    CRC_LSB,W
BANK0
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    RS232_Tx
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA
BANK1

movf    CRC_MSB,W
BANK0
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    RS232_Tx
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se ha completado la transmisión de datos hacia el esclavo MODBUS
;*****************INICIALIZA TIMER A 3.5 CARACTERES *****************
;Se inicializa un TIMER de 3.5 caracteres (tiempo entre paquetes SCI)
BANK0
bcf             PIR1,TMR1IF             ;limpia la bandera TMR1IF
movlw   TIMER3.5_L      ;Se carga el registro del TMR para 3.5 caracteres
movwf   TMR1L
movlw   TIMER3.5_H
movwf   TMR1H
bsf             T1CON,TMR1ON    ;TMR1IF en ON

;Se ha completado la transmisión de datos hacia el esclavo MODBUS
;********** SE CONSTESTA CON UN ACK AL PAQUETE RECIVIDO *********
;Ha finalizado el tratamiento a la trama que genero la
;petición MODBUS
RESPONSE_WITH_ACK#2

;Se activan las banderas correspondientes a un ACK
BANK1
movlw   0x00
movwf   data_flags0
bsf             data_flags0,4           ;Se habilita la bandera TCP_ACK

;Se procede a enviar un ACK del paquete recivido
bcf             PCLATH,4        ;se cambia a la pagina 1
bsf             PCLATH,3
call    socket_control
bsf             PCLATH,4        ;se cambia a la pagina 2
bcf             PCLATH,3

;********** SE CIERRA EL PAQUETE EN TRATAMIENTO **********
;Petición MODBUS aceptada
;Ahora se procede a cerrar el tratamiento al paquete que
```

```
;genero esta petición
bcf                     Request_MODBUS_present,2

;Se establece la pagina 0 en el RTL8019AS
BANK0
movlw   CR
movwf   Register_Address
movlw   0x22
movwf   Register_Value
bcf                     PCLATH,4
bcf                     PCLATH,3        ;PAGINA 0 DE PROGRAMA

call    outport
bsf                     PCLATH,4
bcf                     PCLATH,3        ;PAGINA 2 DE PROGRAMA


movlw   BNDRY
movwf   Register_Address
movf    Next_Ptr_Pck,W
movwf   Register_Value
bcf                     PCLATH,4
bcf                     PCLATH,3        ;PAGINA 0 DE PROGRAMA

call    outport
bsf                     PCLATH,4
bcf                     PCLATH,3        ;PAGINA 2 DE PROGRAMA


;**********************MANEJO DE ERRORES EN CABECERAS **********************
;Si existio un error en la recepción del PDU (a partir del codigo de función),
;se envia un mensaje de error
BANK1
movlw   0x01
andwf   MB_PDU_error,W
btfss   STATUS,Z
goto    ERROR_IN_SLAVE#2        ;Enviar paquete de notificación de error

;************** SE HABILITA LA  INTERRUPCION DEL TMR0 **************
;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE EHTERNET
BANK1
bsf                     INTCON,T0IE
;************ SE HABILITAN LAS INTERRUPCIONES VIA SCI **************
bsf                     PIE1,RCIE

        ;**** ESPERA FINALIZACIÓN DE TIEMPO DE 3.5 CARACTERES ******
        ;Se espera a que finalize un tiempo de 3.5 caracteres
        ;Este tiempo es el minimo entre tramas
        BANK0
        WAIT_t3.5#2
                btfss   PIR1,TMR1IF
                goto    WAIT_t3.5#2

        ;Se apaga el timer1
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

;************ SE DESABILITAN LAS INTERRUPCIONES VIA SCI **************
BANK1
bcf                     PIE1,RCIE

;//////////Espera por la contestación del esclavo MODBUS///////////////

;/CAPA:                         4 -> APLICACION
                                /
;/MEDIO:                 Serial
                                /
;/DESCRIPCIÓN:           Espera a que el esclavo MODBUS responda a la petición           /
;/                                       enviada. Inicia el reloj para el tiempo de espera                       /
;////////////////////////////////////////////////////////////////////

;Se inicializa el CRC con 0xFFFF
;con el fin de calcular el CRC de la trama recivida serialmente
BANK1
movlw   0XFF
movwf   CRC_LSB
movwf   CRC_MSB

;Se activa el tiempo de espera para la respuesta del esclavo
;Esto es requerido para no esperara indefinidamente por la respuesta

;Se establece a cero el contador del lazo externo
BANK2
clrf    CounterExt

RESPONSE_TIME_OUT2#2
        ;Se inicializa a cero el contador de rebasos del TIMER1
        BANK0
        clrf    Counter

        RESPONSE_TIME_OUT1#2
                ;Se inicializa el timer 1 (13.107ms por cada rebasamiento)
                BANK0
```

```
                                    bcf             PIR1,TMR1IF                 ;limpia la bandera TMR1IF
                                    movlw   0X00
                                    movwf   TMR1L
                                    movlw   0X00
                                    movwf   TMR1H
                                    bsf             T1CON,TMR1ON        ;TMR1IF en ON

                                    ;Se espera hasta que se de el rebaso del TIMER1 (de FFFF a 0000 )
                                    ;o se reciva el primer byte serial desde el medidor
                                    BANK0
                                    WAIT_OVERFLOW_TIMER1#2
                                            btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                                            goto    FIRST_BYTE_HEADER#2
                                            btfss   PIR1,TMR1IF         ;Revisa si el TIMER1 se ha desbordado
                                            goto    WAIT_OVERFLOW_TIMER1#2

                                    ;Se apaga el timer1
                                    bcf             T1CON,TMR1ON        ;TMR1IF en OFF

                                    ;Se ha producido un rebaso y por lo tanto se debe incrementar el contador
                                    incf    Counter,F

                                    ;Se revisa si el numero de rebasos del timer 1 ha sido completado
                                    movf    Counter,W
                                    sublw   MAX_OVERFLOW_TIMER1_1
                                    btfss   STATUS,Z
                                    goto    RESPONSE_TIME_OUT1#2

            ;Se ha finalizado el contador de rebasos internos
            ;Ahora se procede al lazo externo
            ;Se incrementa el contador externo
            BANK2
            incf    CounterExt,F

            ;Se revisa si el numero de rebasos se ha superado
            movf    CounterExt,W
            sublw   MAX_OVERFLOW_TIMER1_2
            btfss   STATUS,Z
            goto    RESPONSE_TIME_OUT2#2

                    ;Se establece el codigo de exepción
                    BANK1
                    movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                    movwf   Exception_Code

                    goto    ERROR_IN_SLAVE#2            ;Se ha superado el tiempo de espera

;//////////////////////Recepción de la contestación//////////////////////
;/CAPA:                     4 -> APLICACION
                            /
;/MEDIO:            Serial
                            /
;/DESCRIPCIÓN:      Empieza la recepción de datos desde el dispositivo          /
;/                                   esclavo  serial MODBUS
                    /
;////////////////////////////////////////////////////////////////////

;Ha llegado el primer byte serial, que forma parte de la cabecera MODBUS serial
;|----------------------|
;| Dirección del esclavo |   1 byte
;|----------------------|
;|   Codigo de Función   |   1 byte
;|----------------------|
;| Dirección de inicio       |       1 byte
;|----------------------|
;|Cantidad de registros      | 0 hasta 251 bytes
;|----------------------|
;|                           CRC                                   |       2 bytes
;|----------------------|


;Recepción de la cabecera MODBUS serial desde el esclavo (MEDIDOR DE ENERGIA)
FIRST_BYTE_HEADER#2
;********** SE DESABILITA LA INTERRUPCION DEL TMR0 **********
;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE ETHERNET
BANK1
bcf                 INTCON,T0IE

;---------- Se captura la dirección del esclavo ---------
BANK0
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        RS232_Rx
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA
BANK1
movwf       sci_data0           ;Se almacena el dato obtenido serialmente

;Si fue enviado por el esclavo esperado se acepta, de lo contrario
;se ignora y se continua con la espera (notar que el timer1 no ha sido detenido)
movf        TCP_Unit_identifier,W
subwf       sci_data0,W
```

```
btfsc	STATUS,Z
goto	SLAVE_CORRECT#2	;Si no es del esclavo esperado se ignora

		;************** SE HABILITA LA INTERRUPCION DEL TMR0 **************
		;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE ETHERNET
		bsf		INTCON,T0IE
		BANK0
		goto	WAIT_OVERFLOW_TIMER1#2	;Si no es del esclavo esperado se ignora


SLAVE_CORRECT#2
;El dato recivido es del esclavo que se esperaba, por lo tanto se inicia
;el proceso para la recepción de la trama

;Se calcula el CRC del dato recivido
BANK1
movf	sci_data0,W
movwf	Data_Modbus_Serial
bcf		PCLATH,4
bsf		PCLATH,3		;PAGINA 1 DE PROGRAMA
call	Generate_CRC
bsf		PCLATH,4
bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

;Se apaga el timer1
BANK0
bcf		T1CON,TMR1ON	;TMR1IF en OFF

;Se deben iniciar el timer1 a t1.5 (espera 1.5 caracteres antes de finalizar la
;recepción del paquete MODBUS del esclavo)
bcf		PCLATH,4
bsf		PCLATH,3		;PAGINA 1 DE PROGRAMA
call	Timer1_To_1.5
bsf		PCLATH,4
bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

;---------- Ahora se espera recivir el codigo de función o de error ---------
WAIT_RCV_FUNC_CODE#2
		btfsc	PIR1,RCIF	;Revisa si ha llegado un byte serial
		goto	FUNC_CODE_RCV#2
		btfss	PIR1,TMR1IF		;Revisa si el TIMER1 se ha desbordado
		goto	WAIT_RCV_FUNC_CODE#2

		;El tiempo de 1.5 caracteres se ha superado
		;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

		;Se apaga el timer1
		BANK0
		bcf		T1CON,TMR1ON	;TMR1IF en OFF

		BANK1
		movlw	EXCEPTION_04	;Error por falla en el dispositivo esclavo
		movwf	Exception_Code
		goto	ERROR_IN_SLAVE#2

;Se ha recivido el byte de codigo de función o de error
FUNC_CODE_RCV#2
		;Se apaga el timer1
		BANK0
		bcf		T1CON,TMR1ON	;TMR1IF en OFF

		;Se captura el codigo de función
		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	RS232_Rx
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA
		BANK1
		movwf	sci_data0

		;Se calcula el CRC del dato recivido
		movwf	Data_Modbus_Serial
		bcf		PCLATH,4
		bsf		PCLATH,3		;PAGINA 1 DE PROGRAMA
		call	Generate_CRC
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA


		;Se compara con el codigo de función esperado
		movf	TCP_function_code,W
		subwf	sci_data0,W
		btfsc	STATUS,Z
		goto	RCV_START_ADDR1#2

		;Se compara con el codigo de error esperado
		movlw	0x90
		subwf	sci_data0,W
		btfsc	STATUS,Z
		goto	RCV_EXCEPTION_CODE#2

				;Si no es ninguno de estos codigo, existe un error
```

```
                movlw   EXCEPTION_04    ;Error por codigo de función no esperado
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;--------- SE HA RECIVIDO UNA EXCEPCIÓN AL PAQUETE ENVIADO ----------
;Se espera recivir el codigo de exepción
RCV_EXCEPTION_CODE#2
;Se deben iniciar el timer1 a 1.5 caracteres
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

WAIT_RCV_EXCEPTION_CODE#2
        btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
        goto    EXEPTION_CODE_RCV#2
        btfss   PIR1,TMR1IF     ;Revisa si el TIMER1 se ha desbordado
        goto    WAIT_RCV_EXCEPTION_CODE#2

        ;El tiempo de 1.5 caracteres se ha superado
        ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

        BANK1
        movlw   EXCEPTION_04    ;Error por falla en el dispositivo esclavo
        movwf   Exception_Code
        goto    ERROR_IN_SLAVE#2

;Se ha recivido el byte de codigo de excepción
EXEPTION_CODE_RCV#2
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

        ;Se captura el codigo excepcion
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA
        BANK1
        movwf   sci_data0

        ;Se almacena el codigo de excepción
        movwf   sci_data0
        movwf   Exception_Code

        ;--------------------  Se captura el CRC ----------------------
        ;Se deben iniciar el timer1 a 1.5 caracteres
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
        call    Timer1_To_1.5
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        WAIT_RCV_CRC_ERROR#2
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    RCV_CRC_ERROR1#2
                btfss   PIR1,TMR1IF     ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_CRC_ERROR#2

                goto    ERROR_IN_SLAVE#2

        RCV_CRC_ERROR1#2
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON    ;TMR1IF en OFF

        ;Se captura el CRC LSB
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        ;Se deben iniciar el timer1 a 1.5 caracteres
        bcf             PCLATH,4
        bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
        call    Timer1_To_1.5
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        WAIT_RCV_CRC_ERROR2#2
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    RCV_CRC_ERROR2#2
                btfss   PIR1,TMR1IF     ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_CRC_ERROR2#2
```

```
                goto        ERROR_IN_SLAVE#2

        RCV_CRC_ERROR2#2
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON        ;TMR1IF en OFF

        ;Se captura el CRC MSB
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bsf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

        goto    ERROR_IN_SLAVE#2


;---------- RECEPCIÓN DE LA DIRECCIÓN DE INICIO ----------
;Es la cantidad de bytes en los datos
RCV_START_ADDR1#2
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

WAIT_RCV_START_ADDR1#2
        btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
        goto    START_ADDR_RCV1#2
        btfss   PIR1,TMR1IF         ;Revisa si el TIMER1 se ha desbordado
        goto    WAIT_RCV_START_ADDR1#2

        ;El tiempo de 1.5 caracteres se ha superado
        ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON        ;TMR1IF en OFF

        BANK1
        movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
        movwf   Exception_Code
        goto    ERROR_IN_SLAVE#2

;Se ha recivido el byte de dirección de inicio MSB
START_ADDR_RCV1#2
        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON        ;TMR1IF en OFF

        ;Se captura la dirección de inicio MSB
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        call    RS232_Rx
        bsf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA
        BANK1
        movwf   sci_data0

        ;Se calcula el CRC del dato recivido
        movwf   Data_Modbus_Serial
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
        call    Generate_CRC
        bsf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

        ;Se compara con la dirección de inicio esperada
        movf    TCP_starting_address1,W
        subwf   sci_data0,W
        btfsc   STATUS,Z
        goto    RCV_START_ADDR2#2

                movlw   EXCEPTION_04        ;Error por codigo de función no esperado
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;Ha llegado el segundo byte de la dirección (LSB)
RCV_START_ADDR2#2
;Se inicia el timer1 a 1.5 caraceres
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

WAIT_RCV_START_ADDR2#2
        btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
        goto    START_ADDR_RCV2#2
        btfss   PIR1,TMR1IF         ;Revisa si el TIMER1 se ha desbordado
        goto    WAIT_RCV_START_ADDR2#2
```

```
                ;El tiempo de 1.5 caracteres se ha superado
                ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON        ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;Se ha recivido el segundo byte de dirección de inicio LSB
START_ADDR_RCV2#2
                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se captura la dirección de inicio MSB
                bcf             PCLATH,4
                bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bsf             PCLATH,4
                bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA
                BANK1
                movwf   sci_data0

                ;Se calcula el CRC del dato recivido
                movwf   Data_Modbus_Serial
                bcf             PCLATH,4
                bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
                call    Generate_CRC
                bsf             PCLATH,4
                bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

                ;Se compara con la dirección de inicio esperada
                movf    TCP_starting_address0,W
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    CONT_WITH_QUANTITY_REG1#2

                        movlw   EXCEPTION_04        ;Error por codigo de función no esperado
                        movwf   Exception_Code
                        goto    ERROR_IN_SLAVE#2


;Se prepara para recivir la cantidad de registros MSB
CONT_WITH_QUANTITY_REG1#2
;Se inicia el timer1 a 1.5 caraceres
bcf             PCLATH,4
bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf             PCLATH,4
bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

WAIT_RCV_QUANTITY_REG1#2
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    START_QUANTITY_REG1#2
                btfss   PIR1,TMR1IF         ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_QUANTITY_REG1#2

                ;El tiempo de 1.5 caracteres se ha superado
                ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON        ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;Se ha recivido el byte de dirección de inicio MSB
START_QUANTITY_REG1#2
                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se captura la cantidad de registros MSB
                bcf             PCLATH,4
                bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bsf             PCLATH,4
                bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA
                BANK1
                movwf   sci_data0

                ;Se calcula el CRC del dato recivido
                movwf   Data_Modbus_Serial
                bcf             PCLATH,4
                bsf             PCLATH,3            ;PAGINA 1 DE PROGRAMA
```

```
                call    Generate_CRC
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se compara con la cantidad de registros esperada MSB
                BANK2
                movf    TCP_quantity_register1,W
                BANK1
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    CONT_WITH_QUANTITY_REG2#2

                        movlw   EXCEPTION_04    ;Error por codigo de función no esperado
                        movwf   Exception_Code
                        goto    ERROR_IN_SLAVE#2

;Continua con la recepción de la cantidad de registros
CONT_WITH_QUANTITY_REG2#2
;Se inicia el timer1 a 1.5 caraceres
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

WAIT_RCV_QUANTITY_REG2#2
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    START_QUANTITY_REG2#2
                btfss   PIR1,TMR1IF     ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_QUANTITY_REG2#2

                ;El tiempo de 1.5 caracteres se ha superado
                ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04    ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;Se ha recivido el byte de dirección de inicio MSB
START_QUANTITY_REG2#2
                ;Se apaga el timer1
                BANK0
                bcf             T1CON,TMR1ON    ;TMR1IF en OFF

                ;Se captura la cantidad de registros MSB
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA
                BANK1
                movwf   sci_data0

                ;Se calcula el CRC del dato recivido
                movwf   Data_Modbus_Serial
                bcf             PCLATH,4
                bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
                call    Generate_CRC
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se compara con la cantidad de registros esperada MSB
                BANK2
                movf    TCP_quantity_register0,W
                BANK1
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    CONTINUE_RECEPTION_CRC1#2

                        movlw   EXCEPTION_04    ;Error por codigo de función no esperado
                        movwf   Exception_Code
                        goto    ERROR_IN_SLAVE#2


;----------------------- RECEPCIÓN DEL CRC ------------------------------
CONTINUE_RECEPTION_CRC1#2

;Se inicia el timer1 a 1.5 caraceres
bcf             PCLATH,4
bsf             PCLATH,3        ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

WAIT_RCV_CRC1#2
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    START_CRC1#2
```

```
                btfss   PIR1,TMR1IF              ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_CRC1#2

                ;El tiempo de 1.5 caracteres se ha superado
                ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

                ;Se apaga el timer1
                BANK0
                bcf                     T1CON,TMR1ON        ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;Se ha recivido el primer byte del CRC (LSB)
START_CRC1#2
                ;Se apaga el timer1
                BANK0
                bcf                     T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se captura CRC LSB
                bcf                     PCLATH,4
                bcf                     PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bsf                     PCLATH,4
                bcf                     PCLATH,3            ;PAGINA 2 DE PROGRAMA
                BANK1
                movwf   sci_data0

                ;Se compara con la cantidad de registros esperada LSB
                movf    CRC_LSB,W
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    CONTINUE_RECEPTION_CRC2#2

                        movlw   EXCEPTION_04        ;Error por codigo de función no esperado
                        movwf   Exception_Code
                        goto    ERROR_IN_SLAVE#2


;Se inicia la espera por el CRC
CONTINUE_RECEPTION_CRC2#2

;Se inicia el timer1 a 1.5 caraceres
bcf                     PCLATH,4
bsf                     PCLATH,3            ;PAGINA 1 DE PROGRAMA
call    Timer1_To_1.5
bsf                     PCLATH,4
bcf                     PCLATH,3            ;PAGINA 2 DE PROGRAMA

WAIT_RCV_CRC2#2
                btfsc   PIR1,RCIF   ;Revisa si ha llegado un byte serial
                goto    START_CRC2#2
                btfss   PIR1,TMR1IF             ;Revisa si el TIMER1 se ha desbordado
                goto    WAIT_RCV_CRC2#2

                ;El tiempo de 1.5 caracteres se ha superado
                ;Esto corresponde a un error con el dispositivo esclavo MODBUS (serial)

                ;Se apaga el timer1
                BANK0
                bcf                     T1CON,TMR1ON        ;TMR1IF en OFF

                BANK1
                movlw   EXCEPTION_04        ;Error por falla en el dispositivo esclavo
                movwf   Exception_Code
                goto    ERROR_IN_SLAVE#2

;Se ha recivido el primer byte del CRC
START_CRC2#2
                ;Se apaga el timer1
                BANK0
                bcf                     T1CON,TMR1ON        ;TMR1IF en OFF

                ;Se captura CRC MSB
                bcf                     PCLATH,4
                bcf                     PCLATH,3            ;PAGINA 0 DE PROGRAMA
                call    RS232_Rx
                bsf                     PCLATH,4
                bcf                     PCLATH,3            ;PAGINA 2 DE PROGRAMA
                BANK1
                movwf   sci_data0

                ;Se compara con la cantidad de registros esperada MSB
                movf    CRC_MSB,W
                subwf   sci_data0,W
                btfsc   STATUS,Z
                goto    SEND_RESPONSE#2

                        movlw   EXCEPTION_04        ;Error por codigo de función no esperado
                        movwf   Exception_Code
```

```
                goto        ERROR_IN_SLAVE#2

;*************** CONSTRUIR EXCEPCIÓN MODBUS DE RESPUESTA ***************
;
;* CAPA:          4 -> APLICACIÓN                                          *
;                 *
;* DESCRIPCIÓN    Define el tipo de excepción generada en el esclavo       *
;*********************************************************************
;
;Se define el tipo de error ocurrido y se envia una respuesta
ERROR_IN_SLAVE#2

        ;************** SE DESABILITA LA INTERRUPCION DEL TMR0 **************
        ;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE EHTERNET
        BANK1
        bcf             INTCON,T0IE

        ;Se apaga el timer1
        BANK0
        bcf             T1CON,TMR1ON        ;TMR1IF en OFF

        BANK1
        ;Se activa la bandera de error
        movlw   0x01
        movwf   Error_flag

        ;Se establece el tipo de error ocurrido
        movlw   0x80
        movwf   Error_Code
        movf    TCP_function_code,W
        addwf   Error_Code,F

;************************CONSTRUIR RESPUESTA MODBUS************************
;
;* CAPA:          4 -> APLICACIÓN                                          *
;                 *
;* DESCRIPCIÓN    Contruye el paquete de respuesta a la petición MODBUS    *
;*********************************************************************
;
SEND_RESPONSE#2

;****************** RETARDO DE 3.5 CARACTERES SERIAL **********************
;Previo a la transmisión de paquete MODBUS via serial se genera un
;retardo de 3.5 caracteres
;Programación del TMR1 con un tiempo de 3.5 caracteres
BANK0
bcf             PIR1,TMR1IF                                 ;limpia la bandera TMR1IF
movlw   TIMER3.5_L                      ;Se carga el registro del TMR para 3.5  caracteres
movwf   TMR1L
movlw   TIMER3.5_H
movwf   TMR1H
bsf             T1CON,TMR1ON        ;TMR1IF en ON

        ;Ahora es importante conocer el estado del socket antes de responder
        ;Ya que exise la posibilidad de que ahiga sido cerrado miestras se esperaba por la
        ;repuesta del esclavo serial MODBUS
        BANK1
        movlw   ESTAB
        subwf   html_socket,W
        btfsc   STATUS,Z
        goto    CONT_WITH_SEND_FRAME#2 ;Verificar si el estado es FIN
                goto    EXIT_AND_WAIT_T3.5#2

        CONT_WITH_SEND_FRAME#2
        ;Se establece a cero el checksum para las cabeceras
        BANK1
        clrf    tcp_Chksum0
        clrf    tcp_Chksum1
        BANK0
        clrf    Chksum0
        clrf    Chksum1

        ;Se carga la CABECERA ETHERNET en el buffer de anillo del RTL8019AS
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        call    load_ethernet_header
        bsf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 2 DE PROGRAMA

        ;Se carga la cabecera IP en el buffer de anillo del RTL, primero se
        ;establece el tamaño minimo requerido en las cabeceras:
        ;20 bytes para cabecera IP
;20 bytes para cabecera TCP
        ;12  bytes para cabecera MODBUS (7 de cabecera básica)
        ;(En el caso de error, son 9 para cabecera MODBUS)

        ;Confirma si la respuesta sera debido a un error
        BANK1
        movlw   0x01
        andwf   Error_flag,W
        btfss   STATUS,Z
        goto    ERROR_FRAME_IP#2
                ;Es una respuesta a una escritura sin problemas
                BANK0
                clrf    IP_packet_length1
                movlw   0X34                                ;52 (en decimal)
```

```
                movwf   IP_packet_length0
                goto    END_ERROR_FRAME_IP#2

ERROR_FRAME_IP#2
                ;Sera una respuesta a un error
                BANK0
                clrf    IP_packet_length1
                movlw   0X31                    ;49 (en decimal)
                movwf   IP_packet_length0
END_ERROR_FRAME_IP#2

                ;Se establece el protocolo dentro de la trama IP
                movlw   TCP
                movwf   IP_send_protocol

                ;Se carga la cabecera IP
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    load_IP_header
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;---------- Se carga la cabecera TCP ----------
                ;Se definen las banderas en la cabecera TCP
                BANK1
                movlw   0x50            ;Tamaño de la cabecera TCP (valor fijo)
                movwf   data_flags1
                movlw   0x00
                movwf   data_flags0
                bsf             data_flags0,4   ;Se habilita la bandera TCP_ACK
                bsf             data_flags0,3   ;Se habilita la bandera TCP_PUSH

                ;Se almacena el tamaño de la cabecera TCP y MODBUS
                ;Primero se establece el tamaño minimo para ambas cabeceras:
                ;20 bytes de cabecera TCP
                ;12  bytes de cabecera MODBUS
                ;(En el caso de error la cabecera MODBUS se reduce a 9 bytes)

                ;Confirma si la respuesta sera debido a un error
                BANK1
                movlw   0x01
                andwf   Error_flag,W
                btfss   STATUS,Z
                goto    ERROR_FRAME_TCP#2
                        ;Sera una respuesta normal a una escritura
                        clrf    tcp_length1
                        movlw   0X20                    ;32 en decimal
                        movwf   tcp_length0
                        goto    END_ERROR_FRAME_TCP#2

                ERROR_FRAME_TCP#2
                        ;Sera una respuesta a un error
                        clrf    tcp_length1
                        movlw   0X1D                    ;29 en decimal
                        movwf   tcp_length0
                END_ERROR_FRAME_TCP#2

                ;Con el proposito de cargar el checksum en la cabecera TCP
                ;se calcula el checksum de la cabecera MODBUS sin cargarla
                ;en el buffer del RTL
                BANK0
                clrf    Chksum0
                clrf    Chksum1

                ;Se agrega al checksum el NUMERO DE TRANSACCIÓN
                BANK1
                movf    TCP_transaction_id0,W
                BANK0
                movwf   Out_Word0
                BANK1
                movf    TCP_transaction_id1,W
                BANK0
                movwf   Out_Word1

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    checksum
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se agrega al checksum el IDENTIFICADOR DE PROTOCOLO
                BANK1
                movf    TCP_protocol_id0,W
                BANK0
                movwf   Out_Word0
                BANK1
                movf    TCP_protocol_id1,W
                BANK0
                movwf   Out_Word1

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
```

```
call        checksum
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA


;Se agrega al checksum la LONGITUD MODBUS
;Se establece la longitud de cabecera (obligatoria)
; 1 byte para el identificador de unidad
; 1 byte para el codigo de función
; 2 para la dirección de inicio
; 2 para la cantidad de registros
; (En el caso de un error solo son 3 bytes)

;Confirma si la respuesta sera debido a un error
BANK1
movlw       0x01
andwf       Error_flag,W
btfss       STATUS,Z
goto        ERROR_FRAME_TCP2#2
            ;Es una respuesta a una escritura sin problemas
            movlw       0x06
            movwf       TCP_bytes_data0
            clrf        TCP_bytes_data1
            goto        END_ERROR_FRAME_TCP2#2
ERROR_FRAME_TCP2#2
            ;Es una respuesta a un error
            movlw       0x03
            movwf       TCP_bytes_data0
            clrf        TCP_bytes_data1
END_ERROR_FRAME_TCP2#2

;Se procede a sumar al checksum
BANK1
movf        TCP_bytes_data0,W
BANK0
movwf       Out_Word0
BANK1
movf        TCP_bytes_data1,W
BANK0
movwf       Out_Word1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        checksum
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

;Confirma si la respuesta sera debido a un error
BANK1
movlw       0x01
andwf       Error_flag,W
btfss       STATUS,Z
goto        LOAD_ERROR_FRAME#2

            ;Se agrega al checksum el codigo de funcion e
            ;identificador
            movf        TCP_function_code,W
            BANK0
            movwf       Out_Word0
            BANK1
            movf        TCP_Unit_identifier,W
            BANK0
            movwf       Out_Word1

            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        checksum
            bsf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

            ;Se agrega al checksum la dirección de inicio
            BANK1
            movf        TCP_starting_address0,W
            BANK0
            movwf       Out_Word0
            BANK1
            movf        TCP_starting_address1,W
            BANK0
            movwf       Out_Word1

            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        checksum
            bsf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

            ;Se agrega al checksum la cantidad de registros
            BANK2
            movf        TCP_quantity_register0,W
            BANK0
            movwf       Out_Word0
            BANK2
```

```
            movf    TCP_quantity_register1,W
            BANK0
            movwf   Out_Word1

            bcf             PCLATH,4
            bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
            call    checksum
            bsf             PCLATH,4
            bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA
            goto    EXIT_LOAD_ERROR_FRAME#2

LOAD_ERROR_FRAME#2
            ;Se agrega al checksum el codigo de error y el
            ;identificador
            BANK1
            movf    Error_Code,W
            BANK0
            movwf   Out_Word0
            BANK1
            movf    TCP_Unit_identifier,W
            BANK0
            movwf   Out_Word1

            bcf             PCLATH,4
            bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
            call    checksum
            bsf             PCLATH,4
            bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

            ;Tambien se agrega el codigo de excepción
            ;y un pad de cero
            BANK1
            movlw   0X00
            BANK0
            movwf   Out_Word0
            BANK1
            movf    Exception_Code,W
            BANK0
            movwf   Out_Word1

            bcf             PCLATH,4
            bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
            call    checksum
            bsf             PCLATH,4
            bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

EXIT_LOAD_ERROR_FRAME#2

;Se guarda el checksum de los datos
BANK0
movf    Chksum0,W
BANK1
movwf   tcp_Chksum0
BANK0
movf    Chksum1,W
BANK1
movwf   tcp_Chksum1

bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    load_tcp_header
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;--------- Se empieza a cargar la cabecera MODBUS --------
;Se ensambla la cabera MODBUS
BANK1
movlw   NIC_DATA
movwf   Register_Address1

;Se carga el IDENTIFICADOR DE TRANSACCIÓN MSB
movf    TCP_transaction_id1,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se carga el IDENTIFICADOR DE TRANSACCIÓN LSB
movf    TCP_transaction_id0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se carga el IDENTIFICADOR DE PROTOCOLO (MODBUS =0) MSB
movf    TCP_protocol_id1,W
movwf   Register_Value1
bcf             PCLATH,4
```

```
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se carga el IDENTIFICADOR DE PROTOCOLO (MODBUS =0) LSB
movf    TCP_protocol_id0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se carga  la LONGITUD DE LOS DATOS (MODBUS =0) MSB
movf    TCP_bytes_data1,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se carga la LONGITUD DE LOS DATOS (MODBUS =0) LSB
movf    TCP_bytes_data0,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Se carga el IDENTIFICADOR DE UNIDAD
movf    TCP_Unit_identifier,W
movwf   Register_Value1
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    outport1
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;Confirma si la respuesta sera debido a un error
movlw   0x01
andwf   Error_flag,W
btfss   STATUS,Z
goto    LOAD_ERROR_FRAME2#2
        ;Se carga el PDU MODBUS (de acuerdo a la petición realizada)
        ;Se carga el CODIGO DE FUNCIÓN
        movf    TCP_function_code,W
        movwf   Register_Value1
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    outport1
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        ;Se carga la DIRECCIÓN DE INICIO MSB
        movf    TCP_starting_address1,W
        movwf   Register_Value1
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    outport1
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        ;Se carga la DIRECCIÓN DE INICIO LSB
        movf    TCP_starting_address0,W
        movwf   Register_Value1
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    outport1
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        ;Se carga la CANTIDAD DE REGISTROS MSB
        BANK2
        movf    TCP_quantity_register1,W
        BANK1
        movwf   Register_Value1
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        call    outport1
        bsf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

        ;Se carga la CANTIDAD DE REGISTROS LSB
        BANK2
        movf    TCP_quantity_register0,W
        BANK1
        movwf   Register_Value1
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
```

```
                call    outport1
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                goto    EXIT_LOAD_ERROR_FRAME2#2

LOAD_ERROR_FRAME2#2
                ;Se carga el codigo de error
                movf    Error_Code,W
                movwf   Register_Value1
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    outport1
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se carga el codigo de excepción
                movf    Exception_Code,W
                movwf   Register_Value1
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    outport1
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                ;Se carga un pad de cero
                movlw   0x00
                movwf   Register_Value1
                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    outport1
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA
                goto    EXIT_LOAD_ERROR_FRAME2#2

                EXIT_LOAD_ERROR_FRAME2#2

;Establesco el tamaño base de cabeceras = 63
;14 bytes para cabecera Ethernet
;20 bytes para cabecera IP
;20 bytes para cabecera TCP
;12 bytes para cabecera MODBUS
;(en el caso de error la cabecera MODBUS es de 9 bytes+1 de pad)

;Confirma si la respuesta sera debido a un error
BANK1
movlw   0x01
andwf   Error_flag,W
btfss   STATUS,Z
goto    ERROR_FRAME_ETHERNET#2
                ;Es una trama de respuesta a una escritura correcta
                BANK0
                clrf    Len_pack1
                movlw   0X42                    ;66 en decimal
                movwf   Len_pack0
                goto    END_ERROR_FRAME_ETHERNET#2
ERROR_FRAME_ETHERNET#2
                ;Ha existido un error en la escritura de los datos
                BANK0
                clrf    Len_pack1
                movlw   0X40                    ;64 en decimal
                movwf   Len_pack0
END_ERROR_FRAME_ETHERNET#2

bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    TransmitLinkMgmt
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;La cantidad de datos bytes enviados en esta trama
;12 bytes para cabecera TCP
;Confirma si la respuesta sera debido a un error
BANK1
movlw   0x01
andwf   Error_flag,W
btfss   STATUS,Z
goto    ERROR_FRAME_ETH#2
                ;La escritura de datos fue correcta
                movlw   0x0C                                    ;12 en decimal
                movwf   tcp_data_Len0
                clrf    tcp_data_Len1
                goto    END_ERROR_FRAME_ETH#2
ERROR_FRAME_ETH#2
                ;La escritura de datos fue correcta
                movlw   0x09                                    ;9 en decimal
                movwf   tcp_data_Len0
                clrf    tcp_data_Len1
                goto    END_ERROR_FRAME_ETH#2
END_ERROR_FRAME_ETH#2

;se le suma la cantidad de datos enviados al numero
```

```
                ;de acuse de recivo
                BANK1
                movf    xm_seq1_1,W
                BANK0
                movwf   numero1_1
                BANK1
                movf    xm_seq1_0,W
                BANK0
                movwf   numero1_0

                BANK1
                movf    tcp_data_Len1,W
                BANK0
                movwf   numero2_1
                BANK1
                movf    tcp_data_Len0,W
                BANK0
                movwf   numero2_0

                bcf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
                call    sumar
                bsf             PCLATH,4
                bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

                movf    resultado0,W
                BANK1
                movwf   xm_seq1_0
                BANK0
                movf    resultado1,W
                BANK1
                movwf   xm_seq1_1

                ;Se le suma uno a la segunda palabra del numero de secuencia
                ;si existio un desborde en la suma de la palabra LSB
                BANK0
                movf    CarryOut,W
                sublw   0x01
                btfss   STATUS,Z
                goto    NOT_CARRY_OUT6#2

                        ;Existio un desborde al sumar el primer byte
                        BANK1
                        movf    xm_seq0_1,W
                        BANK0
                        movwf   numero1_1
                        BANK1
                        movf    xm_seq0_0,W
                        BANK0
                        movwf   numero1_0
                        movlw   0x00
                        movwf   numero2_1
                        movlw   0x01
                        movwf   numero2_0

                        bcf             PCLATH,4    ;se cambia a la pagina 0
                        bcf             PCLATH,3
                        call    sumar
                        bsf             PCLATH,4    ;se cambia a la pagina 2
                        bcf             PCLATH,3

                        movf    resultado0,W
                        BANK1
                        movwf   xm_seq0_0
                        BANK0
                        movf    resultado1,W
                        BANK1
                        movwf   xm_seq1_1
                NOT_CARRY_OUT6#2

                EXIT_AND_WAIT_T3.5#2
                ;************** SE HABILITA LA  INTERRUPCION DEL TMR0 **************
                ;EL TIMER0 CONTROLA LAS INTERRUPCIÓNES POR LLEGADA DE PAQUETE ETHERNET
                BANK1
                bsf             INTCON,T0IE
                ;************* SE HABILITAN LAS INTERRUPCIONES VIA SCI ***************
                bsf             PIE1,RCIE

                        ;***** ESPERA A QUE FINALICE EL RETARDO DE 3.5 CARACTERES *******
                        ;Segun estandar se debe esperar a que finalice un retardo de 3.5
                        ;caracteres serial
                        BANK0
                        START_DELAY_t3.5#2
                                btfss   PIR1,TMR1IF
                                goto    START_DELAY_t3.5#2

        bcf             PCLATH,4    ;se cambia a la pagina 1
        bsf             PCLATH,3
goto    EXIT_TREATMENT_MODBUS
```

```
;*******************************************************************
;*****************************************************************             *
;NOMBRE:                tcp_listen                                                              *
                *
;CAPA:                                 4 -> TRANSPORTE
                *               *
;DESCRIPCIÓN:           Inicia el establecimiento del socket, envia un           *              *
;                                                     SYN,ACK     a al que envio la petición de iniciación*  *
;                                                     del socket
                                *               *
;ENTRADAS:                             Ninguna
                                *               *
;SALIDAS:               Envia un paquete SYN,ACK                                                *               *
;*****************************************************************             *
tcp_listen;
                                *               *
;*******************************************************************
                BANK1
                ;Almacena el numero de secuencia de la trama llegante en el numero
                ;de acuse de la trama que se transmitira
                movf            seq0_1,W
                movwf           xm_ack0_1

                movf            seq0_0,W
                movwf           xm_ack0_0

                movf            seq1_1,W
                movwf           xm_ack1_1

                movf            seq1_0,W
                movwf           xm_ack1_0

                ;Almacena el puerto de la trama llegante para propositos
                ;de control del socket
                movf            tcp_source_port1,W
                movwf           port1
                movf            tcp_source_port0,W
                movwf           port0

                ;Almacena la IP del emisor de  paquete para propositos
                ;de control del socket
                BANK0
                movf            Source_IP0,W
                BANK1
                movwf           ip0
                BANK0
                movf            Source_IP1,W
                BANK1
                movwf           ip1
                BANK0
                movf            Source_IP2,W
                BANK1
                movwf           ip2
                BANK0
                movf            Source_IP3,W
                BANK1
                movwf           ip3

                ;Calcula la longitud de los datos en la trama
                ;Para una petición SYN ordinaria deberia ser cero
                movf            offset,W
                BANK0
                movwf           numero2_0
                clrf            numero2_1
                movf            hdr_len,W
                movwf           numero1_0
                clrf            numero1_1

                bcf                             PCLATH,4
                bcf                             PCLATH,3                ;PAGINA 0 DE PROGRAMA
                call            sumar
                bsf                             PCLATH,4
                bcf                             PCLATH,3                ;PAGINA 2 DE PROGRAMA

                movf            resultado0,W
                movwf           numero2_0
                movf            resultado1,W
                movwf           numero2_1

                movf            IP_Length0,W
                movwf           numero1_0
                movf            IP_Length1,W
                movwf           numero1_1

                bcf                             PCLATH,4
                bcf                             PCLATH,3                ;PAGINA 0 DE PROGRAMA
                call            restar
                bsf                             PCLATH,4
                bcf                             PCLATH,3                ;PAGINA 2 DE PROGRAMA

                movf            resultado0,W
```

```
BANK1
movwf       tcp_data_Len0
BANK0
movf        resultado1,W
BANK1
movwf       tcp_data_Len1


;Se le suma el numero de datos al numero de acuse
;Ademas se le suma 1 debido a que se recivio un SYN
movlw       0X01
BANK0
movwf       numero1_0
clrf        numero1_1
BANK1
movf        tcp_data_Len0,W
BANK0
movwf       numero2_0
BANK1
movf        tcp_data_Len1,W
BANK0
movwf       numero2_1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        sumar
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movf        resultado0,W
movwf       numero1_0
movf        resultado1,W
movwf       numero1_1
BANK1
movf        xm_ack1_0,W
BANK0
movwf       numero2_0
BANK1
movf        xm_ack1_1,W
BANK0
movwf       numero2_1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        sumar
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movf        resultado0,W
BANK1
movwf       xm_ack1_0
BANK0
movf        resultado1,W
BANK1
movwf       xm_ack1_1

;Se le suma uno a la segunda palabra del ACK si existio
;un desborde en la suma de la palabra LSB
BANK0
movf        CarryOut,W
sublw       0x01
btfss       STATUS,Z
goto        NOT_CARRY_OUT1      ;numero1>numero2 (no hay acarreo)

            ;Existio un desborde, por lo cual se sumara a la palabra ACK MSB
            BANK1
            movf        xm_ack0_1,W
            BANK0
            movwf       numero1_1
            BANK1
            movf        xm_ack0_0,W
            BANK0
            movwf       numero1_0
            movlw       0x00
            movwf       numero2_1
            movlw       0x01
            movwf       numero2_0

            bcf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
            call        sumar
            bsf                 PCLATH,4
            bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

            movf        resultado0,W
            BANK1
            movwf       xm_ack0_0
            BANK0
            movf        resultado1,W
            BANK1
            movwf       xm_ack1_1
```

```
NOT_CARRY_OUT1

;Se actualiza el numero de secuencia, sumandole 1 debido
;a que se enviara un SYN
BANK1
movf    xm_seq1_1,W
BANK0
movwf   numero1_1
BANK1
movf    xm_seq1_0,W
BANK0
movwf   numero1_0
movlw   0x00
movwf   numero2_1
movlw   0x01
movwf   numero2_0

bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    sumar
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

movf    resultado0,W
BANK1
movwf   xm_seq1_0
BANK0
movf    resultado1,W
BANK1
movwf   xm_seq1_1

;Si el numero de secuencia LSB es cero, sumar 1 al MSB
;Esto indica un acarreo en la primera palabra LSB
;       movlw   0x00
;       subwf   xm_seq1_0,W
;       btfss   STATUS,Z
;       goto    EXIT_IF1
;       movlw   0x00
;       subwf   xm_seq1_1,W
;       btfss   STATUS,Z
;       goto    EXIT_IF1
;       movf    xm_seq0_1,W
;       BANK0
;       movwf   numero1_1
;       BANK1
;       movf    xm_seq0_0,W
;       BANK0
;       movwf   numero1_0
;       movlw   0x00
;       movwf   numero2_1
;       movlw   0x01
;       movwf   numero2_0

;       bcf             PCLATH,4
;       bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
;       call    sumar
;       bsf             PCLATH,4
;       bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA

;       movf    resultado0,W
;       BANK1
;       movwf   xm_seq0_0
;       BANK0
;       movf    resultado1,W
;       BANK1
;       movwf   xm_seq0_1

;EXIT_IF1
;////////////////////////??????????????????????*****************************************************************************************************

;Se carga la cabecera Ethernet en el buffer de anillo del RTL
BANK0
bcf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
call    load_ethernet_header
bsf             PCLATH,4
bcf             PCLATH,3        ;PAGINA 2 DE PROGRAMA


;Se define la longitud IP del paquete a enviar
;20 para cabecera IP, 20 para cabecera TCP y 8 de opciones
clrf    IP_packet_length1
movlw   0X30                                    ;48 en decimal
movwf   IP_packet_length0
;Se define el protocolo en la cabecera IP
movlw   TCP
movwf   IP_send_protocol

;Se define un nuevo numero de indentificación
movf    Identification0,W
movwf   numero1_0
movf    Identification1,W
```

```
movwf       numero1_1
movlw       0x01
movwf       numero2_0
movlw       0x00
movwf       numero2_1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        sumar
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movf        resultado0,W
movwf       Identification0
movf        resultado1,W
movwf       Identification1

;Se carga la cabecera IP
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        load_IP_header
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA


;Se establecen las banderas de la cabecera TCP
BANK1
movlw       0x70                ;Longitud de la cabecera TCP
movwf       data_flags1
movlw       0x00
movwf       data_flags0
bsf                 data_flags0,1       ;Se habilita la bandera TCP_SYN
bsf                 data_flags0,4       ;Se habilita la bandera TCP_ACK

;Se establece el valor del checksum de las opciones
movlw       0x07
movwf       tcp_Chksum1
movlw       0xB8
movwf       tcp_Chksum0

;Se establece la longitud de la cabecera TCP y datos
;20 para la cabecera y 8 para datos
clrf        tcp_length1
movlw       0X1C
movwf       tcp_length0

;Se carga la cabecera TCP y los datos en el buffer del RTL
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        load_tcp_header
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

;Se cargan las opciones en el buffer de anillo del RTL
BANK1
movlw       NIC_DATA
movwf       Register_Address1

movlw       0x02
movwf       Register_Value1

bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        outport1
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movlw       0x04
movwf       Register_Value1
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        outport1
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movlw       0x05
movwf       Register_Value1
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        outport1
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movlw       0xb4
movwf       Register_Value1
bcf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
call        outport1
bsf                 PCLATH,4
bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA

movlw       0x00
```

```
		movwf	Register_Value1
		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	outport1
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

		movlw	0x00
		movwf	Register_Value1
		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	outport1
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

		movlw	0x00
		movwf	Register_Value1
		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	outport1
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

		movlw	0x00
		movwf	Register_Value1
		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	outport1
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

		;Se envia el paquete previamente cargado en el buffer
		;del RTL
		BANK0
		clrf	Len_pack1
		movlw	0X3E			;Se envian 62 bytes
		movwf	Len_pack0

		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	TransmitLinkMgmt
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA
		BANK1

		;Actualiza el proximo numero de secuencia a enviar
		;sumandole uno, debido a que se envia un SYN
		movf	xm_seq1_1,W
		BANK0
		movwf	numero1_1
		BANK1
		movf	xm_seq1_0,W
		BANK0
		movwf	numero1_0
		movlw	0x00
		movwf	numero2_1
		movlw	0x01
		movwf	numero2_0

		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	sumar
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

		movf	resultado0,W
		BANK1
		movwf	xm_seq1_0
		BANK0
		movf	resultado1,W
		BANK1
		movwf	xm_seq1_1

;Si el numero de secuencia LSB es cero, sumar uno
		;al numero de secuencia MSB
		movlw	0x00
		subwf	xm_seq1_0,W
		btfss	STATUS,Z
		goto	EXIT_IF2
		movlw	0x00
		subwf	xm_seq1_1,W
		btfss	STATUS,Z
		goto	EXIT_IF2

		movf	xm_seq0_1,W
		BANK0
		movwf	numero1_1
		BANK1
		movf	xm_seq0_0,W
		BANK0
		movwf	numero1_0
		movlw	0x00
```

```
		movwf	numero2_1
		movlw	0x01
		movwf	numero2_0

		bcf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	sumar
		bsf		PCLATH,4
		bcf		PCLATH,3		;PAGINA 2 DE PROGRAMA

		movf	resultado0,W
		BANK1
		movwf	xm_seq0_0
		BANK0
		movf	resultado1,W
		BANK1
		movwf	xm_seq0_1

EXIT_IF2

		BANK1
		;El socket ha cambiado su estado a SYN_RCVD
		;queda en espera de un ACK
		movlw	SYN_RCVD
		movwf	html_socket

		return


;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;$   CAMBIO A LA PAGINA 3 DEL PIC			$$
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
		ORG		0X1800

;*********************************************************************
;*****************************************************************		*
;NOMBRE:		save_value_vars								*
		*
;CAPA:				3 - TRANSPORTE
		*	*
;DESCRIPCIÓN:	Se crea un respaldo de las variables fuera de	*	*
;						la interrupción
				*	*
;*****************************************************************		*
save_value_vars;
				*
;*********************************************************************

		;************ COPIA DE VARIABLES DE USO GENERAL ********************

		BANK0
		movf	Chksum0,W
		BANK2
		movwf	Bck_Chksum0

		BANK0
		movf	Chksum1,W
		BANK2
		movwf	Bck_Chksum1


		BANK0
		movf	Counter,W
		BANK2
		movwf	Bck_Counter

		;********** COPIA DE VARIABLES DE CABECERA ETHERNET  *************

		BANK0
		movf	sourceParam0,W
		BANK2
		movwf	Bck_sourceParam0

		BANK0
		movf	sourceParam1,W
		BANK2
		movwf	Bck_sourceParam1

		BANK0
		movf	sourceParam2,W
		BANK2
		movwf	Bck_sourceParam2

		BANK0
		movf	sourceParam3,W
		BANK2
		movwf	Bck_sourceParam3

		BANK0
		movf	sourceParam4,W
		BANK2
		movwf	Bck_sourceParam4
```

```
BANK0
movf        sourceParam5,W
BANK2
movwf       Bck_sourceParam5

;********** COPIA DE VARIABLES DE CABECERA INTERNET  *************

BANK0
movf        Source_IP0,W
BANK2
movwf       Bck_Source_IP0

BANK0
movf        Source_IP1,W
BANK2
movwf       Bck_Source_IP1

BANK0
movf        Source_IP2,W
BANK2
movwf       Bck_Source_IP2

BANK0
movf        Source_IP3,W
BANK2
movwf       Bck_Source_IP3

;********** COPIA DE VARIABLES DE CABECERA TCP  *************

BANK1
movf        tcp_dest_port0,W
BANK2
movwf       Bck_tcp_dest_port0

BANK1
movf        tcp_dest_port1,W
BANK2
movwf       Bck_tcp_dest_port1

BANK1
movf        tcp_source_port0,W
BANK2
movwf       Bck_tcp_source_port0

BANK1
movf        tcp_source_port1,W
BANK2
movwf       Bck_tcp_source_port1

BANK1
movf        xm_seq0_0,W
BANK2
movwf       Bck_xm_seq0_0

BANK1
movf        xm_seq0_1,W
BANK2
movwf       Bck_xm_seq0_1

BANK1
movf        xm_seq1_0,W
BANK2
movwf       Bck_xm_seq1_0

BANK1
movf        xm_seq1_1,W
BANK2
movwf       Bck_xm_seq1_1


BANK1
movf        xm_ack0_0,W
BANK2
movwf       Bck_xm_ack0_0

BANK1
movf        xm_ack0_1,W
BANK2
movwf       Bck_xm_ack0_1

BANK1
movf        xm_ack1_0,W
BANK2
movwf       Bck_xm_ack1_0

BANK1
movf        xm_ack1_1,W
BANK2
movwf       Bck_xm_ack1_1
```

```
            BANK1
            movf        data_flags0,W
            BANK2
            movwf       Bck_data_flags0

            BANK1
            movf        data_flags1,W
            BANK2
            movwf       Bck_data_flags1


            BANK0
            return

;*********************************************************************
;*****************************************************************              *
;NOMBRE:                restore_value_vars                                                                                      *               *
;CAPA:                          3 - TRANSPORTE
            *           *
;DESCRIPCIÓN:           Se reestablecen los valores de las variables            *               *
;                                               previo a la interrupción                                                        *               *
;*****************************************************************              *
restore_value_vars
;*********************************************************************

            ;************ COPIA DE VARIABLES DE USO GENERAL ********************

            BANK2
            movf        Bck_Chksum0,W
            BANK0
            movwf       Chksum0

            BANK2
            movf        Bck_Chksum1,W
            BANK0
            movwf       Chksum1

            BANK2
            movf        Bck_Counter,W
            BANK0
            movwf       Counter

            ;********** COPIA DE VARIABLES DE CABECERA ETHERNET  *************

            BANK2
            movf        Bck_sourceParam0,W
            BANK0
            movwf       sourceParam0

            BANK2
            movf        Bck_sourceParam1,W
            BANK0
            movwf       sourceParam1

            BANK2
            movf        Bck_sourceParam2,W
            BANK0
            movwf       sourceParam2

            BANK2
            movf        Bck_sourceParam3,W
            BANK0
            movwf       sourceParam3

            BANK2
            movf        Bck_sourceParam4,W
            BANK0
            movwf       sourceParam4

            BANK2
            movf        Bck_sourceParam5,W
            BANK0
            movwf       sourceParam5

            ;********** COPIA DE VARIABLES DE CABECERA INTERNET  *************

            BANK2
            movf        Bck_Source_IP0,W
            BANK0
            movwf       Source_IP0

            BANK2
            movf        Bck_Source_IP1,W
            BANK0
            movwf       Source_IP1

            BANK2
            movf        Bck_Source_IP2,W
            BANK0
            movwf       Source_IP2
```

```
		BANK2
		movf	Bck_Source_IP3,W
		BANK0
		movwf	Source_IP3

		;********** COPIA DE VARIABLES DE CABECERA TCP  *************

		BANK2
		movf	Bck_tcp_dest_port0,W
		BANK1
		movwf	tcp_dest_port0

		BANK2
		movf	Bck_tcp_dest_port1,W
		BANK1
		movwf	tcp_dest_port1

		BANK2
		movf	Bck_tcp_source_port0,W
		BANK1
		movwf	tcp_source_port0

		BANK2
		movf	Bck_tcp_source_port1,W
		BANK1
		movwf	tcp_source_port1

		BANK2
		movf	Bck_xm_seq0_0,W
		BANK1
		movwf	xm_seq0_0

		BANK2
		movf	Bck_xm_seq0_1,W
		BANK1
		movwf	xm_seq0_1

		BANK2
		movf	Bck_xm_seq1_0,W
		BANK1
		movwf	xm_seq1_0

		BANK2
		movf	Bck_xm_seq1_1,W
		BANK1
		movwf	xm_seq1_1


		BANK2
		movf	Bck_xm_ack0_0,W
		BANK1
		movwf	xm_ack0_0

		BANK2
		movf	Bck_xm_ack0_1,W
		BANK1
		movwf	xm_ack0_1

		BANK2
		movf	Bck_xm_ack1_0,W
		BANK1
		movwf	xm_ack1_0

		BANK2
		movf	Bck_xm_ack1_1,W
		BANK1
		movwf	xm_ack1_1

		BANK2
		movf	Bck_data_flags0,W
		BANK1
		movwf	data_flags0

		BANK2
		movf	Bck_data_flags1,W
		BANK1
		movwf	data_flags1

		BANK0
return


load_ascii_word
		BANK1
		bcf			PCLATH,4
		bcf			PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	inport1
		bsf			PCLATH,4
		bsf			PCLATH,3		;PAGINA 3 DE PROGRAMA
		movf	Register_Value1,W
		BANK2
```

```
		movwf	INSD2

		;Se recive la parte menos significativa de IP3
		BANK1
		bcf			PCLATH,4
		bcf			PCLATH,3		;PAGINA 0 DE PROGRAMA
		call	inport1
		bsf			PCLATH,4
		bsf			PCLATH,3		;PAGINA 3 DE PROGRAMA
		movf	Register_Value1,W
		BANK2
		movwf	INSD1
return



;*******************************************************************
;*****************************************************************		*
;NOMBRE:		Change_IP_device								*		*
;CAPA:				2 - INTERNET
		*		*
;DESCRIPCIÓN:		Se almacena el nuevo valor de la IP en la EEPROM* *
;						del PIC, para actualizarse al finalizar el		*		*
;						socket establecido
		*		*
;*****************************************************************		*
Change_IP_device
;*******************************************************************

		;La IP se recive en asscii y por lo tanto se debe convertir a exadecimal
		;Se recive la parte mas significativa de IP3
		call	load_ascii_word
		;Se convierte el ascci recivido a hexadecimal, el dato
		;queda en W
		call	Ascii_to_hex
			;Se almacena el dato capturado de la IP
			BANK2
			movwf	VALOR_EEPROM
			movlw	0x00
			movwf	ADDR_EEPROM
			call	save_value_eeprom

		;Se recive la parte mas significativa de IP2
		call	load_ascii_word
		;Se convierte el ascci recivido a hexadecimal, el dato
		;queda en W
		call	Ascii_to_hex
			;Se almacena el dato capturado de la IP
			BANK2
			movwf	VALOR_EEPROM
			movlw	0x01
			movwf	ADDR_EEPROM
			call	save_value_eeprom

		;Se recive la parte mas significativa de IP1
		call	load_ascii_word
		;Se convierte el ascci recivido a hexadecimal, el dato
		;queda en W
		call	Ascii_to_hex
			;Se almacena el dato capturado de la IP
			BANK2
			movwf	VALOR_EEPROM
			movlw	0x02
			movwf	ADDR_EEPROM
			call	save_value_eeprom

		;Se recive la parte mas significativa de IP0
		call	load_ascii_word
		;Se convierte el ascci recivido a hexadecimal, el dato
		;queda en W
		call	Ascii_to_hex
			;Se almacena el dato capturado de la IP
			BANK2
			movwf	VALOR_EEPROM
			movlw	0x03
			movwf	ADDR_EEPROM
			call	save_value_eeprom

		;Se activan las banderas correspondientes a un ACK
		BANK1
		movlw	0x00
		movwf	data_flags0
		bsf			data_flags0,0x04	;Se habilita la bandera TCP_ACK

		;Se procede a enviar un ACK del paquete recivido
		bcf			PCLATH,4
		bsf			PCLATH,3		;PAGINA 1 DE PROGRAMA
		call	socket_control
		bsf			PCLATH,4
		bsf			PCLATH,3		;PAGINA 3 DE PROGRAMA
		BANK0
```

```
return


;*********************************************************************
;*****************************************************************          *
;NOMBRE:              save_value_eeprom                                                                              *          *
;DESCRIPCIÓN:         Almacena un byte en la EEPROM del PIC                           *          *
;ENTRADAS:                      ADDR_EEPROM: Direccion en la EEPROM del PIC           *          *
;                                         VALOR_EEPROM: Valor a almacenar                                   *          *
;*****************************************************************          *
save_value_eeprom
;*********************************************************************
          BYTE_SAVE
                                ;Se almacena el dato en la EEPROM del PIC
                                BANK3
                                btfsc     EECON1, WR          ;esperemos si hay un proceso de escritura en pogreso
                                goto      $-1
                                bcf                 EECON1,WREN         ;desactivamos la funcion de escritura.
                                BANK2
                                movf      ADDR_EEPROM,W
                                movwf     EEADR               ;cargamos la direccion en la escriviremos
                                movf      VALOR_EEPROM,W
                                movwf     EEDATA
                                BANK3
                                bcf                 EECON1, EEPGD       ;seleccionamos la memoria de datos
                                bsf                 EECON1, WREN        ;activamos la funcio de escritura
                                movlw     0x55
                                movwf     EECON2              ;escrivimos 0x55 en EECON2
                                movlw     0xaa
                                movwf     EECON2              ;escrivimos 0xaa en EECON2
                                bsf                 EECON1,WR           ;iniciamos la operacion de escritura.
                                BANK3
                                btfsc     EECON1, WR          ;esperemos a que el proceso de escritura termine
                                goto      $-1
                                bcf                 EECON1,WREN         ;desactivamos la funcion de escritura.
                                BANK2

                                call      LEER_DATO           ;leemos el byte escrito en la EEPROM para verificar

                                subwf     VALOR_EEPROM,W                ;que se guardo correctamente
                                btfss     STATUS,Z
                                goto      BYTE_SAVE
return


;*********************************************************************
;*****************************************************************          *
;NOMBRE:              LEER_DATO
          *           *
;DESCRIPCIÓN:         Leer un byte en la EEPROM del PIC                               *          *
;ENTRADAS:                      ADDR_EEPROM: Direccion en la EEPROM del PIC           *          *
;SALIDAS:             Almacena el dato en W                                                             *          *
;*****************************************************************          *
LEER_DATO
;*********************************************************************
                      BANK2
                      movf      ADDR_EEPROM,W                 ;direccion a leer
                      movwf     EEADR
                      BANK3
                      bcf                 EECON1,EEPGD        ;seleccionamos la memoria de datos
                      bsf                 EECON1,RD;iniciamos la lectura
                      btfsc     EECON1, RD          ;esperemos que la lectura termine
                      goto      $-1
                      BANK2
                      movf      EEDATA,W  ;ponemos el dato leido en W
return


;*********************************************************************
;*****************************************************************          *
;NOMBRE:              Ascii_to_hex                                                                                   *          *
;DESCRIPCIÓN:         Convierte un dato en ascii (2 bytes) a HEX                 *          *
;ENTRADAS:                      INSD1, INSD2
          *           *
;SALIDAS:             Almacena el dato en W                                                             *          *
;*****************************************************************          *
Ascii_to_hex
;*********************************************************************
                      BANK2
                      call      ascii_to_bit_msb    ;concatenar INSD2 y INSD1 en un solo byte
                      call      ascii_to_bit_lsb

                      BANK2
                      movf      INSD2,W
                      iorwf     INSD1,W
return


;*********************************************************************
```

```
;****************************************************************          *
;NOMBRE:             ascii_to_bit_msb                                                                        *        *
;DESCRIPCIÓN:        Converite de Ascci a HEX (4 bits MSB)                 *        *
;ENTRADAS:                   INSD2
                     *       *
;SALIDAS:            Almacena el dato en INSD2                                                       *        *
;****************************************************************          *
ascii_to_bit_msb
;********************************************************************

                     BANK2
                     movlw   0x30                ;es el dato ASCII un cero?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_0                ;si

                     movlw   0x31                ;es el dato ASCII un 1?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_1                ;si

                     movlw   0x32                ;es el dato ASCII un 2?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_2                ;si

                     movlw   0x33                ;es el dato ASCII un 3?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_3                ;si

                     movlw   0x34                ;es el dato ASCII un 4?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_4                ;si

                     movlw   0x35                ;es el dato ASCII un 5?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_5                ;si

                     movlw   0x36                ;es el dato ASCII un 6?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_6                ;si

                     movlw   0x37                ;es el dato ASCII un 7?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_7                ;si

                     movlw   0x38                ;es el dato ASCII un 8?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_8                ;si

                     movlw   0x39                ;es el dato ASCII un 9?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_9                ;si

                     movlw   0x41                ;es el dato ASCII un 0xa?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_a                ;si

                     movlw   0x42                ;es el dato ASCII un 0xb?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_b                ;si


                     movlw   0x43                ;es el dato ASCII un 0xc?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_c                ;si

                     movlw   0x44                ;es el dato ASCII un 0xd?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_d                ;si


                     movlw   0x45                ;es el dato ASCII un 0xe?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_e                ;si

                     movlw   0x46                ;es el dato ASCII un 0xf?
                     subwf   INSD2,W
                     btfsc   STATUS,Z
                     goto    es_f                ;si
```

```
                movlw   0x61            ;es el dato ASCII un 0xa?
                subwf   INSD2,W
                btfsc   STATUS,Z
                goto    es_a            ;si

                movlw   0x62            ;es el dato ASCII un 0xb?
                subwf   INSD2,W
                btfsc   STATUS,Z
                goto    es_b            ;si

                movlw   0x63            ;es el dato ASCII un 0xc?
                subwf   INSD2,W
                btfsc   STATUS,Z
                goto    es_c            ;si

                movlw   0x64            ;es el dato ASCII un 0xd?
                subwf   INSD2,W
                btfsc   STATUS,Z
                goto    es_d            ;si

                movlw   0x65            ;es el dato ASCII un 0xe?
                subwf   INSD2,W
                btfsc   STATUS,Z
                goto    es_e            ;si

                movlw   0x66            ;es el dato ASCII un 0xf?
                subwf   INSD2,W
                btfsc   STATUS,Z
                goto    es_f            ;si

es_0    clrf    INSD2
                return
es_1    movlw   0x10
                movwf   INSD2
                return
es_2    movlw   0x20
                movwf   INSD2
                return
es_3    movlw   0x30
                movwf   INSD2
                return
es_4    movlw   0x40
                movwf   INSD2
                return
es_5    movlw   0x50
                movwf   INSD2
                return
es_6    movlw   0x60
                movwf   INSD2
                return
es_7    movlw   0x70
                movwf   INSD2
                return
es_8    movlw   0x80
                movwf   INSD2
                return

es_9    movlw   0x90
                movwf   INSD2
                return
es_a    movlw   0xa0
                movwf   INSD2
                return
es_b    movlw   0xb0
                movwf   INSD2
                return
es_c    movlw   0xc0
                movwf   INSD2
                return
es_d    movlw   0xd0
                movwf   INSD2
                return
es_e    movlw   0xe0
                movwf   INSD2
                return
es_f    movlw   0xf0
                movwf   INSD2
                return


;*********************************************************************
;****************************************************************          *
;NOMBRE:         ascii_to_bit_lsb                                                                    *        *
;DESCRIPCIÓN:    Converite de Ascci a HEX (4 bits LSB)          *        *
;ENTRADAS:               INSD1
                *        *
;SALIDAS:        Almacena el dato en INSD1                                                   *        *
;****************************************************************          *
ascii_to_bit_lsb
;*********************************************************************
                BANK2
```

```
movlw   0x30            ;es el dato ASCII un 0?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x0           ;si

movlw   0x31            ;es el dato ASCII un 1?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x1           ;si

movlw   0x32            ;es el dato ASCII un 2?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x2           ;si

movlw   0x33            ;es el dato ASCII un 3?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x3           ;si

movlw   0x34            ;es el dato ASCII un 4?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x4           ;si

movlw   0x35            ;es el dato ASCII un 5?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x5           ;si

movlw   0x36            ;es el dato ASCII un 6?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x6           ;si

movlw   0x37            ;es el dato ASCII un 7?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x7           ;si

movlw   0x38            ;es el dato ASCII un 8?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x8           ;si

movlw   0x39            ;es el dato ASCII un 9?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_x9           ;si

movlw   0x41            ;es el dato ASCII un 0xa?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xa           ;si

movlw   0x42            ;es el dato ASCII un 0xb?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xb           ;si

movlw   0x43            ;es el dato ASCII un 0xc?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xc           ;si

movlw   0x44            ;es el dato ASCII un 0xd?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xd           ;si

movlw   0x45            ;es el dato ASCII un 0xe?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xe           ;si

movlw   0x46            ;es el dato ASCII un 0xf?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xf           ;si

movlw   0x61            ;es el dato ASCII un 0xa?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xa           ;si

movlw   0x62            ;es el dato ASCII un 0xb?
subwf   INSD1,W
btfsc   STATUS,Z
goto    es_xb           ;si

movlw   0x63            ;es el dato ASCII un 0xc?
subwf   INSD1,W
```

```
                btfsc   STATUS,Z
                goto    es_xc           ;si

                movlw   0x64            ;es el dato ASCII un 0xd?
                subwf   INSD1,W
                btfsc   STATUS,Z
                goto    es_xd           ;si

                movlw   0x65            ;es el dato ASCII un 0xe?
                subwf   INSD1,W
                btfsc   STATUS,Z
                goto    es_xe           ;si

                movlw   0x66            ;es el dato ASCII un 0xf?
                subwf   INSD1,W
                btfsc   STATUS,Z
                goto    es_xf           ;si

es_x0   clrf    INSD1
                return
es_x1   movlw   0x01
                movwf   INSD1
                return
es_x2   movlw   0x02
                movwf   INSD1
                return
es_x3   movlw   0x03
                movwf   INSD1
                return
es_x4   movlw   0x04
                movwf   INSD1
                return
es_x5   movlw   0x05
                movwf   INSD1
                return
es_x6   movlw   0x06
                movwf   INSD1
                return
es_x7   movlw   0x07
                movwf   INSD1
                return
es_x8   movlw   0x08
                movwf   INSD1
                return

es_x9   movlw   0x09
                movwf   INSD1
                return
es_xa   movlw   0x0a
                movwf   INSD1
                return
es_xb   movlw   0x0b
                movwf   INSD1
                return
es_xc   movlw   0x0c
                movwf   INSD1
                return
es_xd   movlw   0x0d
                movwf   INSD1
                return
es_xe   movlw   0x0e
                movwf   INSD1
                return
es_xf   movlw   0x0f
                movwf   INSD1

return


;*******************************************************************
;*****************************************************************  *
;NOMBRE:            LOAD_IP_FROM_EEPROM                                                 *
                *
;DESCRIPCIÓN:       Actualiza el valor de la IP                                 *       *
;ENTRADAS:                  Datos en EEPROM del PIC                                     *
                *
;SALIDAS:           Ninguna
                *       *
;*****************************************************************  *
LOAD_IP_FROM_EEPROM
;*******************************************************************
                ;Se asigna la dirección IP local
                BANK2
                movlw   0x00                    ; Numero IP mas significativo
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_IP0

                BANK2
                movlw   0x01
```

```
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_IP1

                BANK2
                movlw   0x02
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_IP2


                BANK2
                movlw   0x03                    ;Numero de IP menos significativo
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_IP3
return


;*****************************************************************
;*****************************************************************  *
;NOMBRE:         LOAD_MAC_FROM_EEPROM                                                    *       *
;DESCRIPCIÓN:    Actualiza el valor de la IP                                     *       *
;ENTRADAS:               Datos en EEPROM del PIC                                         *
                *
;SALIDAS:        Ninguna
                *       *
;*****************************************************************  *
LOAD_MAC_FROM_EEPROM;
                *
;*****************************************************************
                ;Se asigna la MAC local
                BANK2
                movlw   0x04                    ; MAC MSB
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_Addr0

                BANK2
                movlw   0x05
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_Addr1

                BANK2
                movlw   0x06
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_Addr2

                BANK2
                movlw   0x07
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_Addr3

                BANK2
                movlw   0x08
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_Addr4

                BANK2
                movlw   0x09                    ;MAC LSB
                movwf   ADDR_EEPROM
                call    LEER_DATO
                BANK0
                movwf   Host_Addr5
return

Reset_IP
                BANK0
;////////////////////////////////
;               movlw   0xAA
;               bcf             PCLATH,0x04
;               bcf             PCLATH,0x03             ;PAGINA 0 DE PROGRAMA
;               call    RS232_Tx
;               bsf             PCLATH,0x04
;               bsf             PCLATH,0x03             ;PAGINA 3 DE PROGRAMA
;////////////////////////////////

                movlw   HOST_IP0
                movwf   Host_IP0
                        ;Se almacena el dato capturado de la IP
```

```
                BANK2
                movwf   VALOR_EEPROM
                movlw   0x00
                movwf   ADDR_EEPROM
                call    save_value_eeprom

        BANK0
        movlw   HOST_IP1
        movwf   Host_IP1
                ;Se almacena el dato capturado de la IP
                BANK2
                movwf   VALOR_EEPROM
                movlw   0x01
                movwf   ADDR_EEPROM
                call    save_value_eeprom

        BANK0
        movlw   HOST_IP2
        movwf   Host_IP2
                ;Se almacena el dato capturado de la IP
                BANK2
                movwf   VALOR_EEPROM
                movlw   0x02
                movwf   ADDR_EEPROM
                call    save_value_eeprom

        BANK0
        movlw   HOST_IP3
        movwf   Host_IP3
                ;Se almacena el dato capturado de la IP
                BANK2
                movwf   VALOR_EEPROM
                movlw   0x03
                movwf   ADDR_EEPROM
                call    save_value_eeprom

        BANK0
        bsf             Request_MODBUS_present,4

return


        END
```

A.7 Glosario

ACK:

Un bit de control (del inglés 'acknowledge') que no ocupa posición del espacio de números de secuencias y que indica que el campo de acuse de recibo de este segmento especifica el próximo número de secuencia que el emisor de este segmento espera recibir, por lo que también realiza un acuse de recibo de todos los números de secuencia anteriores.

Cabecera:

Información de control al principio de un mensaje, segmento, datagrama, paquete o bloque de datos.

Conexión:

Un camino lógico de comunicación identificado por un par de conectores.

Datagrama:

Un mensaje que se envía en una red de comunicaciones de ordenadores por intercambio de paquetes.

Fragmento:

Una porción de una unidad lógica de datos, en particular, un fragmento de Internet es una porción de un datagrama de Internet.

Host:

Un computador. En particular, un origen o destino de los mensajes desde el punto de vista de la red de comunicaciones.

Octeto:

Un byte de ocho bits.

Paquete:

Un conjunto de datos con una cabecera que puede estar o no lógicamente completa. Más a menudo, se refiere a un empaquetamiento físico de datos que lógico.

Proceso:

Un programa en ejecución. Un origen o destino de datos desde el punto de vista de TCP o cualquier otro protocolo de 'host' a 'host'.

Protocolo de Comunicación:

Un protocolo de comunicaciones de datos es un conjunto de normas, o un acuerdo, que determina el formato y la transmisión de datos.

Valor de Relleno (Padding):

El campo Valor de Relleno de la cabecera Internet se utiliza para asegurar que el campo de datos comienza en una dirección múltiplo de 32 bits. El relleno es cero.