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

“Diseño y construcción de una solución Web para controlar instrumentos con opción GPIB conectados a un ordenador servidor”

PRESENTADO POR:

OSBALDO ALEXANDER CACERES RIVAS
LUIS ALEIXANDRE LEON VASQUEZ

PARA OPTAR AL TÍTULO DE:
INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, OCTUBRE DEL 2006

UNIVERSIDAD DE EL SALVADOR

RECTORA :

DRA. MARÍA ISABEL RODRÍGUEZ

SECRETARIA GENERAL :

LICDA. ALICIA MARGARITA RIVAS DE RECINOS

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIO :

ING. OSCAR EDUARDO MARROQUÍN HERNÁNDEZ

ESCUELA DE INGENIERÍA ELÉCTRICA

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 :

“Diseño y construcción de una solución WEB para controlar instrumentos con opción GPIB conectados a un ordenador servidor”

Presentado por :
OSBALDO ALEXANDER CACERES RIVAS
LUIS ALEIXANDRE LEON VASQUEZ

Trabajo de Graduación aprobado por :

Docente Director :
ING. HUGO MIGUEL COLATO RODRÍGUEZ

San Salvador, octubre del 2006.

Trabajo de Graduación Aprobado por:

Docente Director :

ING. HUGO MIGUEL COLATO RODRÍGUEZ

DEDICATORIA

AGRADECIMIENTOS

PREFACIO

El desarrollo de los dispositivos electrónicos para medición ha variado ampliamente, desde los dispositivos más simples como multimetros análogos hasta equipos automáticos de medición de gran escala, pero hay algo en común entre estos tipos de dispositivos y es que ambos requieren operación manual de sus controles para realizar el trabajo para el que se han diseñado o para alguna configuración o cambio en su funcionamiento. Entre los avances de estos dispositivos podemos mencionar la capacidad de almacenar datos, obtener un número de muestras de alguna señal y ponerlos a disposición del usuario a través de distintos tipos de puerto para comunicación ya sea entre un PC, alguna tarjeta de memoria o con otro dispositivo de medición. Existe una variedad de protocolos o tipos de comunicación entre los dispositivos de los que hablamos como por ejemplo el de nuestro interés el IEEE 488.

La principal ventaja de comunicar los dispositivos de medición con un PC es que podemos tener acceso a datos o resultados de mediciones y acceso a la configuración remota del aparato sin tener que operar manualmente los controles del panel frontal del aparato de medición. Lo favorable es que todos los ajustes se hacen desde una estación central con la que podemos controlar varios dispositivos a la vez y los datos obtenidos pueden ser procesados y presentados según la aplicación.

El objetivo principal de este proyecto es diseñar un sistema capaz de controlar instrumentos de medición eléctrica con la opción GPIB conocida también como IEEE 488. El diseño de la tarjeta controladora es general para dispositivos con opción GPIB permitiendo controlar cualquier instrumento que disponga de una interfase GPIB, mientras que la aplicación grafica es específica de acuerdo a los requerimientos finales, por tal razón en el desarrollo de este proyecto se realizara nada mas una interfase grafica para un instrumento el Tektronix tds640 propiedad de la empresa COSESNA.

RESUMEN DEL TRABAJO

En este trabajo se ha tratado de proporcionar una solución eficaz y óptima a la necesidad de un sistema para controlar de forma remota una serie de instrumentos con la característica en común que implementan el protocolo IEEE488, por tal motivo se ha construido la Tarjeta Controladora GPIB la cual se conecta en red por medio del bus GPIB con los instrumentos y ejerce el papel de controlador del bus, con esta tarjeta se puede controlar cualquier instrumento que implemente el protocolo, laTarjeta Controladora GPIB posee un diseño basado en microcontroladores donde la unidad central es un microcontrolador de la familia microchip, es este el que se encarga de realizar la función controlador exigida por el estándar IEEE488.

Para que la tarjeta tenga comunicación con el usuario por medio de un ordenador se ha implementado en la tarjeta el protocolo TCP/IP, este permite la conexión con una computadora o con una red LAN por medio de un cable UTP, esta es otra tarea de la que también se encarga el microcontrolador.

La tarjeta se gestiona como cualquier host con una dirección IP y un número de puerto por el cual atiende las peticiones, permitiendo al usuario controlar y acceder a los instrumentos en forma remota a través del bus GPIB.

A pesar de que la tarjeta puede ser utilizada para controlar cualquier instrumento siempre hay que hacer una interfase grafica especifica para el instrumento a tratar con el fin de poder presentar al usuario la información obtenida del instrumento, para esta parte del trabajo se ha utilizado el lenguaje de programación java para construir una interfase grafica y así poder emular los controles del instrumento Tektronix TDS640A el cual es un osciloscopio de cuatro canales, esta interfase reside dentro del ordenador permitiendo establecer comunicación con la tarjeta controladora y comunicación con el instrumento a tratar.

La última etapa de este proyecto es configurar al servidor con el cual se comunica la Tarjeta Controladora con un servicio http para que dentro de cualquier red se pueda acceder y controlar al instrumento Tektronix en forma remota.

En su total el sistema diseñado y construido consta de un servicio http, servicio manejador multiplataforma, servicio de almacenamiento en base de datos y una tarjeta controladora que implementa varios protocolos de comunicación como el IEEE488, TCP/IP y un protocolo diseñado para la tarjeta para intercambiar datos con los servicios montados en el servidor.

TABLA DE CONTENIDOS

Capítulo Página

LISTA DE TABLAS

LISTA DE FIGURAS

CAPITULO I

GENERALIDADES

Introducción

Este capitulo es un acercamiento general al desarrollo de este documento, aquí se describen los objetivos, justificación y alcances del proyecto; luego el problema del control remoto de instrumentos conectados al bus IEEE488 es planteado el cual es el punto de partida para el desarrollo de la solución; también se muestra un breve análisis para la solución del problema definiendo las características y condiciones que debe cumplir el sistema.

Finalmente son definidos los componentes de *software y* hardware elegidos para diseñar la solución al problema, y es mostrado un esquema donde una tarjeta controladora del bus GPIB es el centro del sistema, esta ultima interactúa con módulos de programas que prestan servicios a clientes conectados a la red ethernet (LAN) para controlar cualquier dispositivo conectado al bus GPIB vía WEB.

1.0 Objetivos

GENERAL:

- Diseñar y proponer una solución completa sobre el control y acceso remoto vía WEB (Internet o intranet) de instrumentos conectados al bus IEEE488.

ESPECIFICOS:

- Diseñar y construir una tarjeta inteligente (standalone) que cumpla con la norma IEEE488.1, la cual se conecte a una red ethernet
- Diseñar y programar en lenguaje Java y ensamblador del PIC16F877A los controladores (*drivers*) de la tarjeta IEEE488.
- Diseñar y construir un sitio WEB utilizando tecnología Java tal que permita obtener información de control y configuración de los equipos conectados al bus.

1.1 Justificación

Con los avances en el área de la computación muchos proyectos de ingeniería están siendo llevados a cabo utilizando tecnología de código abierto (*open source*) y pastillas microcontroladores (microprocesadores) obteniendo productos a muy bajo costo y de muy buena calidad, lo que incide a que hoy en día el uso de estas tecnologías sea mucho mas frecuente para obtener soluciones, las ventajas de los microcontroladores hacen que estos sean ampliamente utilizados debido a su excelente rendimiento y a su bajo costo. En este proyecto se espera obtener una solución bastante eficaz a un costo mucho menor al que se podría obtener por este en el mercado salvadoreño.

1.2 Alcances

- Diseñar una Interfase WEB utilizando tecnología Java la cual cumpla con los requerimientos para controlar el Instrumento Tektronix TDS640A
- Diseñar una interfase para el bus IEEE488.
- Controlar remotamente vía WEB el equipo conectado al bus IEEE488.

1.3 Planteamiento del problema

Cuando se adquiere un instrumento de medición con grandes cualidades hay que tratar de explotar todo lo que se nos ofrece y así sacar mejor provecho de los recursos con los que contamos ya sea para mejorar producción o para optimizar procesos de cualquier índole.

En nuestro caso tenemos al dispositivo:

1. Osciloscopio Tektronix TDS640A.

Este tiene opción GPIB (*General Purpose Interfaces Bus)*, del cual se profundizará más adelante en el capitulo II), esta opción permite establecer comunicación entre los instrumentos por medio de un bus de datos a donde uno de estos dispositivos conectados controla el flujo de las operaciones a realizar información.
Desde este punto de vista el problema es diseñar y desarrollar un sistema capaz de controlar tales dispositivos conectados al bus y obtener información o datos desde estos para procesarla y presentarla al cliente.

1.4 Análisis del problema

En esta solución para el control remoto de instrumentos GPIB se comienza según el esquema de la figura 1.1, donde se plantean las entradas del problema a una caja negra en la cual se realizan varios procesos para llegar a una salida que es la solución del problema; en los siguientes párrafos se describen cada uno de los componentes del esquema de la figura 1.1

Figura 1.1: Esquema de planteamiento del programa

ENTRADA:

- Conocimiento del estándar IEEE 488: este estándar define la estructura del bus GPIB como líneas de control, líneas de datos, protocolos y las

distintas secuencias y estados para establecer comunicación y realizar la configuración remota de un dispositivo.

- Conocimiento sobre microcontroladores:

- Conocimiento sobre desarrollo de programas computacionales en la plataforma Java

- Conocimiento sobre arquitecturas de redes LAN en computadoras específicamente la capa física del modelo OSI

- Identificación de los instrumentos a controlar así como sus funciones, variables que los instrumentos son capaces de medir, marcas, obtención de manuales técnicos y modelos del instrumento.

- Definición de los controles remotos del instrumento, definir que opciones de control del medidor pueden ser manejadas remotamente por medio del bus GPIB.

- Definición de los parámetros de configuración o resultados de las mediciones que el instrumento pone a disposición para lectura remota

- Definir el controlador; un dispositivo del sistema que realizará la función de controlar el flujo de información en el bus.

PROCESO

La creación de la solución lleva los siguientes pasos:

Diseño de solución prototipo

- Diseño de la tarjeta controladora: definición de hardware y software de los módulos necesarios para poder implementar los protocolos GPIB y TCP/IP.

- Diseño de servicios para el control remoto de instrumentos: definición de tecnología a utilizar para la prestación del servicio Web al cliente y servicio de control de la tarjeta GPIB.

Construcción de solución prototipo

- Construcción de la tarjeta controladora y sus distintos componentes
- Construcción del firmware de la tarjeta controladora

- Construcción del software respectivo para los servicios Web del cliente y del manejador multiplataforma GPIB usando Java.

Ajustes en cuanto a funcionamiento de hardware y software

- Modificaciones de cualquier tipo que lleven a un mejor funcionamiento de cada uno de los elementos de la solución.

SALIDA:

Son los resultados esperados después de los procesos mencionados anteriormente, tal solución cuenta con los siguientes componentes:

- Sitio WEB creado completamente con tecnología Java
- Tarjeta controladora “standalone” (host) con opción ethernet
- Controlador o manejador multiplataforma (Linux/Windows) de la tarjeta controladora.
- Base de datos en MySQL para el almacenamiento persistente de configuración.

1.5 Solución al problema

La solución propuesta consta con las características que se plantearon en el diagrama de entrada salida de la sección anterior, los elementos de la solución se pueden clasificar en los siguientes grupos: elementos del bus IEEE-488, elementos del bus ethernet y servicios de aplicación de la tarjeta. Una descripción de estos grupos es hecha en las siguientes sub-secciones.
La figura1.2 muestra un esquema de las diferentes entidades del sistema propuesto.

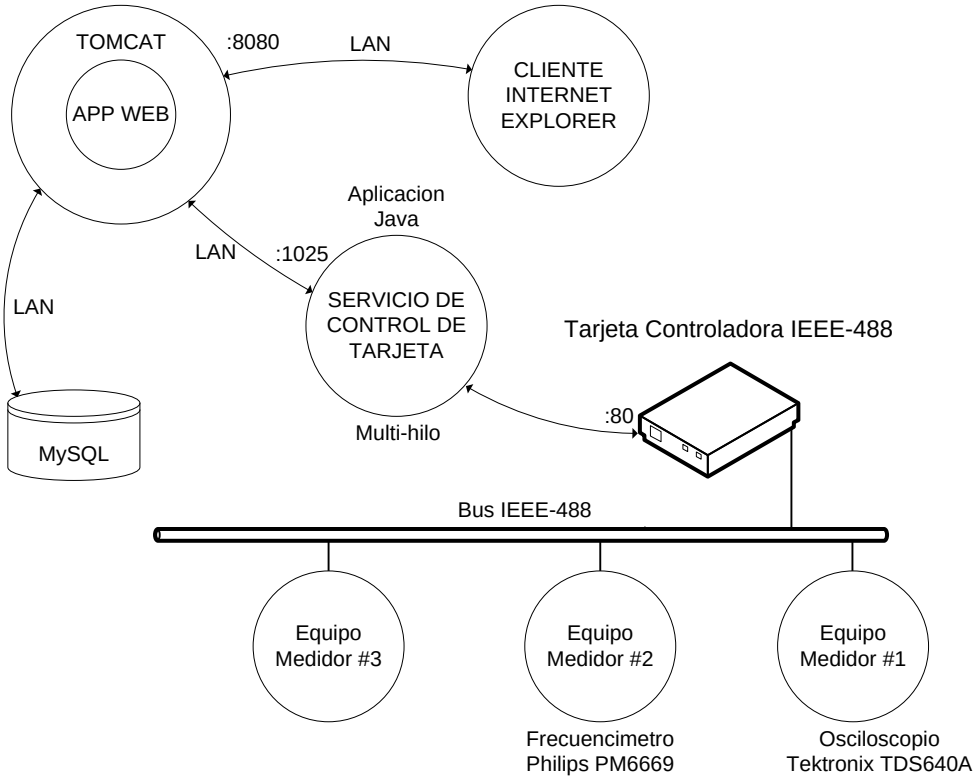


Figura 1.2. Sistema GPIB propuesto desde un punto de vista físico o global.

1.5.1 Elementos en el bus IEEE-488

La norma IEEE488 define un bus de 8 bit, que actualmente se aplica predominantemente para el control de aparatos de medición eléctrica. El bus IEEE488 es un sistema maestro/servidor, en el que por regla general un controlador regula el acceso del bus de los otros dispositivos. Antes de la transmisión propia de datos útiles, este controlador tiene que direccionar el instrumento o dispositivo deseado, según el sentido de datos necesario. Los dos componentes de este bus IEEE-488 son:

- La tarjeta controladora del bus IEEE-488: cumple con la función controlador del bus

- Osciloscopio Tektronix TDS640A: instrumento con opción GPIB a controlar en forma remota

1.5.1.1 La tarjeta del bus IEEE- 488

El circuito controlador es la parte central de un sistema GPIB, este posee varias capacidades, entre estas se listan las siguientes:

- Tiene la capacidad de enviar la señal IFC para limpiar todas las interfaces de los dispositivos y vuelve el control al controlador del sistema.

- Tiene la capacidad de controlar instrumentos remotamente por medio de la señal REN, permitiendo así que los dispositivos respondan a los datos del bus una vez que él este dispuesto para escuchar.

- Responde a las peticiones de servicio por parte de los instrumentos

- Tiene la capacidad de direccionar a uno o varios dispositivos para colocarse en un estado de escuchar o receptor de los comandos que se transmitan a través del bus.

- Tiene la capacidad de direccionar a uno y solo un dispositivo para colocarse en un estado de transmisor de datos a través del bus.

Otras funcionalidades de la tarjeta controladora

La elección para un controlador debe satisfacer muchas necesidades no solo la de controlar el bus sino también brindar el control del bus al usuario, y para eso se propone un sistema Tarjeta controladora basada en el microcontrolador con tecnología RISC (por ser de bajo costo y alto rendimiento) la cual en conjunto con un servicio de software (corriendo o ejecutándose en una computadora personal), proporcionaran las siguientes ventajas:

- El microcontrolador se encarga de realizar la función de controlar el bus.

- La conexión existente entre el servicio de software y el microcontrolador proporciona flexibilidad tal que el usuario pueda tomar el control del bus, ya sea para configurar instrumentos o para realizar mediciones.

- Funcionalidad de extender el sistema a nivel de desarrollo del software el cual interactúa con el servicio de software ejecutándose en un equipo servidor; estos pueden ser sistemas de escritorio o sistemas de publicación WEB (cliente-servidor),

1.5.1.2 Osciloscopio Tektronix TDS640A

El osciloscopio Tektronix TDS640A es un dispositivo que muestra de forma grafica una señal eléctrica, esta grafica muestra como la señal cambia sobre el tiempo, representando el eje vertical (Y) el voltaje, el horizontal (X) representa el tiempo y la intensidad o brillo de la pantalla es llamado el eje Z.
A continuación se muestra la lista de características del instrumento a controlar y una breve descripción de sus capacidades funcionales.

Osciloscopio Tektronix TDS640A:

- Ancho de banda: 500MHz
- Numero de canales: 4
- Canales simultáneos: 4
- Velocidad máxima de muestreo simultanea: 2GS/s
- Longitud máxima de grabación: 2000pt/s
- Rango escala vertical: 1mV/div – 10V/div
- Rango escala tiempo: 500ps/div – 10s/div
- Impedancia de entrada: 1MΩ/50Ω
- Voltaje máximo de entrada: 400Vrms
- Precisión en la base del tiempo: 0.1%
- Fuentes de disparo: interna y externa
- Modos de disparo: Auto, Edge, Normal, Pulse, Single

El instrumento es capaz de comparar una señal de entrada con otra de plantilla previamente almacenada y cuando la señal de prueba cae fuera de los límites establecidos se activa una alarma la cual es configurable para hacer un sonido de una alarma o realizar una copia en duro de la señal de entrada.

También puede realizar una imagen de la actual vista de la pantalla del instrumento en formato “bmp” (mapa de bits) la cual puede ser observada desde cualquier software visor de imágenes dentro de una computadora.

1.5.2 Elementos en el bus ethernet (LAN)

Estos elementos están definidos para trabajar dentro de una red LAN y son utilizados para proveer comunicación entre la tarjeta controladora y el servicio de control del bus IEEE-488.

1.5.2.1 Tarjeta controladora del bus IEEE-488

Este dispositivo sirviendo como interfase entre los buses IEEE-488 y el ethernet hace posible la comunicación desde cualquier cliente conectado al bus ethernet con los instrumentos conectados al bus IEEE-488. Las funcionalidades implementadas en este se resumen en la siguiente tabla:

Funcionalidad	Ejemplo	Descripción
Controlador bus IEEE-488	Ejecutar la secuencia de transferencia de datos al bus	Realiza la función de controlador del estándar IEE-488
Comunicación TCP/IP	Recibir datos con comandos de interfaz o de dispositivo.	Implementa el protocolo de comunicación TCP/IP tal que permite la recepción y envío de datos por medio del puerto ethernet, para tal fin se crea un "socket" (comunicación entre dos puertos de distintos host) con el servicio de control del bus.
Respuesta mensaje "ping"	Realizar un ping a la tarjeta controladora desde una consola o Terminal de una computadora para determinar si existe conexión entre ambas.	Implementa la respuesta a mensajes ping por medio del protocolo IP/ICMP

Tabla 1.1: funcionalidad de la Tarjeta Controladora del bus IEEE-488.

1.5.2.2 Servicio de comunicación hacia la tarjeta controladora del bus IEEE-488

Este es un componente de software multi-hilo por medio del cual es posible la comunicación con la tarjeta controladora del bus IEEE-488, las funcionalidades implementadas en este se resumen en la siguiente tabla:

Funcionalidad	Ejemplo	Descripción
Crear Objetos de Instrumentos	Creación de un objeto Osciloscopio Tektronix	Crear un objeto con las características y funcionalidades del osciloscopio Tektronix
Comunicación TCP/IP	Enviar datos a la tarjeta controladora	Establecer un socket de comunicación TCP con la tarjeta controladora por un puerto determinado, para permitir comunicación bidireccional.
Servicio de control bus IEEE-488	Se reciben datos de control o configuración para el osciloscopio	Recibe datos provenientes del servicio Tomcat para que tales sean enviados a la tarjeta controladora.
Almacenamiento en base de datos	Almacenar en una tabla los resultados de una medición	Permitir la gestión para el almacenamiento de información en una base de datos
Configuración Tarjeta	Cambiar la dirección IP y puerto TCP de la tarjeta	Envía comandos de configuración para la tarjeta controladora

Tabla 1.2: funcionalidad del servicio de comunicación hacia la tarjeta controladora

1.5.3 Servicios de aplicación para el Osciloscopio Tektronix

El software descrito en esta sección en su conjunto es una aplicación hecha a la medida del osciloscopio Tektronix TDS640A, esta aplicación del osciloscopio esta compuesta por un servicio de base de datos (tablas de medición y parámetros de configuración), un servicio de comunicación con el servicio de control de la tarjeta controladora y un servicio WEB para el cliente.

Software controlador que permite interactuar con Osciloscopio Tektronix TDS640A

Para la creación de este software es utiliza la programación orientada a objetos. Este software implementa las características funcionales del osciloscopio y del estándar IEEE-488.2 (ver capitulo II) las cuales permiten que este sea configurado y controlado en forma remota, entre las funcionalidades de este software tenemos las siguientes:

Funcionalidad	Ejemplo	Descripción
Hilo (proceso) receptor de datos	Este proceso captura los datos cuando el medidor envía información como respuesta a una petición.	Este proceso captura los datos que el servicio de control de la tarjeta pone a su disposición, luego son interpretados y se ejecuta la gestión adecuada con el servidor de base de datos para el almacenamiento de la información.
Objeto Osciloscopio Tektronix	Declaración de un objeto de este tipo para poder controlar el osciloscopio en forma remota	Este objeto proporciona un conjunto de métodos y atributos que le permiten al servicio controlador de la tarjeta enviar los comandos de dispositivo al instrumento.

Tabla 1.3: funcionalidad del software controlador que permite interactuar con el Osciloscopio Tektronix TDS640A

1.5.3.2 Servicio WEB para el cliente

Para controlar el osciloscopio en forma remota el cliente necesitara acceder al servicio de aplicación WEB, tal servicio proporciona una interfase grafica simulando los controles del panel frontal del medidor y establece comunicación vía LAN con el servicio de control de la tarjeta controladora para poder acceder en forma remota el instrumento, un resumen de las funciones de este servicio se muestra en la siguiente tabla.

Funcionalidad	Ejemplo	Descripción
Panel frontal Tektronix	El cliente desde un navegador (ej. Internet Explorer) accede al servicio WEB	Servicio Tomcat de pagina WEB que simula los controles del panel frontal del Osciloscopio Tektronix,
Comunicación con el servicio de control de la tarjeta	Cuando el cliente selecciona una configuración en la página WEB y envía tal información y esta es enviada al servicio de control de la tarjeta controladora.	este establece comunicación con el servicio de control de la tarjeta controladora para acceder al instrumento de modo remoto
Comunicación con el servicio de base de datos	El cliente solicita la lectura de las ultimas mediciones realizadas	Este servicio permite leer las tablas de medición y configuración del instrumento

Tabla 1.4: funcionalidad del servicio WEB del Osciloscopio Tektronix TDS640A

1.5.3.3 Servicio de base de datos en MySQL

Para prestar el servicio de gestión de bases de datos se tiene MySQL (ver figura 1.2), este servicio es usado por la aplicación WEB para leer información de interés y por el servicio de control de la tarjeta controladora para escribir la información obtenida del instrumento. En la siguiente tabla es especifican las funciones de este servicio.

Funcionalidad	Ejemplo	Descripción
Servicio para la aplicación WEB	El cliente solicita la lectura de las ultimas mediciones realizadas entonces se lee la tabla de mediciones y se retornan los datos leídos	Este servicio proporciona la salida de información almacenada en las tablas del osciloscopio o alguna de interés
Servicio para el controlador de la tarjeta	El servicio controlador tiene que almacenar los datos enviados por el instrumento como repuesta a una orden de adquisición de datos.	Este servicio proporciona la entrada de información para ser almacenada en las tablas del osciloscopio o alguna de interés

Tabla 1.5: funcionalidades del servicio de Base de Datos.

CONCLUSIONES DEL CAPITULO I

- ✓ La creación de la solución para el control de instrumentos con el bus IEEE488 esta dividida en las siguientes etapas (ver Figura 1.1) el centro del sistema es la tarjeta controladora, esta interactúa con un ordenador servidor y con instrumentos conectados al bus IEEE488.
  - Tarjeta controladora GPIB
  - Servicio de control de la tarjeta
  - Servicio de aplicación Web
  - Servicio de base de datos

- ✓ Para la construcción de la aplicación en el servidor se utilizará el lenguaje de programación *JAVA*, para aportar su alta flexibilidad y portabilidad.

- ✓ La tarjeta controladora tiene como elemento principal el microcontrolador PIC16F877A de la marca *microchip*, esta maneja el bus IEEE488 y la comunicación con el servidor que es asistida por el controlador de ethernet *realtek* RTL8019AS que implementa los protocolos de comunicación TCP/IP.

- ✓ La tarjeta controladora implementa las funcionalidades básicas del estándar IEEE-488.1

- ✓ El manejador de la tarjeta creado en lenguaje ensamblador del PIC16F877A es el que se encarga de implementar las funciones de un controlador definidas por el estándar IEEE-488.1

- ✓ El servicio de control de la tarjeta controladora es el complemento del manejador de la tarjeta para realizar la función de controlador multiplataforma del bus GPIB.

- ✓ La aplicación de usuario es creada utilizando JavaScript, JSP, HTML y Java; esta mostrará los controles del instrumento y funcionalidades de configuración del sistema obteniendose una interfase muy amigable para el usuario.

- ✓ El servicio de base de datos creado en MySQL es utilizado para persistir información de configuración de la tarjeta controladora y del instrumento.

CAPITULO II

ESTANDAR IEEE 488

Introducción

En este capitulo se presenta una breve historia y descripción del estándar IEEE488, cuando sea necesario para efectos de claridad se hará referencia al numero de sección y pagina del propio estándar. El estándar esta compuesto por dos versiones 488.1 y 488.2; aunque cada uno trata tópicos distintos convirtiéndose así uno complemento del otro, llenando los vacíos que habían quedado en la primer versión.

En la versión del IEEE488.1 del estándar se cubren los aspectos físicos, de hardware que se requieren para implementarlo y las funciones básicas para poder establecer comunicación con los dispositivos del bus GPIB. En la versión siguiente 488.2 se describen los protocolos y el formato de los mensajes de comunicación entre un instrumento y un controlador o una computadora con funciones de controlador IEEE-488 sea este de tipo “stand-alone” o para funcionar como parte de un equipo ordenador.

2.1 Historia del Estándar

Este fue originariamente desarrollado por *Hewlett Packard* en 1965 bajo el nombre de **HPIB** (*Hewlett-Packard Interface Bus*). Su mayor difusión se debe a que posteriormente fue adoptado por la organización IEEE, que en 1978 lo definió mediante el estándar **IEEE 488**, aunque comúnmente se le conoce como GPIB (*General Purpose Interface Bus*).

La funcionalidad del estándar GPIB ha evolucionado a lo largo del tiempo y se encuentra descrito en las siguientes especificaciones:

- IEEE 488.1 (1975): Especificación que define las características de nivel físico (mecánico y eléctrico), así como sus características funcionales básicas.

- IEEE 488.2 (1987): Especificación que define las configuraciones mínimas, los comandos y formatos de datos básicos y comunes a todos los equipos y los protocolos que se siguen en las comunicaciones.

- SCPI (Standar Commands for Programmable Instrumentation): Especificación construida sobre el estándar IEEE 488.2 que define una estructura de comandos estándar aceptados por múltiples instrumentos de muchos fabricantes.

Entre algunos de los fabricantes tenemos:

- **HP:** Hewlett-Packard.
- **Agilent:** Agilent Technologies
- **Cec:** Capital Equipment Corporation
- **Iotech:** IOTech hardware.
- **Keithley:** Keithley
- **Mcc:** Measurement Computing Corporation
- **Ni:** National Instruments.

2.2 Descripción del bus GPIB (IEEE 488.1)

Esta sección es una descripción del estándar para mayores detalles ver la documentación completa del estándar IEEE488.1 incluido en el CD de este trabajo.

2.2.1 Características del bus

Las características más relevantes de este bus GPIB son las siguientes:

- Permite la interconexión de hasta 15 equipos, de los que uno de ellos es el controlador, que establece la función que debe ejercer cada uno de los otros.

- Un dispositivo conectado al bus, puede enviar o recibir información hacia o desde cualquiera de los otros 14 equipos

- El límite práctico de velocidad de intercambio de datos es de 500 Kbytes/s (o lo que es lo mismo 4 Mbits/s).

- La interconexión entre equipos se realiza utilizando cables de 24 y 25 hilos, finalizados en conectores de doble boca (macho por un lado y hembra por el otro), que permite la interconexión de los equipos en cualquier configuración (estrella, línea, o cualquier combinación de ellas).

- Las longitudes máximas permitidas en los cables es de 20 metros. Los cables que se comercializan son de 0.5, 1, 2, 4 y 8 metros.

- Un aspecto importante es que el GPIB requiere **lógica TTL inversa** en los niveles de señal de las líneas del bus.

Un dispositivo conectado al bus GPIB puede estar operando como uno o varios de los siguientes modos de comportamiento:

- Controlador (*Controller*): Con capacidad de direccionar o establecer quien envía o recibe datos y el modo de operación del bus (solo un equipo puede ser "*controller*").

- *Talker*: Con capacidad de enviar datos a otros equipos. sólo un dispositivo a la vez puede ser establecido por el controlador para que opere como *Talker*.

- *Listener*: Con capacidad de recibir datos de otros equipos. En cada bus pueden existir uno o varios equipos con capacidad de recibir datos desde el bus.

- *Idler*: Sin ninguna capacidad respecto del bus.

2.2.2 Líneas de señal del bus GPIB

El bus GPIB está basado en 16 líneas activas, además de la señal de referencia. Estas 16 líneas se organizan en tres buses:

- **Bus de Datos**: Líneas de entrada salida DIO1-DIO8**,** es un bus bidireccional de 8 líneas orientado a la transferencia de bytes o de caracteres ASCII.

- **Bus de sincronización de la transferencia de datos:** Es un conjunto de tres líneas que se utilizan de forma coordinada para asegurar la transferencia de datos entre los equipos.

  DAV: por sus iniciales en Ingles "*Data valid*", es una de las líneas de sincronización que permite la transferencia de datos por el bus de datos. Un estado lógico *TRUE* (0, nivel bajo) en está línea significa que el equipo establecido como "*talker*" activo ha establecido datos válidos sobre el bus de datos que deben ser leídos por todos los equipos establecidos como "*listener*".

  NRFD: por sus iniciales en Ingles "*Not Ready For Data*", es una de las líneas de sincronización, que es gobernada por los equipos establecidos como "*listener*". Cuando esta línea está en estado lógico *TRUE* (0, nivel bajo), significa que algún equipo de entre los "*listener*" no está aún dispuesto para aceptar nuevos datos. El que esta línea se encuentre en estado *TRUE*, inhibe al equipo "*talker*" a que inicie el envío de un nuevo dato. El que esta línea esté en estado lógico *FALSE* (1, nivel alto), significa que todos los equipos "*listener*" se encuentran a la espera de un nuevo dato.

  NDAC: por sus iniciales en Ingles "*Not Data Accepted*", es gobernada por los equipos que están establecidos como "*listener*". Cuando se encuentra en estado lógico *TRUE* (0, nivel bajo), significa que alguno de los equipos establecidos como "*listener*" aún está pendiente de leer un dato, y en consecuencia, el talker debe esperar aún para retirar los datos del bus. Cuando esta línea

se encuentra en estado lógico *FALSE* significa que ya todos los equipos establecidos como "*listener*" han leído el dato transferido.

- **Bus de control**: Está constituido por 5 líneas que se utilizan para transferir comandos entre los equipos, relativos al modo de interpretar los datos que se transfieren o comandos básicos de gobierno de la interfaz del bus.

  ATN: por sus iniciales en Ingles "*Attention*", es una señal que utiliza el controlador para establecer con un estado lógico *TRUE* en ella que el dato que se envía por el bus de datos es un comando enviado por el "*controller*".

  IFC: por sus iniciales en Ingles "*InterFace Clear*", está bajo el exclusivo uso del controlador. Cuando es establecido en esta línea un estado lógico *TRUE*, todos los equipos conectados al bus deben de reestablecer su interfase GPIB.

  SRQ: por sus iniciales en Ingles "*Service Request*", es utilizado por los equipos conectados al bus para comunicar al controlador que requieren ser atendidos por alguna causa (ha concluido una actividad, se ha producido un error, existe algún dato para transferir, etc.). Cuando el controlador detecta un estado lógico *TRUE* en esta línea, debe iniciar una encuesta (*polling*) para determinar que equipo causó el requerimiento, y en el caso de que proceda, satisfacer su demanda.

  REN: por sus iniciales en Ingles "*Remote Enable*", es una línea con la que el controlador al establecerla a un estado lógico *TRUE*, habilita a todos los equipos conectados al bus para que reciban datos, o comandos en forma remota.

  EOI: por sus iniciales en Ingles "*End Or Identify*", Esta línea tiene dos funciones:

  En primer lugar, el "*talker*" puede indicar, poniendo a estado lógico *TRUE* esta línea, que concluye su envío de datos.

  En segundo lugar, esta línea es utilizada por el controlador para iniciar una encuesta paralela. En este caso el controlador debe poner simultáneamente a estado lógico *TRUE* las señales ATN y EOI, y como respuesta a ello, los equipos que previamente hayan sido configurados para participar en la encuesta paralela transfieren sus bits de estado sobre el bus.

En la figura 2.1 se muestra la distribución de líneas en un conector GPIB que también se listan en la tabla 2.1.

LINEAS DE DATOS	PIN	LINEAS DE CONTROL	PIN	LINEAS DE SINCRONIZACION	PIN
DIO1	1	IFC	9	DAV	6
DIO2	2	REN	17	NRFD	7
DIO3	3	ATN	11	NDAC	8
DIO4	4	SRQ	10		
DIO5	13	EOI	5		
DIO6	14				
DIO7	15				
DIO8	16				

Tabla 2.1 listado de las líneas de señal del conector GPIB

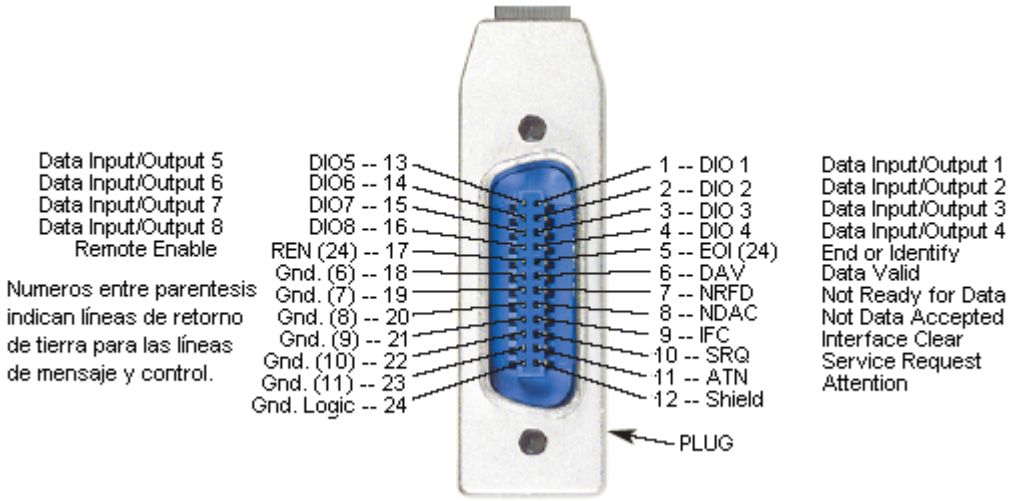


Figura 2.1. Conector para el bus GPIB.

2.2.3 Dirección GPIB

Los dispositivos conectados al bus GPIB deben tener un código o dirección de bus comprendido entre 0-30. Este código debe ser establecido en cada equipo, bien estableciendo unos conmutadores presentes en su panel trasero, o mediante su software interno.

2.2.4 Mensajes GPIB

Entre los equipos conectados al bus GPIB se transfieren mensajes constituidos por secuencias de caracteres alfanuméricos (caracteres ASCII), hay dos tipos de mensajes de acuerdo con el estado de la señal de control “ATN”.

- **Mensajes de Datos**: estos son mensajes que contiene información relativa a la funcionalidad de un equipo. Ejemplos son: instrucción de programación, resultado de medida, estatus de un equipo, etc.

- **Mensajes de Comando**: son mensajes que tiene como función controlar el modo de operación del bus. Ejemplos son: Inicialización del bus, cambio del modo de operación de un equipo, transferencia del control, etc.

MENSAJES DE COMANDO

Los comandos son siempre enviados del desde el controlador a los otros equipos para sincronizar su estado de operación o para establecer su estado de operación. La señal ATN debe ser establecida a estado lógico *TRUE* por el "controlador", para indicar que el mensaje sobre las líneas de datos es un “mensaje de comando”.

El "controlador" puede enviar cinco tipos de comandos en forma de caracteres ASCII (ver pagina 98 Anexo B del estándar IEEE 488.1) a los otros equipos. Si los caracteres enviados los vemos en forma de byte los tres bits b7, b6 y b5. Definen la naturaleza de cada comando:

b7	b6	b5	Tipo de commando
0	0	0	Direccionamiento
0	0	1	Universal
0	1	x	Listen
1	0	x	Talk
1	1	x	Secondary

Tabla 2.2 Tipos de comandos de interfaz

Comandos estado TALK/LISTEN

MTA *My Talk Address*: Establece el modo Talker en el equipo con dirección "ddddd" mandando el siguiente byte.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	Modo Talk		Dirección de dispositivo				
X	1	0	d	d	d	d	d

UNT *Untalk*: El equipo en modo Talker pasa a modo "*Idler*", significa que no responde a comandos

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	Modo Talk						
X	1	0	1	1	1	1	1

MLA *My Listen Address*: Establece el modo Listener en el equipo con dirección "ddddd" mandando el siguiente byte:

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	Modo Listen		Dirección de dispositivo				
X	0	1	d	d	d	d	d

UNL *Unlisten*: Todos los equipos en modo Listener pasan a modo *Idler*

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	Modo Listen						
X	0	1	1	1	1	1	1

Comandos universales

Estos son Mensajes enviados por el Controlador con un alcance a todos los equipos.

LLO: **Local Lockout,** Se deshabilitan los paneles de control de todos los equipos conectados al Bus.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	1	0	0	0	1

DCL: **Device Clear** Inicializa las interfaces hardware-software de los equipos.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	1	0	1	0	0

PPU: **Parallel Poll Unconfigure** Se cancela la programación previa de los equipos a fin de responder en la encuesta paralela (Parallel Poll)

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	1	0	1	0	1

SPE: **Serial Poll Enable** Habilita a todos los equipos a fin de que respondan a la encuesta serie (Serie Poll).

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	1	1	0	0	0

SPD: **Serie Poll Disable** Deshabilita el modo de encuesta serie.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	1	1	0	0	1

Comandos de direccionamiento

Estos comandos van destinados y afectan únicamente a aquellos equipos que previamente han sido establecidos en "*Listener*".

GTL: **Go To Local,** Retorna el control de los paneles a todos los equipos en estado Listener.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	0	0	0	0	1

SDC: **Selected Device Clear,** inicializa las interfaces hardware/software de los equipos en estado Listener.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	0	0	1	0	0

PPC: **Parallel Poll Configure** Configura la respuesta a una encuesta paralela (Pararell Poll) los equipos en estado Listener. Los equipos quedan a la espera de un comando MSA.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	0	0	1	0	1

MSA: **My Secondary Address** Establece la línea (bbb) y el estado (c=0 => request = FALSE) con la que responden a una encuesta paralela de los equipos Listener.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	1	1	x	c	b	b	b

GET: **Group Trigger** Dispara el *trigger* de los equipos en estado Listener.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	0	1	0	0	0

TCT: Take Control Establece como Active *Controller* al equipo que está establecido como Listener.

bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1
X	0	0	0	1	0	0	1

2.2.5 Funciones de interfaz

Una función de interfaz es el elemento del sistema por medio del cual un dispositivo puede recibir, procesar, y enviar mensajes.
En la tabla 2.3 se muestra la lista de las funciones de interfaz con las que puede contar un dispositivo con opción GPIB.

Función de Interfaz	Símbolo
Source handshake	SH
Acceptor handshake	AH
Talker or extended talker	T o TE
Listener or extended listener	L o LE
Service request	SR
Remote local	RL
Parallel poll	PP
Device clear	DC
Device trigger	DT
Controller	C

Tabla 2.3: repertorio de funciones de interfaz.

SH: Source Handshake

La función de interfaz SH provee un dispositivo con la capacidad para garantizar la transferencia apropiada de mensajes multilínea. Una secuencia enclavada de "handshake" entre la función SH y una o más funciones aceptores de handshake (AH) (cada una contenida dentro de dispositivos separados) garantiza la transferencia asíncrona de cada mensaje. La función de interfaz SH controla la iniciación de, y la terminación de, la transferencia de un byte de mensaje.
.

AH: Acceptor Handshake

La función AH provee un dispositivo con la capacidad para garantizar la recepción apropiada de mensajes multilínea remotos. Una secuencia enclavada de "handshake" entre una función SH y una o más funciones AH (cada una contenida dentro de dispositivos separados) garantiza el traslado asincrónico de cada byte de mensaje. Una función AH puede demorar tanto la iniciación de, o la terminación de, una transferencia de un mensaje multilínea hasta que esté preparado para continuar con el proceso de transferencia.

T: Talker

La función de interfaz **T** provee un dispositivo con la capacidad para enviar datos dependientes de dispositivo (incluyendo datos de condición durante una secuencia encuesta serie) sobre la interfaz de otros dispositivos. Esta capacidad existe únicamente cuando la función de interfaz T es direccionada como talker.

L: Listener

La función de interfaz **L** provee un dispositivo con la capacidad para recibir datos dependientes de dispositivo (incluyendo datos de condición) sobre la interfaz desde otros dispositivos. Esta capacidad existe solamente cuando la función está dirigida para escuchar o recibir datos.

SR: Solicitud de servicio

La función de interfaz **SR** provee un dispositivo con la capacidad para pedir servicio asincrónicamente al controlador a cargo de la interfaz.

RL: Local Remoto

La función indica al dispositivo que hay información de entrada desde el panel frontal de control (local) o hay información correspondiente de entrada desde la interfaz (remoto) para ser usada.

Inicialmente todos los controles locales de las Funciones de dispositivo asociadas (panel frontal o trasero) son operativas y el dispositivo puede almacenar, pero no responder a el mensaje dependiente de dispositivo correspondiente desde la interfaz.

PP: polling paralelo

La función de interfaz **PP** provee un dispositivo con la capacidad para presentar un mensaje PPR al controlador a cargo sin estar previamente direccionado como "*talker*".

Las líneas de señal de DIO1 hasta DIO8 se usan para transmitir los bits de condición del dispositivo durante el polling paralelo. A fin de que un dispositivo responda con un mensaje PPR, un dispositivo deberá ser asignado (configurado) a una sola línea DIO o por el controlador o por un mensaje local.

Un controlador inicia una secuencia paralela de polling mientras que cualquier dispositivo pide la iniciación de una secuencia de polling serial.

Un polling paralelo habilita la transferencia de datos de condición desde múltiples dispositivos concurrentemente mientras que una secuencia de polling serial colecta datos de condición desde cada dispositivo.

DC: Device Clear

La función de interfaz **DC** provee el dispositivo con la capacidad para limpiarse (inicializarse) o individualmente o como parte de un grupo de dispositivos.

DT: Device Trigger

La función de interfaz **DT** provee el dispositivo con la capacidad para tener su operación básica comenzada o individualmente o como parte de un grupo de dispositivos.

C: controler

La función de interfaz **C** provee un dispositivo con la capacidad para enviar direccionamiento a dispositivos, comandos universales y comandos dirigidos a otros dispositivos sobre la interfaz. También provee la capacidad para conducir el poleo paralelo para determinar cual de los dispositivos requiere servicio.
Una función de interfaz C puede ejercer sus capacidades solo cuando esta enviando el mensaje ATN sobre la interfaz

2.2.6 Secuencia Handshake

Una secuencia “Handshake” es utilizada para la transferencia de datos entre dos dispositivos, donde el dispositivo emisor implementa las funciones Source y Talker y el receptor implementa las funciones Aceptor y Listener

Para iniciar un ciclo de transferencia de datos entre dos dispositivos algunos de ellos deben ser configurados como escuchas y uno como hablante. Inicialmente todas las líneas están inactivas (nivel alto), luego el controlador envía el mensaje “*Interface Clear*” (IFC), luego envía la señal “*Remote Enable*” (REN) y después envía la señal “Atención” (ATN), ahora la transferencia satisfactoria de datos depende de los dispositivos escuchas y hablantes, el ciclo de transferencia de datos se muestra en la figura 2.2 y se detalla según los siguientes pasos.

1. La función “*Source*” pone DAV en nivel alto indicando dato no valido.

2. La función “*Aceptor*” pone NRFD en nivel bajo indicando no listo para datos, y pone NDAC en nivel bajo indicando no aceptor de datos.

3. La función “*Source*” revisa por una condición de error (ambos NRFD y NDAC en alto).

4. Entonces pone el byte a transmitir en las líneas DIO.

5. La función “*Source*” espera un tiempo para que el dato en las líneas DIO sea estable.

6. La función “*Aceptor*” indica que esta listo para aceptar el dato, poniendo NRFD en nivel alto.

7. La función “*Source*”, revisa sí NRFD esta en nivel alto.

8. Pone DAV en nivel bajo para indicar que el dato en las líneas DIO es valido.

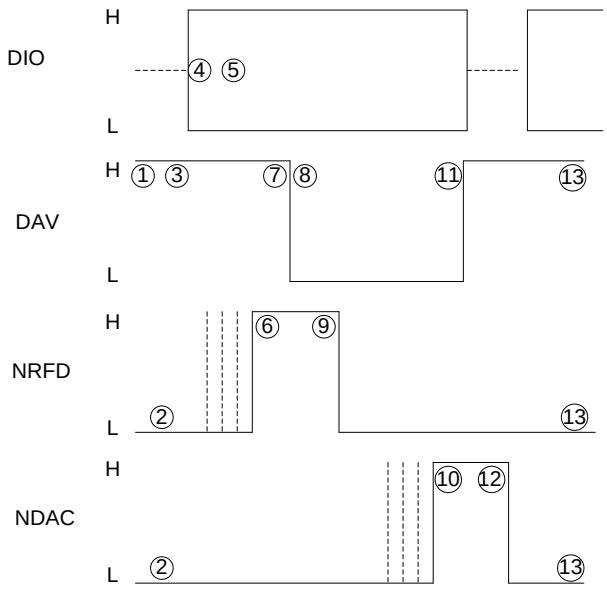
9. El último *aceptor* pone NRFD en nivel bajo para indicar que no esta listo para un nuevo dato, entonces acepta el dato actual. Otros aceptores lo hacen a sus propias velocidades.

10. El *aceptor* pone NDAC en nivel alto para indicar que este ha aceptado el dato.

11. La función "*Source*", tiene que estar censando a NDAC si es alto, entonces pone DAV en alto. Esto indica al aceptor que el dato en las líneas DIO debe considerarse no valido.

12. Los aceptores, deben censar DAV en alto y poner NDAC en bajo como preparación para el siguiente ciclo.

13. El ciclo regresa a la condición número uno lista para un nuevo ciclo de "*Handshake*".



L=nivel bajo voltaje, true

H=nivel alto voltaje, false

* ① - ⑬ = pasos de una secuencia handshake

Figura 2. 2. Secuencia Handshake, y los distintos estados de las líneas que intervienen en este ciclo de transferencia de datos.

* números que representan los pasos en una secuencia handshake, ver sección 2.2.6

En la siguiente figura se muestra el flujograma de las funciones internas involucradas en una secuencia handshake.

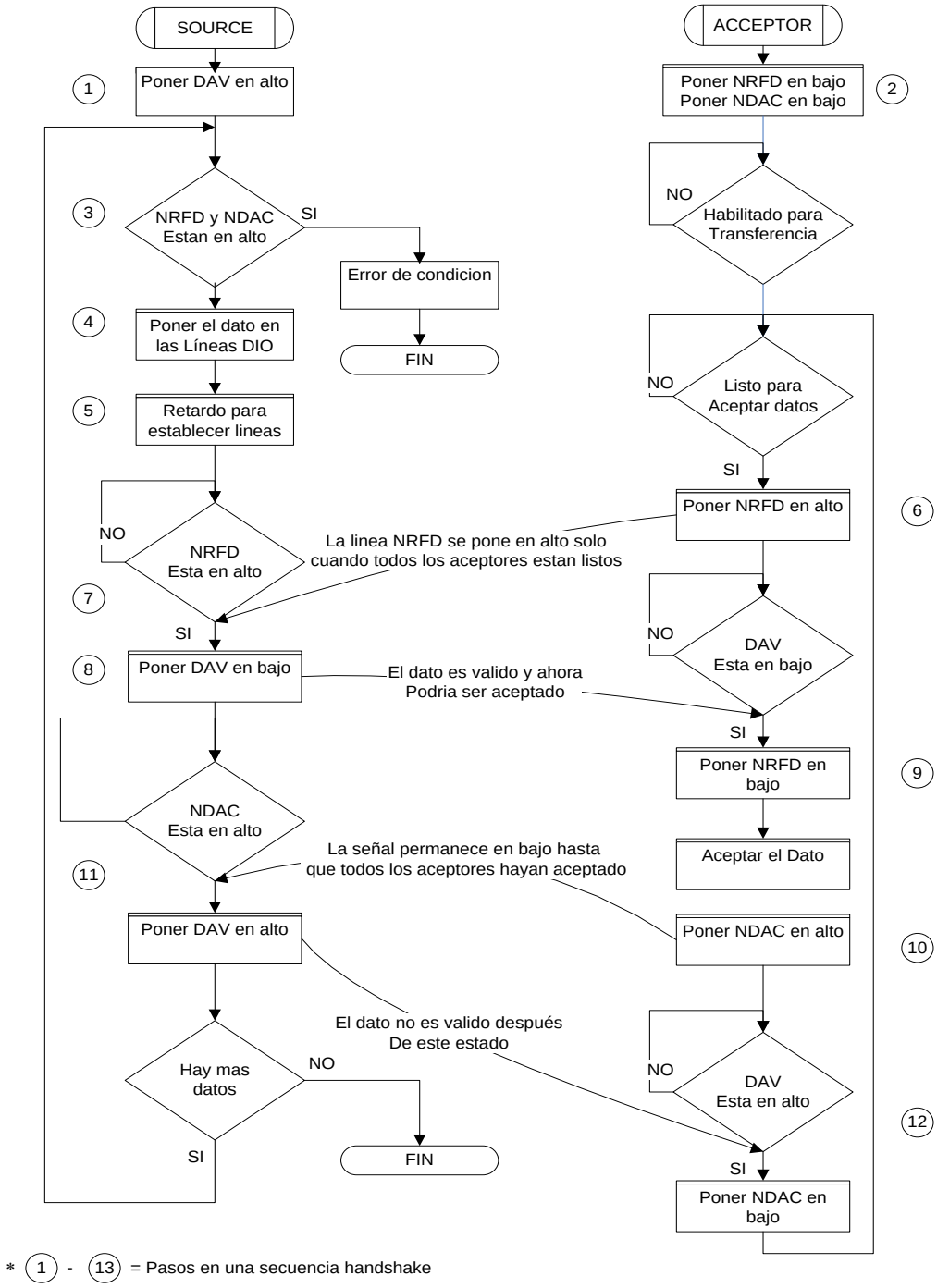


Figura 2.3. Diagramas de flujo de las funciones Source y Aceptor handshake.

* números que representan los pasos en una secuencia handshake, ver sección 2.2.6

2.2.7 Especificaciones eléctricas

- La relación entre los estados lógicos y los estados o niveles eléctricos para la codificación de los mensajes remotos presentes en las líneas de señal es la siguiente:

Estado Lógico	Nivel de señal eléctrica
FALSO, 0	≥ +2.0V, estado alto
VERDADERO, 1	≤ +0.8V, estado bajo

Tabla 2.4: Logica TTL inversa.

Los estados alto y bajos están basados en niveles TTL para que la fuente de voltaje no exceda +5.2V DC y sea referenciada a tierra lógica.

- Para las líneas de señal SRQ, NRFD y NDAC pueden usarse dispositivos tipo colector abierto.

- Para las líneas de señal DIO1-8, DAV IFC, ATN, REN y EOI pueden usarse dispositivos tipo colector abierto o de tres estados, con excepción de DIO1-8 deben ser de tipo colector abierto para aplicaciones de un censo Paralelo (*Parallel Polling*).

- Las especificaciones de los dispositivos para manejar las líneas de señal son las siguientes:

NIVELES DE VOLTAJES Y CORRIENTES DEL ESTÁNDAR GPIB				
Nivel de señal	Salida(hacia el bus)		Entrada(hacia el dispositivo)	
	Voltaje	Corriente	Voltaje	Corriente
ALTO	≥+2.4V	–5.2mA	+2.0V	----
BAJO	≤+0.5V	+48mA	+0.8V	----

Tabla 2.5: Niveles de corriente y voltaje del estándar GPIB.

Para líneas de salida el dispositivo manejador de la línea debe ser capaz de entregar 48mA continuamente.

- Cada línea de señal será terminada dentro del sistema por una carga resistiva con el propósito importante de establecer un voltaje de estado

estacionario cuando todos los conductores en una línea están en el estado de alta impedancia. Esta carga también se utiliza para mantener una impedancia uniforme del dispositivo en la línea y para mejorar inmunidad de ruido.

2.3 Protocolos de comunicación GPIB (IEEE 488.2)

El estándar IEEE 488.2 define los modos de operación básicos de los equipos que lo satisfacen. Define los protocolos de intercambio de mensajes con el que el equipo y el controlador se comunican, así como, facilidades básicas de control del instrumento.

2.3.1 Características de un protocolo básico

- El equipo y el controlador se comunican intercambiando mensajes de ordenes y mensajes de respuesta.

- Los mensajes de ordenes son enviados por el controlador, y pueden ser de dos tipos:

  Ordenes de control: Que requieren un cambio de estado del equipo, pero que no requieren ninguna respuesta.

  Ordenes de requerimiento: También llamados “Query" estos mensajes solicitan información sobre el estado del equipo o sobre información que posee.

- El equipo solo habla (envía un mensaje de salida) como respuesta de una orden de requerimiento.

- El controlador solo admite un mensaje de salida (respuesta de una orden de requerimiento), y lo requiere antes de enviar un nuevo mensaje de órdenes. En caso contrario se genera una situación de bloqueo.

- La regla básica del protocolo es:

  "El equipo solo habla cuando está dispuesto a ello, y en ese caso, tiene que hablar antes de que se le ordene hacer una tarea nueva"

- El controlador puede enviar un mensaje conteniendo múltiples órdenes de requerimientos. A esto se le denomina un "requerimiento compuesto". Los

diferentes requerimientos dentro del mensaje deben estar separados por el delimitador "**;**". Los mensajes de respuesta son colocados en la cola de salida, separados entre sí, por el mismo delimitador "**;**".

- Los comandos son ejecutados en el orden en que han sido recibidos.

2.3.2 Comandos básicos SCPI

El estándar IEEE 488.2 define un conjunto de órdenes básicas que realizan funciones comunes a todos los equipos, con independencia de su naturaleza.

COMANDO	NOMBRE DEL COMANDO	FUNCION
*CLS	Clear Status Command	-Despeja el registro de estado y los registros de incidencia.
*ESE	Event Status Enable Command	-Habilita bits del registro de habilitación de incidencias
*ESE?	Event Status Enable Query	-Interroga el registró de habilitación de Incidencias estándar.
*ESR?	Event Status Register Query	-Interroga el registro de Incidencias estándar.
*IDN	Identification Query	-Identifica tipo de instrumento y versión software.
*LRN?	Learn Device Setup Query	-Requiere el estado actual del equipo.
*OPC	Operation Complete Command	-Fija el bit de "Operación completa" del registro estándar.
*OPC?	Operation Complete Query	-Responde con "1" si se han ejecutado ordenes previas.
*OPT?	Option Identification Query	-Requiere la opción instalada en el equipo.
*RCL	Recall Command	-Restaura el estado del equipo del registro save/recall.
*RST	Reset Command	-Sitúa al equipo en el estado básico de referencia.
*SAV	Save Command	-Almacena el estado actual en un registro save/recall.
*SRE	Service Request Enable Command	- Habilita los bits del registro de habilitación de Byte de estado.
*SRE?	Service Request Enable Query	- Requiere el contenido del registro SER de habilitación del byte de estado
*STB?	Read Status Byte Query	- Requiere el estado del registro resumido del Byte de estado.
*TRG	Trigger Command	- Arranca o dispara la operación del equipo de forma remota.
*TST?	Selft-Test Query	-Requiere el resultado del autotest del equipo.
*WAI	Wait-to-Continue Command	- Espera a que se realicen todas las operaciones pendientes.

Tabla 2.6. Comandos básicos SCPI.

2.3.3 Estructura de los mensajes de dispositivo

Los mensajes dependientes de dispositivo que se envían a cualquier dispositivo con el fin de que estos realicen alguna tarea son cadenas alfanuméricas, con los siguientes elementos.

Cabecera (*Header*): es el identificador de la orden que se ejecuta, está compuesta por uno o mas mnemónicos separados por los delimitadores " **:** ". Cada uno identifica un nivel dentro del árbol de órdenes de cada instrumento, y en conjunto, identifican unívocamente una orden.

El identificador o mnemónico de un nivel se compone de unos caracteres obligatorios, y otros que pueden incluirse optativamente, a efecto de mayor legibilidad del programa.

Dos aspectos importantes de las cabeceras de comandos de dispositivos son:

- Las cabeceras de las ordenes de requerimiento finalizan en el carácter "**?**".

- Las cabeceras de las ordenes básicas del protocolo IEEE 488.2 (ver tabla 2.6) no están incluidas en el árbol y siempre empiezan por el carácter "*", por ejemplo “*RST”.

Separador: Es un espacio o conjunto de espacios.

Datos de programas: Son valores numéricos o alfanuméricos que cualifican la orden. Pueden ser uno o varios, según la orden de que se trate. En este caso los datos deben estar separados por comas o combinaciones de comas y espacios.

Terminador: se puede usar un carácter de fin de transmisión o de avance de línea como <LF> para terminar el envió de datos o al mismo tiempo el controlador puede activar la línea EOI para indicar el fin de transmisión.

Las cabeceras pueden ser de tres tipos: cabeceras de comando simple, cabeceras de comando compuesto y cabeceras de comando común.

Cabeceras de comando simple: estas contienen un solo mnemónico de programa, la sintaxis se muestra en la figura 2.4.

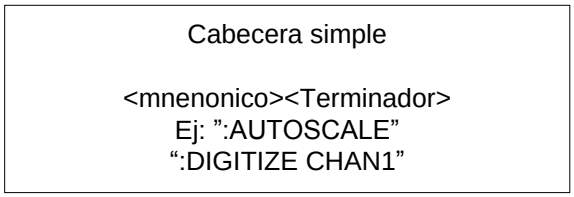


Figura 2.4. Sintaxis de una cabecera de comando simple.

Cabeceras de comando compuesto: son combinaciones de dos mnemónicos de programa el primero selecciona el subsistema de comandos del instrumento y el segundo selecciona una función dentro del subsistema.

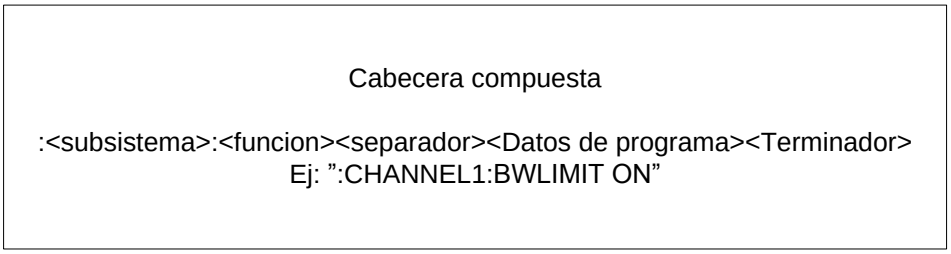


Figura 2.5. Sintaxis de una cabecera de comando compuesta.

Cabeceras de comando común: son los comandos básicos del estándar definidos en la tabla 2.5 y estos son comunes para cualquier instrumento que implemente el protocolo, su sintaxis se muestra en la siguiente figura 2.6.

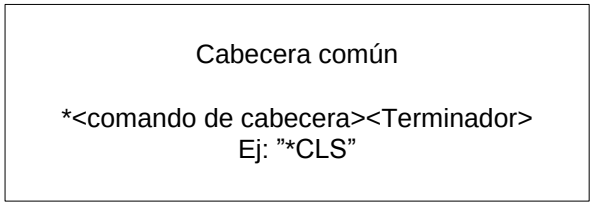


Figura 2.6. Sintaxis de cabecera de comando común para un instrumento con opción GPIB.

CONCLUSIONES DEL CAPITULO II

- ✓ El estándar IEEE488 surgió en el año de 1975 y a evolucionado después de su nacimiento, manteniendo vigencia y funcionalidad hasta la fecha, convirtiéndose en un estándar de comunicación para el control de instrumentos ampliamente utilizado.

- ✓ Un dispositivo conectado al bus GPIB puede estar operando como hablante, escucha o controlador (*talker, listener o controller*).

- ✓ El estándar define un conjunto de comandos, mensajes y funciones de interfaz para establecer la comunicación entre los dispositivos conectados al bus.

- ✓ El bus utiliza lógica TTL inversa y esta formado por 16 líneas; 8 de datos bidireccionales, 5 de manejo del bus y 3 para sincronización.

- ✓ SCPI es una especificación construida sobre el estándar IEEE 488.2 y define una estructura de comandos estándar aceptados por múltiples instrumentos de muchos fabricantes.

- ✓ La función de interfaz AH es implementada por los dispositivos en estado de “escucha” (Listener).

- ✓ La función de interfaz SH es implementada por los dispositivos en estado de “hablante” (Talker).

- ✓ Los mensajes dependientes de dispositivos son específicos para cada instrumentos pero deben cumplir con el estándar IEEE-488.2

REFERENCIAS BIBLIOGRAFICAS

Para mayor información acerca del estándar GPIB, refiérase a los siguientes documentos:

R. ANSI/IEEE Std 488.1-1987. IEEE Standard Digital Interface for Programmable Instrumentation (The Institute of Electrical and Electronics Engineers, Inc. 345 East 47th Street, New Cork, NY 10017,USA).

R. History of GPIB. Breve historia del bus GPIB publicado por *National Instrument* en la siguiente dirección de Internet: (http://zone.ni.com/devzone/conceptd.nsf/webmain/1DF3ADFEBDFB9F77862567C300782C11)

CAPITULO III

DISEÑO Y CONSTRUCCION DE LA SOLUCION

Introducción

Este capitulo describe la selección, diseño, construcción y función de las etapas de todo el sistema GPIB. Se realiza una breve discusión de la elección de los componentes de cada una de las etapas en las que se deriva la solución propuesta.

El capitulo inicia mostrando un esquema en el cual se observan los componentes generales de la propuesta de la solución del sistema GPIB; continuaremos exponiendo las características de hardware y software que poseen los módulos que componen la tarjeta controladora y los elementos usados para dichos módulos. Finalmente se explicaran los componentes de *software* que formaran parte de los servicios con los que cuenta la solución, describiendo sus funciones y características.

3.1 Esquema completo del sistema

En el siguiente esquema se muestra el diseño del sistema describiendo cada uno de sus componentes, tal esquema de la solución propuesta se muestra en la figura 3.1:

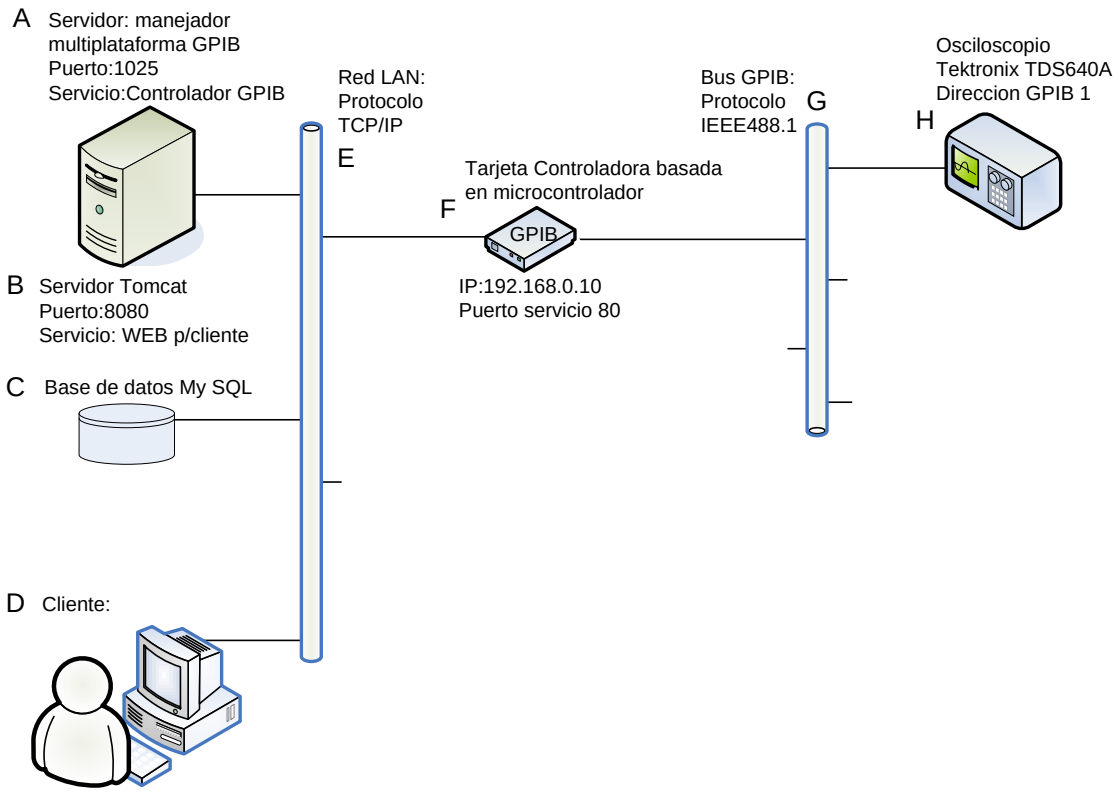


Figura 3.1. Componentes del sistema GPIB

En la Figura 3.1 se muestra la conexión de los elementos que forman parte de la solución, el controlador del sistema GPIB es la “tarjeta controladora” que posee dos módulos, uno para el manejo de la comunicación TCP/IP y otro para implementar la función controlador requerida por el estándar IEEE 488; además el controlador se complementa con los distintos servicios residiendo en un equipo servidor para dar interacción con el cliente.

3.1.1 Especificaciones técnicas de cada componente

La lista siguiente muestra las especificaciones de los componentes mostrados en la figura 3.1

A. Servicio de comunicación con la tarjeta controladora: esta herramienta de software esta desarrollada usando JAVA y presta un servicio de comunicación con la tarjeta controladora por medio del puerto 1025.
B. Servicio de aplicación WEB para el cliente: aplicación Web desarrollada usando JSP(Java Server Page), Java Scrip y código HTML, la cual presta su servicio montado en un servidor Tomcat por medio del puerto 8080.
C. Servicio de base de datos: este presta su servicio en forma local al servicio de comunicación con la tarjeta controladora (literal A) permitiendo
D. Cliente: el usuario debe solicitar el servicio por medio de un navegador como por ejemplo Internet Explorer, Mozilla Firefox, Netscape etc.
E. Red ethernet: red local cumpliendo con la capa física del modelo OSI
F. Tarjeta controladora del bus IEEE-488: presta el servicio de controlar el bus por medio del puerto 80 y direccion IP 192.168.1.87
G. Bus IEEE-488: hardware que implementa los requerimientos físicos del estándar IEEE-488.1
H. Osciloscopio Tektronix TDS640A: osciloscopio de 4 canales con ancho de banda 500MHZ

3.1.2 Funcionamiento del sistema

En la figura 3.2 se muestra una secuencia común en la comunicación o funcionamiento del sistema GPIB propuesto, los 4 círculos representan los elementos principales del sistema y las flechas con un numeral encerrado en un circulo representa el numero de secuencia de comunicación entre dos elementos.

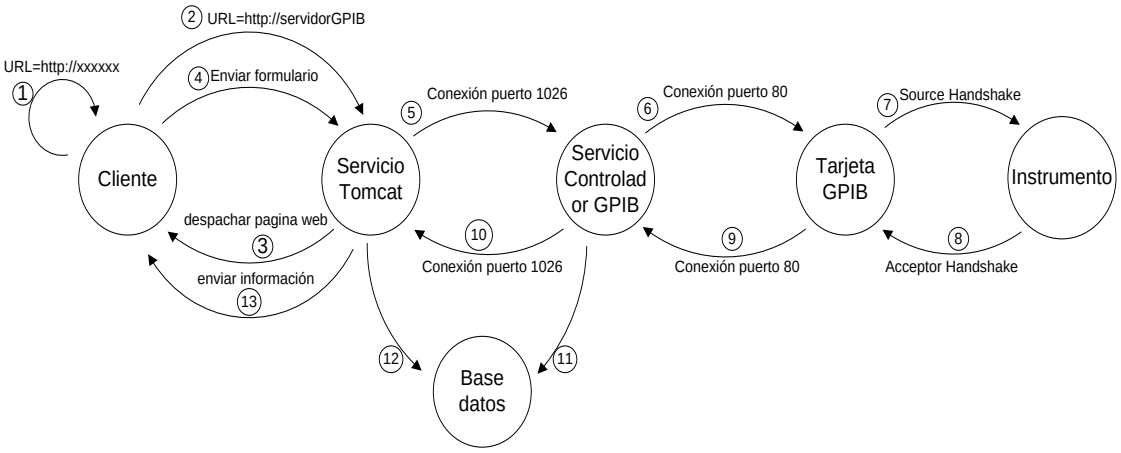


Figura 3.2 muestra el diagrama de flujo de información.

Descripción del flujo de datos entre los componentes de la solución:

1. Si el cliente digita en el URL una dirección que no sea la del servicio http de la solución propuesta no se realiza ninguna acción.

2. El cliente desde una computadora en red con el sistema GPIB escribe en el URL del navegador la dirección del host donde se encuentra el servicio Tomcat prestando servicio en el puerto 8080.

3. El servidor Tomcat despacha una página Web al cliente con formularios para emular los controles del Instrumento.

4. Por medio del servicio Web el cliente envía un formulario donde escoge que opciones configurar del Instrumento.

5. El servicio Tomcat establece comunicación con el servicio controlador GPIB por medio del puerto 1026 y le envía los datos obtenidos del formulario del cliente.

6. El servicio controlador GPIB interpreta los datos obtenidos del cliente y genera la información adecuada para enviarla a la tarjeta GPIB la cual escucha por medio del puerto de comunicación TCP/IP 80.

7. En esta etapa de la comunicación la tarjeta envía por medio de la secuencia Handshake los mensajes dependientes de dispositivo o de interfase al instrumento. si el mensaje es de control la comunicación

termina acá, pero si el mensaje es de petición entonces continúa con los siguientes números de secuencia.

8. La información solicitada al Instrumento es proporcionada y enviada a la tarjeta controladora por medio de la secuencia Handshake a través del Bus GPIB

9. La tarjeta controladora envía la información por medio Ethernet al servicio controlador GPIB por el puerto 80.

10. el servicio manejador GPIB por medio del puerto 1026 le indica al servicio Tomcat que la información solicitada se encuentra en la base de datos

11. El servicio manejador GPIB almacena la información proveniente del bus GPIB en la respectiva tabla de la base de datos.

12. El servicio Tomcat lee de la base de datos la información requerida por el cliente.

13. El servicio Tomcat despacha vía Web la información solicitada al cliente

3.2 Diseño y construcción de la Tarjeta Controladora GPIB

La tarjeta controladora GPIB lleva su nombre por la función principal que realiza y es ser el controlador del bus IEEE 488, el componente principal es el microcontrolador PIC16F877A, la elección de este se hizo por las ventajas que ofrece su bajo costo, su gran difusión y alto rendimiento.
Entre las características específicas de interés de este microcontrolador que podemos mencionar se encuentran:

- Memoria de programa tipo flash
- 8Kbytes de memoria para programa
- 368 bytes de memoria RAM para datos
- 256 bytes para datos en EEPROM
- 35 palabras de instrucción
- 200nseg de ejecución de una instrucción
- 33 pines de entrada/salida
- Reloj de entrada hasta 20MHz

3.2.1 Esquema Eléctrico de la Tarjeta Controladora GPIB

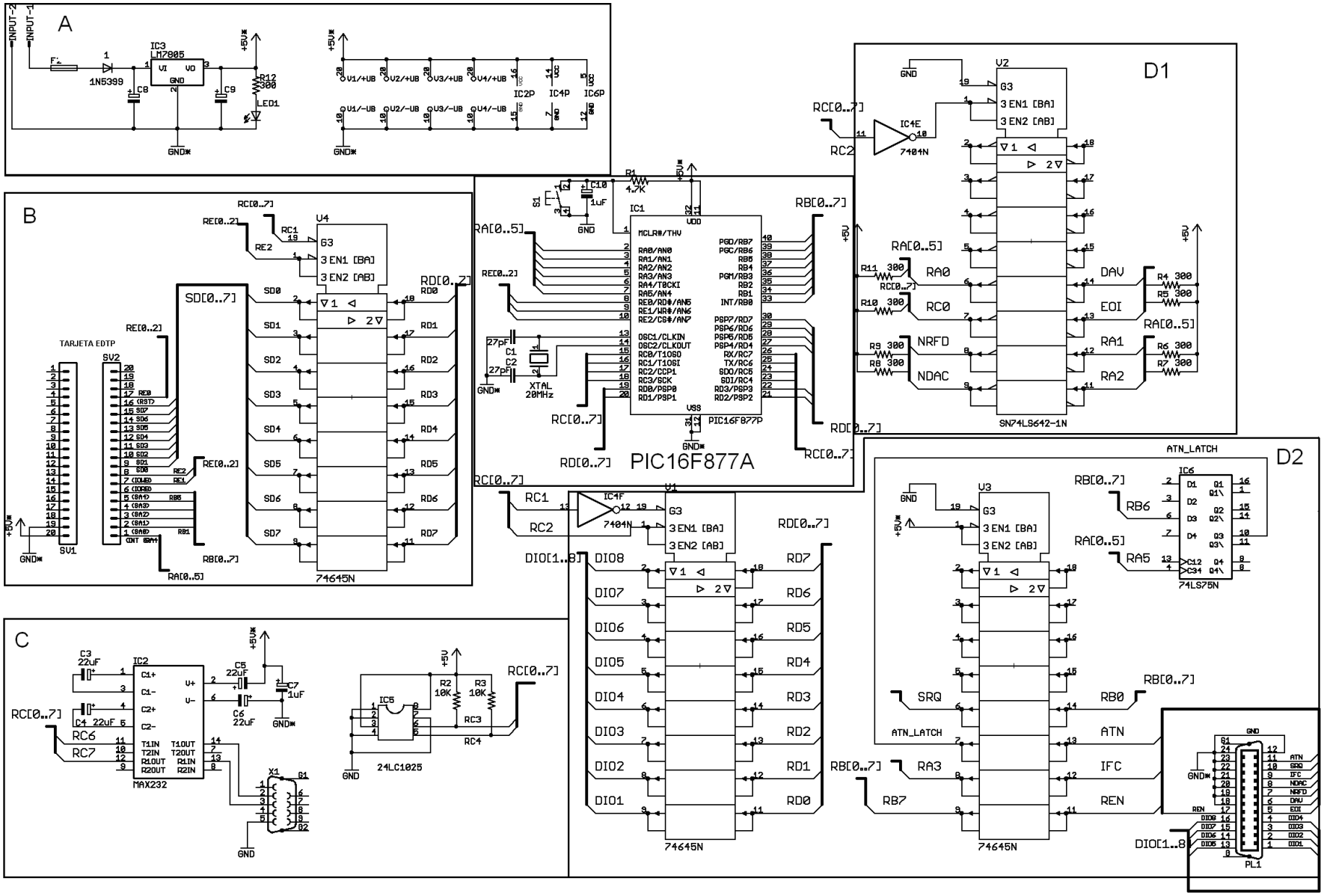


Figura 3.3. Diagrama completo de la tarjeta controladora del bus GPIB.

A continuación se presenta una descripción de la figura 3.3 donde esta se ha dividido por etapas.

- Etapa A: En la area marcada con la letra “A” de la f*igura 3.3* se observa la etapa de alimentación de la tarjeta la cual requiere de un voltaje igual a 5VDC el cual es regulado por el circuito integrado LM7805 a un nivel de voltaje de 5VDC, también dispone de un fusible y un diodo de protección para evitar daños a la tarjeta debido a una conexión con polaridad inversa o una corriente excesiva.

- Etapa B: en la figura se observan los componentes del módulo TCP/IP que esta formado por la parte central el microcontrolador PIC16F877A por la tarjeta EDTP y el circuito integrado SN74LS645-1N (V4).

- Etapa C: En el rectángulo marcado con “C” se observan los componentes MAX232 y una memoria EEPROM (estos son incluidos en el diagrama para una posible mejora de la tarjeta);

- Etapas D1 y D2: En las áreas marcadas con “D1” y “D2” se encuentran los componentes del módulo GPIB que esta compuesto por un chip SN74LS642-1N (V2), dos SN74LS645-1N (V1 y V3), un latch SN74LS75, un inversor SN74LS04 y el conector para el bus GPIB.

La tabla 3.1 muestra la correspondencia de las conexiones de los puertos de entrada salida del microcontrolador con lo buses de los módulos GPIB y TCP/IP.

PIN	NOMBRE	CONECTADO A:	PIN	NOMBRE	CONECTADO A:
1	MCLR#/THV	Señal MCLR	21	RD2/PSP2	transceivers DIO3[1] /transceivers SD2[1]
2	RA0/AN0	Señal DAV del bus GPIB[1]	22	RD3/PSP3	transceivers DIO4[1] /transceivers SD3[1]
3	RA1/AN1	Señal NRFD del bus GPIB[1]	23	SDI/RC4	Señal SDA para el bus I2C
4	RA2/AN2	Señal NDAC del bus GPIB[1]	24	SDO/RC5	Pin libre
5	RA3/AN3	Señal IFC del bus GPIB[1]	25	TX/RC6	Señal Tx para el chip MAX232
6	RA4/T0CKI	Señal INT0 de la tarjeta EDTP	26	RX/RC7	Señal Rx para el chip MAX232
7	RA5/AN4	Habilitador del latch para Señal ATN	27	PSP4/RD4	transceivers DIO5[1] /transceivers SD4[1]
8	RE0/RD#/AN5	Señal RST de la tarjeta EDTP	28	PSP5/RD5	transceivers DIO6[1] /transceivers SD5[1]
9	RE1/WR#/AN6	Señal IORB de la tarjeta EDTP	29	PSP6/RD6	transceivers DIO7[1] /transceivers SD6[1]
10	RE2/CS#/AN7	Señal IORW de la tarjeta EDTP	30	PSP7/RD7	transceivers DIO8[1] /transceivers SD7[1]
11	VDD	+ 5VDC	31	VSS	GND
12	VSS	GND	32	VDD	+ 5VDC
13	OSC1/CLKIN	Señal del cristal 20 MHz.	33	INT/RB0	Señal SRQ del bus GPIB[1]

Tabla 3.1. Distribución de pines del microcontrolador.

[1] *Estas señales no se conectan directamente al microcontrolador, estas pasan a través de los transceivers antes de conectarse al bus.*

PIN	NOMBRE	CONECTADO A:	PIN	NOMBRE	CONECTADO A:
14	OSC2/CLKOUT	Señal del cristal 20 MHz.	34	RB1	Línea SA0 de la tarjeta EDTP
15	RC0/T1OSO	Señal EOI del bus GPIB[1]	35	RB2	Línea SA1 de la tarjeta EDTP
16	RC1/T1OSI	Bit para crear la configuración de los transceivers.	36	PGM/RB3	Línea SA2 de la tarjeta EDTP
17	RC2/CCP1	Bit para crear la configuración de los transceivers.	37	RB4	Línea SA3 de la tarjeta EDTP
18	RC3/SCK	Señal SCL para el bus I2C	38	RB5	Línea SA4 de la tarjeta EDTP
19	RD0/PSP0	transceivers DIO1[1] /transceivers SD0[1]	39	PGC/RB6	Señal ATN antes del latch
20	RD1/PSP1	transceivers DIO2[1] /transceivers SD1[1]	40	PGD/RB7	Señal REN del bus GPIB[1]

Tabla 3.1 (continuación). Distribución de pines del microcontrolador.

Ya que los pines RC1 y RC2 del puerto C son utilizados para el control del conjunto de *transceivers* V1, V2 y V4. Existen cuatro posibles configuraciones y estas son descritas en la Tabla 3.2.

FUNCIÓN EN MÓDULO TCP/IP		FUNCIÓN EN MÓDULO GPIB	RE1 (IORB)	RE2 (IORW)	RC1	RC2
1	Alta impedancia	**SH función *source handshake***	X	X	H	L
2	Alta impedancia	**AH función *aceptor handshake***	X	X	H	H
3	**Lectura al RTL**	Alta impedancia	L	H	L	X
4	**Escritura al RTL**	Alta impedancia	H	L	L	X

X: condición no importa. L: nivel bajo (0 volts). H: nivel alto (5 volts).

Tabla3.2. Configuraciones de control para el conjunto de transceivers V1, V2 y V4.

El microcontrolador habilita solo una de estas configuraciones a la vez según la operación que se este realizando en el programa, por ejemplo si la tarjeta esta usando el modulo GPIB está debe usar la configuración 1 para cuando se necesite realizar las funciones *talker* ó configuración 2 para la funcion *listen.* De otra manera si la tarjeta usa el módulo TCP/IP, esta puede usar la configuración 3 cuando se realiza una lectura ó la configuración 4 cuando se realiza una escritura a los registros del RTL.

3.2.2 Software controlador del microcontrolador

El diagrama de flujo mostrado en la figura 3.4 corresponde al programa principal de la tarjeta controladora el cual utiliza los módulos expuestos en las secciones anteriores. Más adelante se describe cada uno de los procesos que implica el funcionamiento de este programa escrito en código ensamblador para el PIC16F877A.

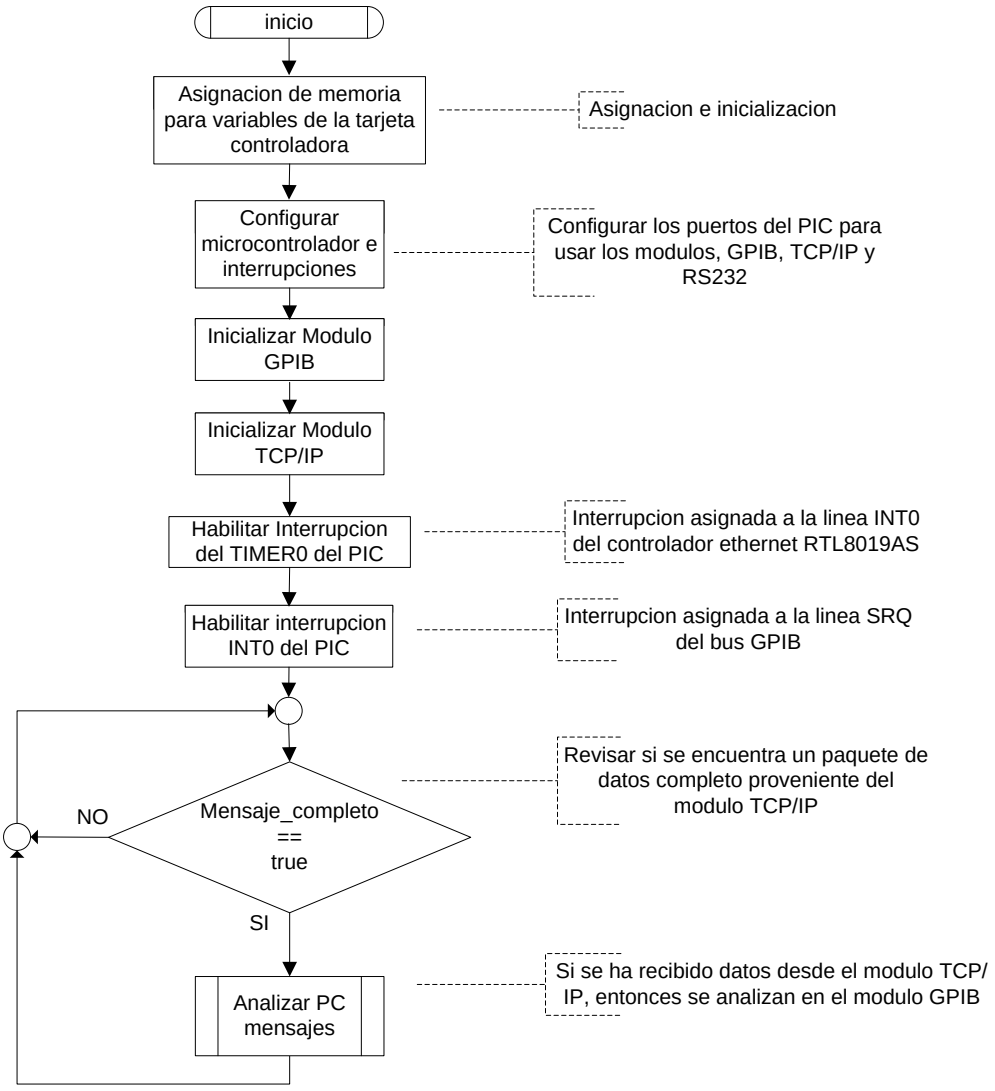


Figura 3.4. Diagrama de flujo del programa principal del PIC16F877A

Descripción del programa principal.

1. En un principio se reserva espacio e inicializa en la memoria de datos del microcontrolador para las variables de los módulos a utilizar.

2. Se configuran los puertos de entrada/salida del PIC asignados a las líneas de cada modulo así también las líneas a usar para manejar los manejadores de bus (Transceivers) y además se configura la interrupción del modulo fuente de interrupción externa INT la cual esta conectada a la línea del bus GPIB SRQ (petición de servicios) y la interrupción del "timer0" del PIC para atender una interrupción por parte del RTL8019AS (que indica un paquete completo recibido en buenas condiciones).

- La inicialización del modulo GPIB permite que la tarjeta controladora tome el control del bus poniendo las líneas en un estado conocido y listo para que la tarjeta controladora realice su función.

- La inicialización del modulo TCP/IP configura e inicializa el controlador ethernet rtl8019as de modo que después de este proceso la tarjeta controladora es reconocida en la red a la que se encuentre conectada.

- Una vez inicializados todos los módulos de la tarjeta controladora se habilita la interrupción para poder atender la petición de servicio (SRQ) por parte de los dispositivos conectados al bus GPIB.

- se habilita la interrupción para poder atender la recepción de paquetes TCP-IP por parte del cliente conectado a la red.

- La tarjeta entra en un estado de espera por la interrupción de la línea SRQ de algún dispositivo del bus y en otro estado de censo del modulo TCP/IP para la recepción de un paquete completo.

- Si se produce una interrupción esta es tratada con la rutina "interrupciones" dentro de la cual se revisa si esta ha sido producida por el modulo INT, de ser así se atiende a la petición de servicio de cualquier dispositivo como indica el estándar IEEE488.1, o si es una interrupción del temporizador "TIMER0"

- Si se ha recibido un paquete completo desde el modulo TCP/IP se pone a disposición del modulo GPIB para que por medio de la rutina analizar_PCmensajes, este los interpreta y ejecuta las ordenes enviadas.

En la figura 3.5 se muestra la subrutina de atención a las interrupciones del microcontrolador.

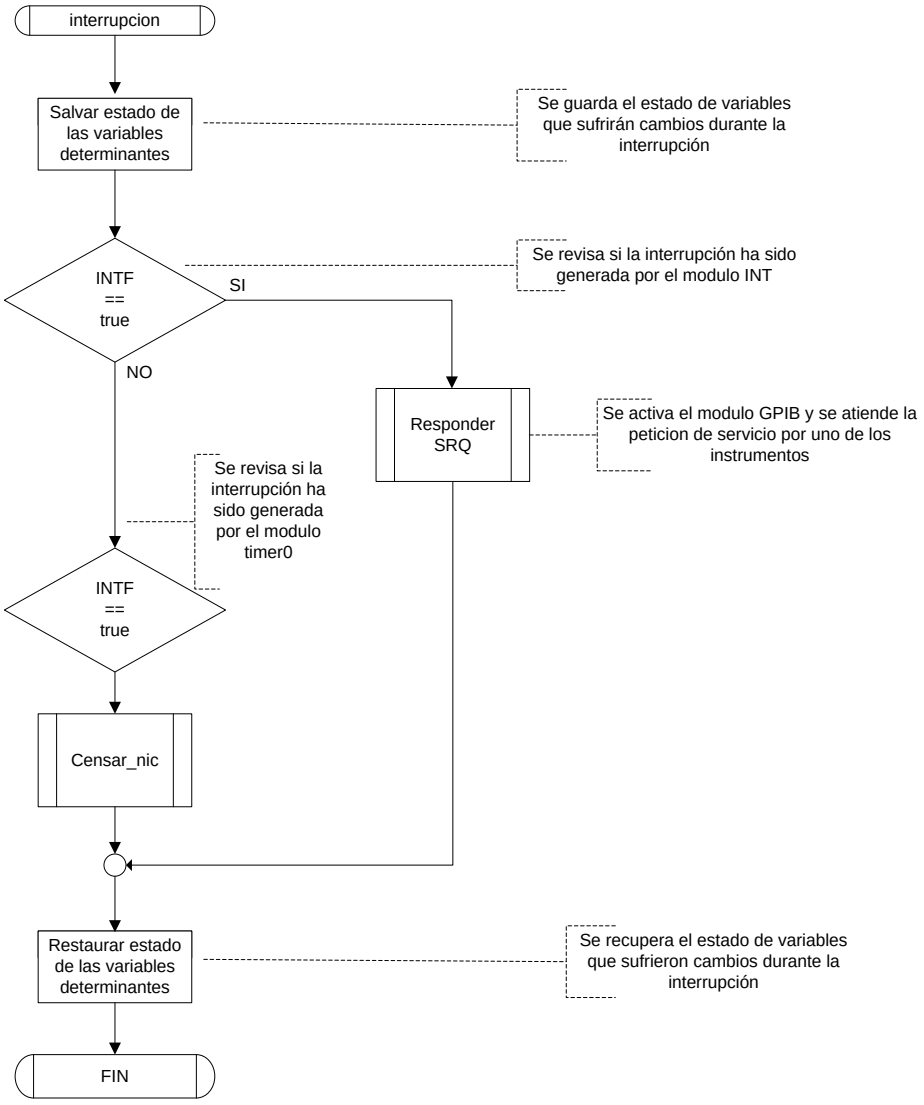


Figura 3.5. Subrutina que atiende las interrupciones del microcontrolador PIC.

3.2.3 Diseño y construcción del módulo TCP/IP

La función de este módulo es la de establecer comunicación entre la tarjeta controladora GPIB y el software que se ejecuta en un equipo servidor de paginas Web, para tal fin este modulo implementa el protocolo TCP/IP para trasmitir y recibir datos.

Para implementar el protocolo de comunicación TCP/IP se utilizara el controlador de ethernet RTL8019AS (este se encuentra dentro de uno de los componentes del sistema, la tarjeta *EDTP* de cual se detallara mas adelante) que implementa la capa física en el modelo OSI, El resto de capas es implementado por software en la memoria del microcontrolador *PIC16F877A*.

Entre las características del controlador Ethernet tenemos las siguientes:

- 16Kbytes de SRAM.
- Soporta Plug and Play auto-detección de UTP, AUI y BNC.
- Soporta 8 líneas de interrupción.
- Soporta full-duplex Ethernet.

3.2.3.1 Módulo TCP/IP

Su función es establecer comunicación entre el microcontrolador y el servicio de control de la tarjeta *GPIB* que residirá dentro del equipo servidor, para dicho fin el modulo implementa el protocolo de Internet TCP/IP.

Este modulo consta de *hardware* y su respectivo *firmware*, este ultimo es el programa escrito en lenguaje ensamblador del microcontrolador y su objetivo es el de formar paquetes de datos correspondientes al Protocolo TCP/IP para la transmisión y desempaquetar las tramas en el caso de la recepción; también pone a disposición del modulo GPIB los datos recibidos del servidor para que este haga su correcta interpretación.

El hardware de este modulo se encarga de cumplir con los requerimientos físicos del protocolo *TCP/IP* y se hace por medio de un tarjeta llamada *EDTP* la cual incluye el circuito integrado *RTL8019AS* que es un controlador ethernet y otros componentes adicionales para su funcionamiento.

3.2.3.2 Componentes del Módulo TCP/IP

Los componentes con los que cuenta este modulo TCP/IP son:

- **Conector *RJ45***: para conectar la tarjeta controladora a una red ethernet.

- **Controlador ethernet *RTL8019AS*:** este es utilizado para implementar el protocolo TCP/IP, para enviar y recibir los paquetes de datos. Este esta contenido en la tarjeta EDTP y en conjunto con el PIC16F877A forman el hardware de este módulo TCP/IP cuyo circuito se muestra en la *Fig 3.6*

- **Puertos de entrada salida del PIC16F877A**: la asignación de las lineas del microcontrolador para el modulo TCP/IP se lista en la siguiente *tabla 3.3*

Líneas PIC	Descripción
Puerto D	Puerto multiplexado para funcionar como bus de datos del modulo GPIB y del modulo TCP/IP
RB1-RB5	Asignado al bus de dirección del microcontrolador RTL8019AS
Puerto E	Asignado el bus de control del microcontrolador RTL8019AS

Tabla 3.3: relación de líneas del PIC16F877A y el modulo TCP/IP

- **Manejador TCP/IP:** este es un código de programa en ensamblador para el PIC16F877A que se encarga de interactuar con el RTL8019AS (tarjeta EDTP); entre sus funciones están revisar si el paquete de datos entrante es para la tarjeta controladora, luego decodifica los paquetes recibidos desde la red para ponerlos a disposición del modulo GPIB, también se encarga de construir paquetes con los datos provenientes del modulo GPIB que se enviaran por la red ethernet hacia el servidor de control de la tarjeta. Para una descripción del funcionamiento del RTL8019AS revisar las hojas de datos en el Disco Compacto de este trabajo.

3.2.3.3 Controlador Ethernet RTL8019AS

Este controlador posee un bus de direcciones uno de datos y uno de control para poder realizar su respectiva configuración o tener acceso al tráfico de la red.

- Bus de direcciones SA0-SA4 por medio del cual se direccionan los registros de control y las posiciones de memoria de los buffer de entrada y salida para el tráfico de red.

- Bus de datos el SD0-SD7 el cual es de escritura/lectura en donde se escribe o lee el valor en una configuración de registros o en el buffer de entrada/salida.

- Bus de control: esta formado por dos líneas de señal IOWB y IORB, la primera se utiliza para indicar una operación de escritura en el bus de datos apuntado por la dirección presente en el bus de direcciones, la segunda indica una operación de lectura al bus de datos.

3.2.3.4 Manejador del módulo TCP/IP.

El manejador o *firmware* TCP/IP esta formado por un conjunto de rutinas que forman parte del programa ejecutado por el microcontrolador PIC16F877A, estas rutinas implementan los protocolos TCP/IP para la comunicación de la tarjeta con el ordenador servidor, estas son descritas en la Tabla 3.4 que se muestra a continuación.

NOMBRE	ENTRADAS	DESCRIPCION	SALIDAS
Inicializar_tarjeta_NIC	(ninguna)	Inicializa tarjeta controladora ethernet (EDTP), configura parámetros como dirección MAC y la deja lista para recepción y transmisión de datos.	(ninguna)
Inport	REGISTRO: tiene la dirección del registro a leer.	Lee los registros del RTL apuntados por la variable REGISTRO y pone el resultado obtenido en la variable VALOR.	VALOR: tiene el valor leído del registro del RTL.
Outport	REGISTRO: contiene la dirección del registro a escribir. VALOR: contiene el dato a escribir en el registro.	Escribe el contenido de la variable VALOR en el registro apuntado por la variable REGISTRO.	(ninguna)
remote_dma_setup	OPERACIÓN: indica la operación a efectuarse: escritura o lectura. DIRECCION: Dirección.	Configura el DMA de forma remota para lectura o escritura.	(ninguna)
censar_nic	(ninguna)	Censa el estado de la tarjeta NIC, si esta ha recibido un paquete lo almacena en BUFFER_ENTRADA hasta completar un mensaje. Cuando un mensaje completo esta presente en BUFFER_ENTRADA se indica usando la variable MENSAJE_COMPLETO	MENSAJE_COMPLETO: Esta variable indica si hay un mensaje listo para ser procesado por el modulo GPIB.

Tabla 3.4. Funciones del módulo TCP/IP.

NOMBRE	ENTRADAS	DESCRIPCION	SALIDAS
leer_paquete	(ninguna)	Lee todos los campos de un paquete y determina que tipo de paquete a llegado luego llama a la rutina correspondiente para tratar el tipo de protocolo correspondiente.	(ninguna)
arp_response	(ninguna)	Esta rutina responde a quien lo solicita con la dirección MAC de la tarjeta controladora.	(ninguna)
load_ethernet_header	source_address physical_address	Carga en el buffer del RTL la cabecera ethernet que es formada por las direcciones MAC de destino y MAC de la tarjeta controladora.	(ninguna)
send_packet	LEN: longitud del paquete a enviar. XMT_BUF_START: apunta a la posición de memoria del RTL donde se encuentra el paquete a enviar.	Le ordena al RTL que envié el paquete de longitud LEN apuntado por XMT_BUF_START	(ninguna)
case_TCP	(ninguna)	Lee y almacena los datos del paquete entrante TCP.	(ninguna)
case_LISTEN	html_socket: estado del socket.	Es utilizada para establecer el socket TCP con la PC que lo solicita.	(ninguna)
case_RCVD	(ninguna)	Es utilizada para establecer el socket TCP con la PC que lo solicita.	(ninguna)
case_ESTAB	(ninguna)	Es utilizada para establecer el socket TCP con la PC que lo solicita.	BUFFER_ENTRADA: MENSAJE_COMPLETO:
tcp_listen	(ninguna)	Responde con un SYN y un ACK para establecer el socket.	(ninguna)
tcp_syn_rcvd	(ninguna)	Coloca a la tarjeta en un estado de sincronización para establecer el socket.	(ninguna)
tcp_estab	(ninguna)	Coloca a la tarjeta en un estado de socket establecido, en este estado se pueden recibir y enviar datos.	(ninguna)
load_IP_header	IP_packet_length0, LSB IP_packet_length1, MSB IP_send_protocol	Carga la cabecera IP en el buffer de la memoria del RTL.	(ninguna)
load_tcp_header	data_flags1 data_flags0 tcp_chksum1 tcp_chksum0 tcp_length1 tcp_length0	Carga la cabecera TCP en el buffer de la memoria del RTL.	(ninguna)
Sumar	numnero1_1: numero1_0: numnero2_1: numero2_0	suma dos números de dos bytes: numnero1 + numnero2 = resultado	resultado0: resultado1:
Restar	numero1: numero2:	Efectúa la resta de numero1 – numero2	resultado0: resultado1:
calcular_check	numero1: numero2:	Realiza el cálculo del checksum.	Chksum:
chksum_buffer_salida	BUFFER_SALIDA:	Calcula el checksum de los datos contenidos en el BUFFER_SALIDA que la tarjeta envía a la PC.	(ninguna)
cargar_mensaje	BUFFER_SALIDA:	Carga en la memoria del RTL los datos a enviar a la red.	(ninguna)

Tabla 3.4 (continuación). Funciones del módulo TCP/IP.

3.2.3.5 Esquema eléctrico del módulo TCP/IP.

La Figura 6.1 muestra los componentes electrónicos que conforman al módulo TCP/IP, aquí se observan tres etapas: la primera formada con el microcontrolador PIC16F877A, el bus transceivers (V4) que es un manejador de líneas de señal de tres estados (SN74LS645-1N) y la tarjeta EDTP.

Se hace uso de los puertos PORTC, PORTD y PORTE del microcontrolador para controlar el funcionamiento de este modulo. El puerto D es utilizado como un bus de datos de ocho bits para intercambiar información entre el microcontrolador y la tarjeta EDTP esto se logra usando entre ellos el chip SN74LS645-1N (V4) que se usa para la transferencia de datos entre los buses, este ultimo componente es utilizado debido por la necesidad de utilizar el bus de datos (PORTD) del microcontrolador que es común para los módulos TCP/IP y GPIB (2 chips SN74LS645-1N V1 y V4 se encargan de que el flujo de datos sea dirigido solo a uno de los módulos a la vez).

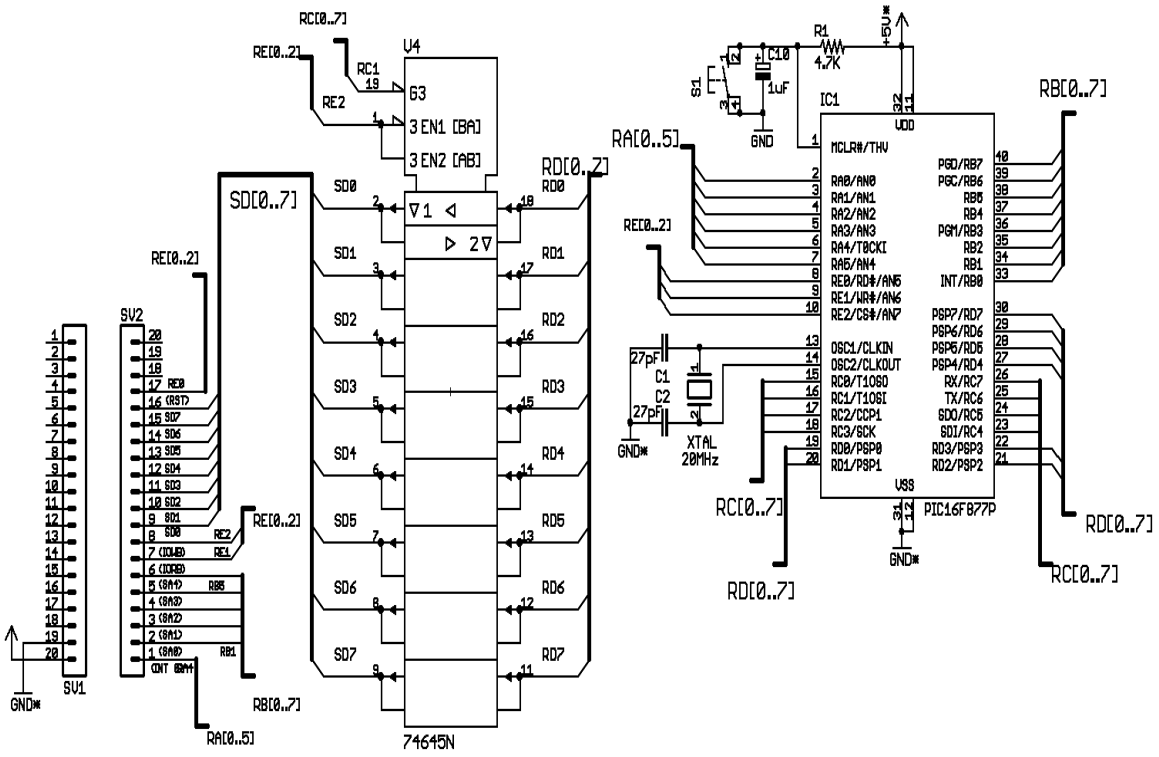


Figura 3.6. Diagrama del modulo TCP/IP.

El puerto C es utilizado para crear la configuración de control adecuada para el conjunto de *transceivers* (un total de 4 están presentes en la tarjeta controladora) esta configuración permite habilitar a estos *transceivers* con las direcciones adecuadas para la transferencia de datos según se ejecute una operación de lectura o escritura en la tarjeta EDTP. Los cinco bits RB1 a RB5 del puerto B son utilizados como el bus de direcciones que requiere la tarjeta EDTP para su funcionamiento. El puerto E proporciona las señales de lectura, escritura y *reset* requeridas para el funcionamiento de la tarjeta EDTP.

3.2.4 Diseño y construcción del módulo GPIB

Este modulo se encarga de controlar los instrumentos conectados al bus GPIB. El componente principal de este modulo es el microcontrolador PIC16F877A, para implementar las funciones básicas del estándar se ha creado el software controlador en el PIC, el cual ha sido escrito usando el lenguaje ensamblador y asistido por el editor de programas MPLAB IDE que es proporcionado de forma gratuita por MICROCHIP.

Entre las funciones del controlador GPIB se encuentran las siguientes:

- Este modulo toma los datos que el modulo TCP/IP pone a su disposición, los interpreta y luego realiza la operación indicada; entre estas operaciones tenemos:
    - Comandos para configurar algún dispositivo.
    - Comandos para controlar el bus.
- Toma datos provenientes del bus GPIB y los pone a disposición del modulo TCP/IP para que estos sean enviados al servidor vía TCP/IP, el significado de algunos de estos datos pueden ser:
    - Muestras de una señal de onda.
    - Resultados de una medición.
    - Datos que muestran la configuración remota del dispositivo.

3.2.4.1 Componentes del Módulo GPIB

El modulo GPIB se desglosa en los siguientes componentes:

HARDWARE:

A. Conector (*female IEEE-488 conector, right angle pc terminal*) Amphenol CHAMP o cinch 57 MicroRibbon: para conectar la tarjeta controladora al BUS GPIB.

B. Bus Transceivers: estos son chips de la familia TTL SN74LS642-1N y SN74LS645-1N utilizados para acoplar las líneas del microcontrolador con el bus GPIB, estos están diseñados para la comunicación asíncrona entre buses de datos en dos direcciones y proveen los niveles de voltaje y corriente requeridos por el BUS GPIB (ver capitulo 2), información adicional de estos chip es mostrada en los anexos.

C. Inversores TTL SN74LS04 usado para crear la combinación lógica de los pines de dirección y habilitadores de los transceivers.

D. Latch TTL SN74LS75 utilizado para mantener el estado de la línea ATN evitando que las pequeñas variaciones del nivel de voltaje afecten el estado de esta línea.

E. Puertos de entrada/salida del PIC16F877A

SOFTWARE:

- Manejador del modulo GPIB: este es un código de programa en ensamblador para el PIC16F877A que contiene todas las características funcionales básicas descritas en el estándar IEEE 488.1 que requiere un controlador de BUS GPIB, además en conjunto con el manejador del modulo TCP/IP se encarga de enviar y recibir mensajes comunicándose así con la aplicación web en el servidor. En la siguiente sección se muestran las funcionalidades y descripción de este software.

3.2.4.2 Software Manejador del modulo GPIB.

El manejador o firmware de la tarjeta controladora se encarga de implementar la función *Controller* del estándar IEEE-488.1 y la secuencia de negociación requerida por el estándar para el intercambio de datos con los dispositivos.

En la siguiente tabla se listan las rutinas con las que cuenta el manejador de la tarjeta GPIB. Estas funciones solo son usadas por el manejador de la tarjeta.

NOMBRE	ENTRADAS	DESCRIPCION	SALIDAS
configurar_microcontrolador	(ninguna)	Configura al microcontrolador y pone en un estado conocido todos los puertos de este.	(ninguna)
SH	DATO_GPIB: tiene el dato a enviar por las líneas de señal DIO.	"*Source handshake*" envía hacia el bus según el estándar IEEE 488.1 el byte contenido en la variable DIODATO.	.
AH	(ninguna)	"*Aceptor handshake*" recibe un dato proveniente del bus GPIB.	DIODATO: tiene el dato recibido.
enviar_mensajes	BUFFER_ENTRADA: contiene la cadena de datos a enviar al BUS. TIPO: indica el tipo de dato a enviar, comando de interfaz o de dispositivo. DIRECCION: indica a que dispositivo se enviara el mensaje. PETICION: indica si el comando es una petición de datos. NLINEAS indica el número de líneas que la tarjeta controladora recibirá desde el bus GPIB	Configura al controlador como emisor y envía una cadena de caracteres utilizando la función SH a un dispositivo o a la interfaz dependiendo del tipo de dato. Si el comando es una petición entonces se llama a recibir_datosGPIB para recibir los datos solicitados.	.
recibir_datosGPIB	DIRECCION: indica a que dispositivo enviara el mensaje. NLINEAS: indica el número de líneas que la tarjeta controladora recibirá desde el bus GPIB.	Configura al controlador como receptor y recibe los datos provenientes de un dispositivo utilizando la función AH.	BUFFER_SALIDA: tiene la cadena de datos recibida.
enviar_IFC	(ninguna)	Envía el mensaje de interfaz IFC, cuyo fin es limpiar la interfaz.	(ninguna)
enviar_REN	(ninguna)	Envía el mensaje REN, con el fin de habilitar el control remoto de un dispositivo.	(ninguna)
enviar_LLO	(ninguna)	Envía el mensaje LLO, con el fin que un dispositivo no responda a sus controles locales y si a controles remotos.	(ninguna)
enviar_DCL	(ninguna)	Envía el mensaje DCL, "*device clear*", para ponerlo en un estado de configuración conocido.	(ninguna)
enviar_GET	(ninguna)	Envía el mensaje GET, con el fin que un dispositivo ejecute su operación para la cual se ha configurado.	(ninguna)
enviar_UNL	(ninguna	Envía el mensaje UNL.	(ninguna)

Tabla 3.5. Rutinas del módulo GPIB que implementan el estándar IEEE488.1.

NOMBRE	ENTRADAS	DESCRIPCION	SALIDAS
enviar_UNT	(ninguna)	Envía el mensaje UNT.	(ninguna)
responder_SRQ	(ninguna)	Atiende la petición de servicio haciendo el llamado a la función encuesta_serie	(ninguna)
encuesta_serie	(ninguna)	Realiza una encuesta serie a los dispositivos en la interfaz para determinar que dispositivo ha activado la línea SRQ.	STB: "*Status Byte*", contiene el byte de estado del dispositivo DIRECCION_CENSO: tiene la dirección del dispositivo que pide atención.

Tabla 3.5 (continuación). Rutinas del módulo GPIB que implementan el estándar IEEE488.1.

3.2.4.3. Esquema eléctrico del módulo GPIB.

La *Figura 3.7* muestra los dispositivos electrónicos utilizados para formar el módulo GPIB la cual esta dividida en cuatro partes que se listan a continuación.

A. en esta etapa se observa el microcontrolador PIC16F877A (IC1) y el pulsador S1 usado para restablecer (RESET).

B. Un chip SN74LS642-1N (V2), un inversor SN74LS04 (IC4).

C. Un SN74LS645-1N (V1 y V3) un inversor SN74LS04 (IC4), y un latch SN74LS75 (IC6).

D. Un SN74LS645-1N (V1 y V3) y un latch SN74LS75 (IC6).

E. El conector para el bus GPIB (PL1)

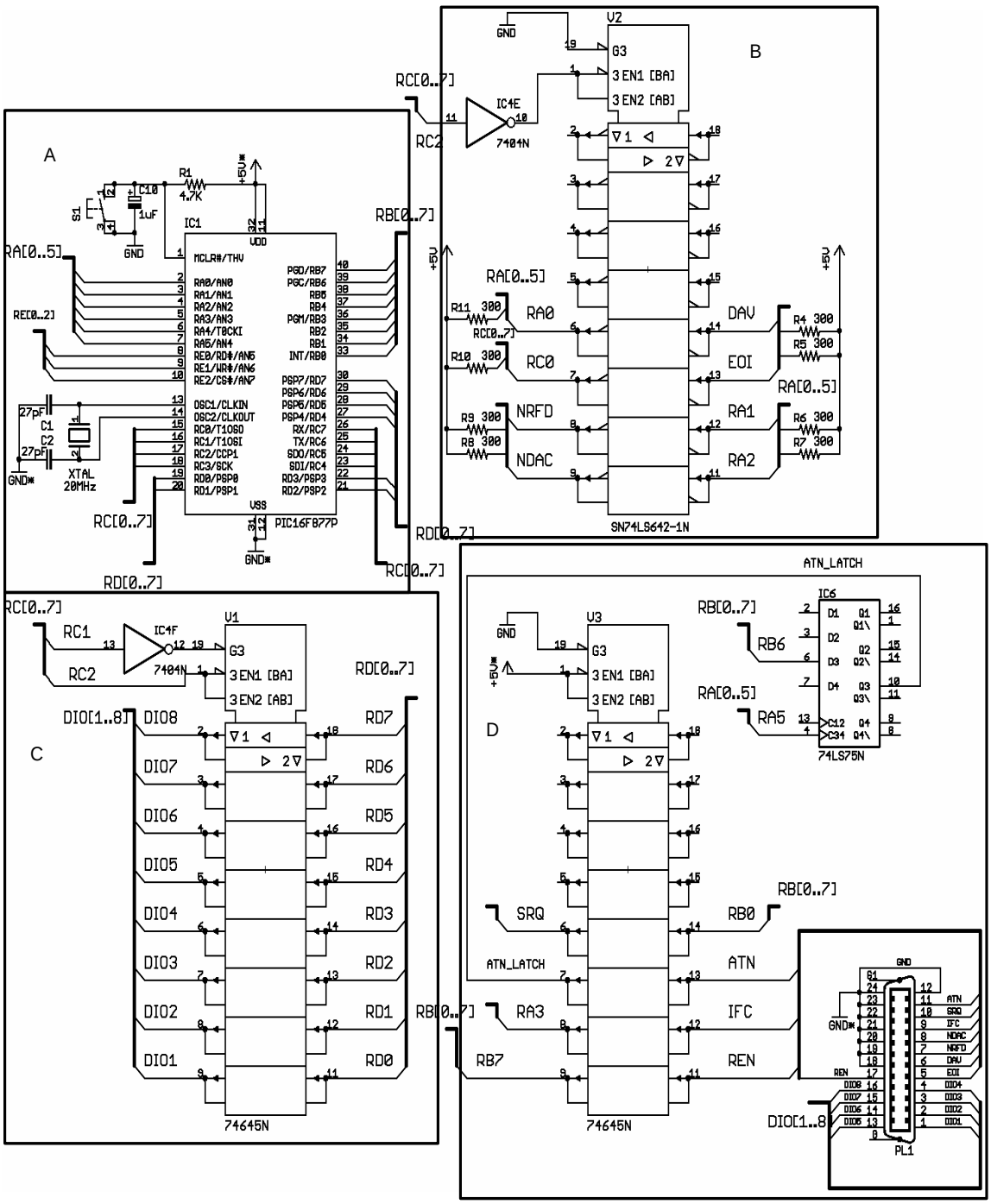


Figura 3.7 Diagrama del modulo GPIB.

3.2.4.4. Descripción de etapas del módulo GPIB.

En la figura 3.7 se observan cuatro etapas del modulo GPIB las cuales se describen a continuación:

- Etapa A: el puerto D del microcontrolador junto con la etapa C es usado como el bus de datos (DIO1-DIO8) del bus GPIB. Además Los pines RC1 Y RC2 del puerto C son utilizados para crear la configuración de control adecuada para el conjunto de *transceivers,* tales configuraciones permiten habilitar a estos *transceivers* con las direcciones adecuadas para la transferencia de datos según lo requiera la configuración del módulo GPIB (modo *talker* o modo *listen*), ver las configuraciones posibles en la tabla 3.2 (seccion 3.2.1)
- Etapa B: El puerto A del microcontrolador (Etapa A) en conjunto con el SN74LS642 es utilizado para proporcionar las señales *handshake* del bus GPIB.
- Etapa C: esta etapa consta del integrado 74645N que conjunto con la etapa A cumplen la función de proporcionar el bus de datos para el bus GPIB.
- Etapa D: el puerto B del microcontrolador (etapa A) junto con los elementos de esta etapa un 74645N forman el bus de control del bus GPIB. Se observa también en la figura que la línea ATN pasa a través de un lacth del chip SN74LS75 para darle estabilidad a esta señales pues esta es crítica para el funcionamiento del bus GPIB debido a que una pequeña variación en el nivel de esta señal puede sacar al instrumento de su estado de remoto o generar un error en su funcionamiento. También El conector para el bus GPIB (*female IEEE-488 conector, right angle pc terminal*) es del tipo Amphenol CHAMP o cinch 57 MicroRibbon con la configuración de pines de la tabla 3.6

LINEAS DE DATOS	PIN	LINEAS DE CONTROL	PIN	LINEAS DE SINCRONIZACION	PIN
DIO1	1	IFC	9	DAC	6
DIO2	2	REN	17	NRFD	7
DIO3	3	ATN	11	NDAV	8
DIO4	4	SRQ	10		
DIO5	13	EOI	15		
DIO6	14				
DIO7	15				
DIO8	16				

Tabla 3.6. Listado de las líneas de señal del conector GPIB.

3.3 Servicio de control de la Tarjeta GPIB

Este servicio es utilizado para aportar comunicación con la tarjeta controladora La implementación de esta etapa se realiza junto con una computadora configurada como servidor, se ha seleccionado el lenguaje de programación Java ya que es de libre distribución. Este software hecho en Java se pretende combinar con una Aplicación Web la cual permita la simulación de estar operando los controles del Osciloscopio Tektronix ya sea para configurarlo, realizar mediciones u obtener los datos de una medición.

Esta aplicación se comunica con otros servicios mostrados en la figura 3.1, (los cuales se tratan en unas secciones adelante) según la tarea que se este realizando, estos servicios son:

- Un servicio Web el cual despliega una página Web que permite emular el osciloscopio y permitir escoger entre los controles del instrumento ya sea para configurarlos o para realizar mediciones.

- El servicio que presta la tarjeta controladora por medio del puerto 80 utilizado para la transferencia de comandos de dispositivos.

- Un servicio de almacenamiento en base de datos MySQL.

3.3.1 Descripción del software del servicio controlador de la tarjeta GPIB

En esta sub-sección se describe el programa escrito en java encargado de establecer comunicación con el servicio que atiende al cliente y la tarjeta controladora. Este código se divide en dos etapas como se muestra en la *figura 3.8.* una de carácter general orientada a establecer comunicación entre la tarjeta controladora y el cliente y otra etapa orientada a manejar los controles que el usuario quiere controlar en forma remota del osciloscopio Tektronix TDS640A, como se había mencionado antes esta etapa de software es especifica para el instrumento a manejar.

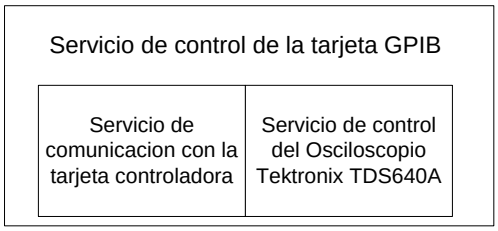


Figura 3.8: etapas del servicio de control de la tarjeta GPIB

3.3.2 Etapa de comunicación con la tarjeta GPIB

La tarjeta tiene manejadores para realizar su función de controlador, pero hay una porción de estos manejadores que son orientados a establecer comunicación con el servicio de comunicación con la tarjeta controladora, para esto hay una serie de mensajes que la tarjeta recibe desde el servicio y otros que la tarjeta debe enviar para el intercambio de información entre estos dos dispositivos que en conjunto realizan el papel de controlador.

3.3.2.1 mensajes que se envían desde el servicio controlador a la tarjeta GPIB

Para poder comunicar el servicio de control con la tarjeta GPIB se disponen de los mensajes mostrados en la tabla 3.1, estos mensajes son ordenes para la tarjeta y son enviados hacia la tarjeta a través del puerto *ethernet* con el objeto de iniciar o ejecutar alguna de las funciones mostradas en la tabla 3.1. La secuencia con la que estos mensajes serán enviados depende de la acción que se requiera ejecutar.

La estructura de estos mensajes es la siguiente:

1 BYTE	VARIOS BYTES
MENSAJE PCT	CADENA DE DATOS

Donde el campo “mensaje PCT” tiene uno de los valores mostrados en la siguiente tabla 3.1

Mensaje PCT	Nombre	Código	Objetivo
INIT	inicializar tarjeta	01	Ordena a la tarjeta controladora inicializarse, configurarse como controlador y colocarse en un estado conocido.
TEM	Tarjeta Enviar Mensajes	02	Ordena a la tarjeta iniciar una secuencia para mandar comandos de interfaz o comandos de dispositivo al bus.
TEML	Tarjeta Enviar Mensaje de Línea	03	Indica a la tarjeta enviar un mensaje de interfaz unilínea o multilínea hacia el bus, ver lista de mensajes en Tabla 3.2.
TRD	Tarjeta Recibir Datos	04	Indica a la tarjeta controladora colocarse en un estado aceptor de datos provenientes del bus GPIB
PDR	Programar Dispositivo en Remoto	05	Indica a la tarjeta controladora programar un dispositivo en modo remoto o en modo local

Tabla 3.7. Mensajes enviados del servidor a la tarjeta.

En la siguiente tabla se muestran los comandos que se pueden enviar con el mensaje TEML.

mensaje TEML	Código	Descripción
EIFC	01	Ordena enviar al bus GPIB el mensaje unilínea “*Interface Clear*” para limpiar la interfaz.
EREN	02	Ordena enviar el mensaje unilínea “*Remote Enable*” que permite el control remoto de un dispositivo.
ELLO	03	Ordena enviar el mensaje “**LLO**” para cerrar los controles locales de un dispositivo.
EDCL	04	Ordena enviar el mensaje “**DCL**” para colocar el dispositivo en su estado predefinido.
EGET	05	Ordena enviar el mensaje “**GET**” para disparar la programación de un dispositivo.
EUNL	06	Ordena enviar el mensaje “**UNL**” para deshabilitar todos lo posibles dispositivos con la capacidad para escuchar.
EUNT	07	Ordena enviar el mensaje “**UNT**” para deshabilitar todos lo posibles dispositivos con la capacidad para hablar.

Tabla 3.8. Mensajes unilínea y multilínea accesibles mediante el mensaje TEML.

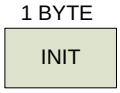
La estructura en que la tarjeta controladora recibe los mensajes es la siguiente:

1 BYTE	VARIOS BYTES
MENSAJE PCT	CADENA DE DATOS

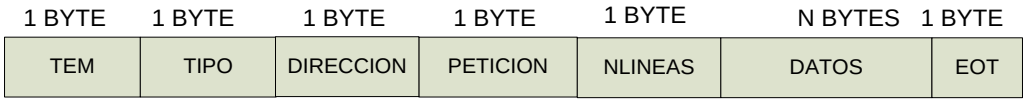
Donde MENSAJE PCT es cualquiera de los mensajes de la tabla 3.1, y CADENA DE DATOS es dependiente del tipo de mensaje PCT como se describe a continuación:

MENSAJE INIT: 01

Este mensaje no posee más datos ya que su función siempre será la misma.



MENSAJE TEM: 02



Donde:

TEM: 02

TIPO: 1, indica que el dato debe ser interpretado como un comando de Interfaz.
0, indica que los datos deben ser interpretados como comandos dependientes de dispositivo.

DIRECCION: Dirección GPIB del dispositivo, es un número entero entre 1 y 30.

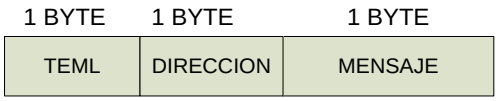
PETICION: Solo se usa cuando TIPO esta a 0.
1, indica que el comando dependiente de dispositivo es una petición.
0, indica que el comando dependiente de dispositivo es de control.

NLINEAS: Esta tiene significado solo cuando el campo petición es 1, este indica el número de líneas que se espera recibir del instrumento.

DATOS: Son los comandos de interfaz o dependientes de dispositivo específicos para el instrumento direccionado.

EOT: Carácter que indica el final del paquete.

MENSAJE TEML: 03



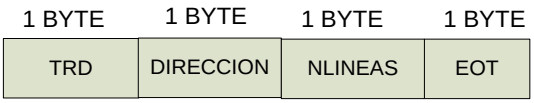
Donde:

TEML: 03

DIRECCION: Dirección GPIB del dispositivo, es un número entero entre 0 y 30.

MENSAJE: Código de los mensajes de interfaz o de línea y multilínea.

MENSAJE TRD: 04



Donde:

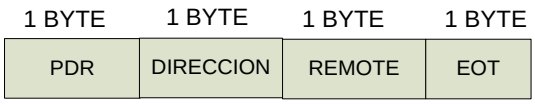
TRD: 04

DIRECCION: Dirección GPIB del dispositivo, es un número entero entre 0 y 30.

NLINEAS: Esta tiene significado solo cuando se requiere información del instrumento, este indica el número de líneas que se espera recibir del instrumento.

EOT: Carácter que indica el final del paquete.

MENSAJE PDR: 05



Donde:

PDR: 05

DIRECCION: Dirección GPIB del dispositivo, es un número entero entre 0 y 30.

REMOTE: Este puede ser *TRUE* (1) para direccionar el dispositivo en remoto, o puede ser *FALSE* (0) para ponerlo en local.

EOT: Carácter que indica el final del paquete.

3.3.2.2 Mensajes que envía la Tarjeta al servicio controlador GPIB

El servicio manejador de la tarjeta que se comunica con el software (programa del PIC16F877A) de la tarjeta también recibe mensajes provenientes de la tarjeta que indican el tipo de datos que la tarjeta le esta enviando como por ejemplo respuesta a peticiones de medición para los cuales se deben tomar decisiones de acciones a tomar. Estos mensajes se muestran en la Tabla 3.3.

MENSAJE	Código	OBJETIVO
DIRECCION	dd	Indica la dirección del dispositivo del cual provienen los datos.
TIPO _ DATO	dd	Indica el tipo de dato retornado según la ejecución de algún comando GPIB.
BYTE_ESTADO	dd	Tiene el último byte de estado obtenido de un censo serie.
FIN_PAQUETE	dd	Indica con 1 si el paquete actual es el ultimo Indica con 0 si el paquete actual no es el ultimo

Tabla 3.9. Mensajes enviados desde la tarjeta GPIB hacia el servidor.

La estructura en que la tarjeta envía estos mensajes junto con los datos es la siguiente:

1 BYTE	1 BYTE	1 BYTE	1 BYTE	N BYTE'S	1 BYTE
DIRECCION	TIPO_DATO	BYTE_ESTADO	FIN_PAQUETE	DATOS	<LF>

Donde:

- DIRECCION: Es el primer mensaje a interpretar de todos los datos y es un campo con longitud de 1 byte, este contiene la dirección GPIB del dispositivo que envía los datos.

- TIPO _ DATO: Este mensaje tiene longitud de 1 byte e indica la procedencia de los datos adjuntos, su clasificación se muestra a continuación.

  03: Indica que los datos del paquete son el resultado de una petición (*query*) de datos.

  04: Indica que los datos del paquete son el resultado de una petición (*query*) de un archivo.

  06: Indica que los datos siguientes son el resultado de una encuesta serie y que el campo BYTE_ESTADO es significativo (este es el único caso para el cual ese campo tiene significado).

- BYTE_ESTADO: Es el byte de estado del dispositivo que pidió atención, además este byte indica por que causa pide atención.

- DATOS: Campo con longitud de n bytes(n=92 bytes máximo), es la información que proporciona el instrumento ya sea por la causa de una petición.
- <LF>: es el carácter de fin de línea, utilizado para indicar el final de la transmisión o del paquete que envía la tarjeta GPIB.

3.3.2.3 Definición del software de la etapa de comunicación con la tarjeta GPIB

En la *tabla 3.4* se listan las distintas clases u objetos con sus respectivos métodos y atributos con los que cuenta el software manejador.

CLASE	DESCRIPCION	ATRIBUTOS	METODOS
ControladorGPIB	Clase única con método main que funciona como programa principal		• main: método principal del programa, ver figura 7.1
InstrumentosGPIB	Clase abstracta madre de todos los instrumentos a controlarse con el sistema GPIB, todos sus métodos y atributos son globales para todos los instrumentos que extiendan de esta clase.	• host: dirección IP asignada a la tarjeta controladora • puerto: numero de puerto al que responde la tarjeta controladora • datosIn: tiene los datos de entrada desde la tarjeta controladora • datosOut: tiene los datos de salida dirigidos hacia la tarjeta controladora	• enviarBytes: habilita al thread correspondiente para enviar datos a la tarjeta controladora • solicitudBytes: habilita al thread correspondiente para recibir datos desde la tarjeta controladora
ServidorGPIB	Objeto utilizado para atender las peticiones del cliente desde la pagina Web	• puerto: puerto por el cual escucha • datosCliente: datos que envía el cliente • InfClientLista: indica información lista para ser procesada	• escuchar: crea un thread implementando la interfase runnable que es quien atiende las conexiones de los clientes.
ComunicTarGPIB	Objeto que crea o establece un socket de comunicación TCP-IP con la tarjeta controladora GPIB Entradas host: dirección a la que responde la tarjeta controladora Puerto: puerto por el cual se comunican con la tarjeta GPIB	• socketTarjeta: socket TCP-IP de comunicación con la tarjeta GPIB • in: Stream de salida para el socket • out: Stream de entrada para el socket	
ThreadEnviar	Clase que extiende de InstrumentosGPIB e implementa Runnable su función es establecer comunicación con la tarjeta controladora y cuando enviarBytes lo permite envía los datos en forma de bytes contenidos en datosOut. Entrada UnSocket: objeto de tipo ComunicTarGPIB que ha establecido un socket con la tarjeta controladora		• run: método de ejecución del thread que espera por la habilitación para enviar los datos por el socket establecido con la tarjeta controladora
ThreadRecibir	Clase que extiende de InstrumentosGPIB e implementa Runnable su función es establecer comunicación con la tarjeta controladora y cuando recibirBytes lo indica recibe los datos en forma de bytes contenidos en datosInt.		• run: método de ejecución del thread que espera por la habilitación para recibir los datos por el socket establecido con la tarjeta controladora

Tabla 3.10. Lista de las clases creadas en Java para poder comunicar al usuario con la tarjeta controladora.

CLASE	DESCRIPCION	ATRIBUTOS	METODOS
TarjetaBD	Clase utilizada para manejar la persistencia (en la base de datos) de los parámetros de la tarjeta controladora y los demás instrumentos.		• DatosTarjeta: devuelve los parámetros de configuración de la tarjeta controladora • ActualizarCampo: crea una conexión con la base de datos tomando la declaración contenida en la variable "declaracion" la envía a MySql Entradas Declaracion: cadena de caracteres que representa la declaración SQL • BuscarRegBD: busca el valor de "registro" del Instrumento en el campo "Configuracion" en la Base de Datos en la tabla especificada Entradas: registro: contiene el valor del campo a buscar tabla: contiene el nombre de la tabla del instrumento

Tabla 3.10 (continuación). Lista de las clases creadas en Java para poder comunicar al usuario con la tarjeta controladora.

3.3.3 Etapa de control del osciloscopio Tektronix TDS640A

La tarjeta tiene manejadores para realizar su función de controlador, pero hay una porción de estos manejadores que son orientados a establecer comunicación con el servicio de comunicación con la tarjeta controladora, para esto hay una serie de mensajes que la tarjeta recibe desde el servicio y otros que la tarjeta debe enviar para el intercambio de información entre estos dos dispositivos que en conjunto realizan el papel de controlador.

3.3.3.1 Listado de objetos de la etapa de control del Osciloscopio Tektronix

En la *tabla 3.5* se listan las clases orientadas a controlar en forma remota al instrumento Tektronix TDS640.

CLASE	DESCRIPCION	ATRIBUTOS	METODOS
TektronixTDS640A	Objeto que extiende de la clase abstracta InstrumentosGPIB, permite el envío de parámetros de configuración para poder controlar en forma remota el osciloscopio TektronixTDS640A.	• acQuire: vector que contiene los mnemónicos que se pueden enviar al instrumento correspondientes al árbol de instrucciones • data: vector que contiene los mnemónicos que se pueden enviar al instrumento correspondientes al árbol de instrucciones • measurement: vector que contiene los mnemónicos que se pueden enviar al instrumento correspondientes al árbol de instrucciones	• inicializar: se encarga de obtener todos los parámetros del instrumento guardados en la base de datos. • acQuire: se encarga de negociar el envío de parámetros de configuración de adquisición de datos enviados por el cliente al instrumento Tektronix • data: se encarga de negociar el envío de parámetros de configuración del formato de datos del osciloscopio Tektronix • measurement: se encarga de negociar el envío de parámetros de configuración de medición de parámetros eléctricos de la señal o forma de onda adquirida por el osciloscopio.
threadTektronix	Este objeto extiende de la clase InstrumentosGPIB y además levanta un thread implementando la interfase Runnable, este thread recibe toda la información que le envía la tarjeta controladora.		• run: método de ejecución del thread que se encarga de recibir y decodificar los datos que le envía la tarjeta controladora para luego decidir si que hacer con los datos.

Tabla 3.11. Lista de las clases creadas en Java para poder comunicar al usuario con el osciloscopio Tektronix

3.3.4 Diagrama de flujo del programa principal

En la figura 3.9 se muestra el método principal “main” del programa del servicio controlador de la tarjeta controladora, este método pertenece a la clase definida como ControladorGPIB.java.

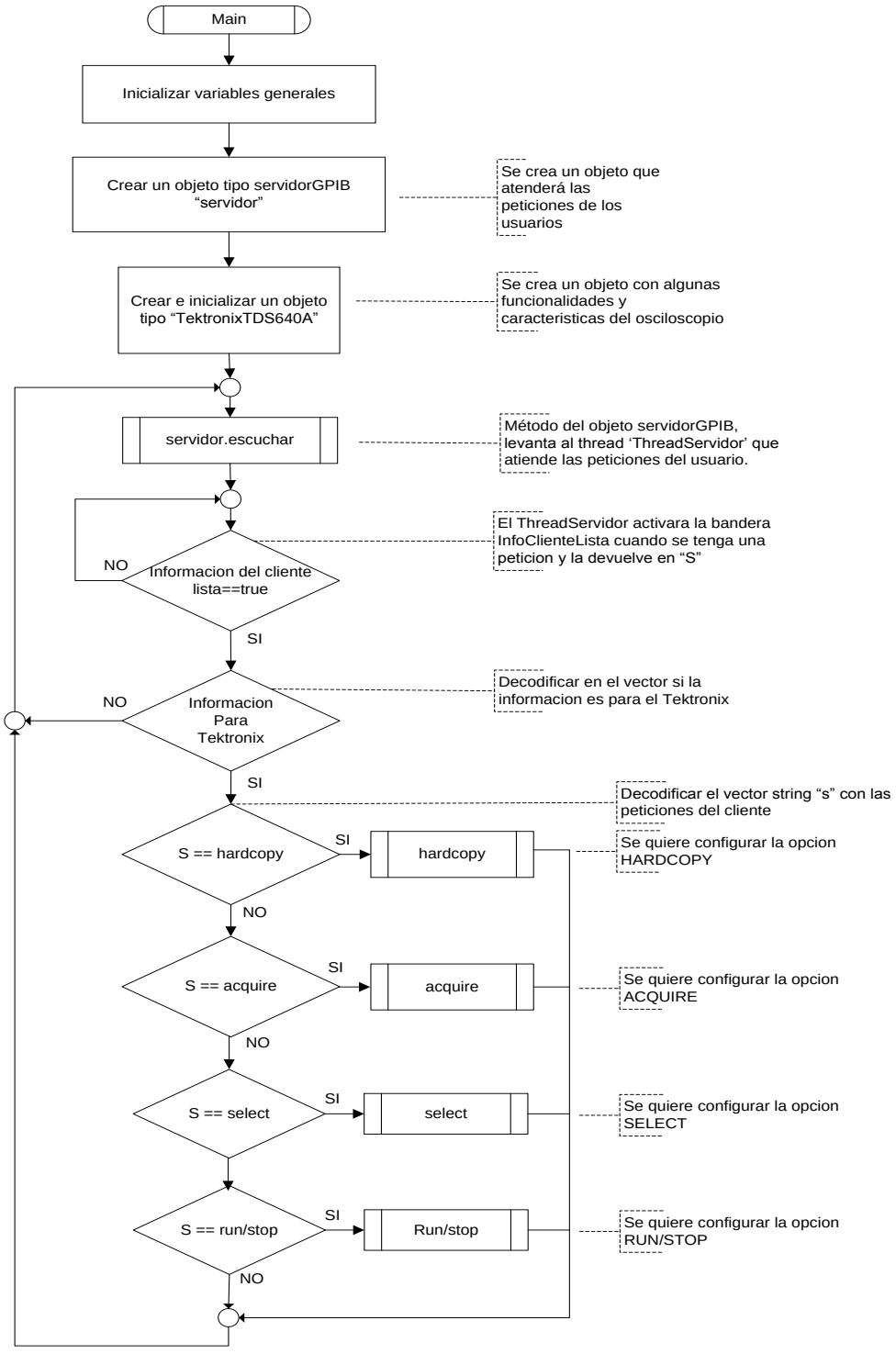


Figura 3.9: Diagrama de flujo del programa principal del servicio controlador de la tarjeta

El diagrama de flujo de la figura 3.9 es desglosado en las siguientes etapas:

1. Inicializar variables de propósito general: en esta primera se declaran e inicializan variables de carácter general tal como contadores y variables para guardar datos temporales.

2. Crear un objeto del tipo ServidorGPIB con nombre “Servidor”, el objeto de este es permitir la recepción de peticiones de un cliente y generar una respuesta en base a la misma petición.

3. Crear e inicializar un objeto tipo Tektronix TDS640AS con nombre “Osciloscopio” el cual hereda de una clase base abstracta madre llamada Instrumentos GPIB, este proceso se describe en la figura 3.10

4. Llamar al método “escuchar” del objeto tipo “servidorGPIB” para comenzar a esperar por peticiones del cliente por medio del inicio de un hilo (thread).

5. Cuando una petición esta lista entonces el hilo (thread) que inicio con el método “escuchar” activa la bandera “InfoClientLista” y es cuando se procede a descifrar el contenido de este de lo contrario se sigue esperando por dicha información.

6. Se revisa cual es el instrumento que el cliente desea controlar, si es el Osciloscopio Tektronix se procede a seguir analizando los datos, de lo contrario se espera por mas peticiones.

7. Si los datos o ordenes son para el Osciloscopio entonces se revisa que control se quiere acceder en forma remota, entre los que se revisan están: Acquire, Hardcopy, Select.

3.3.4.1 Inicialización del servicio manejador de la tarjeta GPIB

La inicialización se refiere a la carga de configuración de la tarjeta GPIB como la de los instrumentos conectados al bus. La carga de la configuración de la tarjeta se realiza cuando creamos por primera vez un objeto tipo “InstrumentoGPIB” ver *tabla 3.4*, esto se realiza según los siguientes pasos y se describe en la *figura 3.10:*

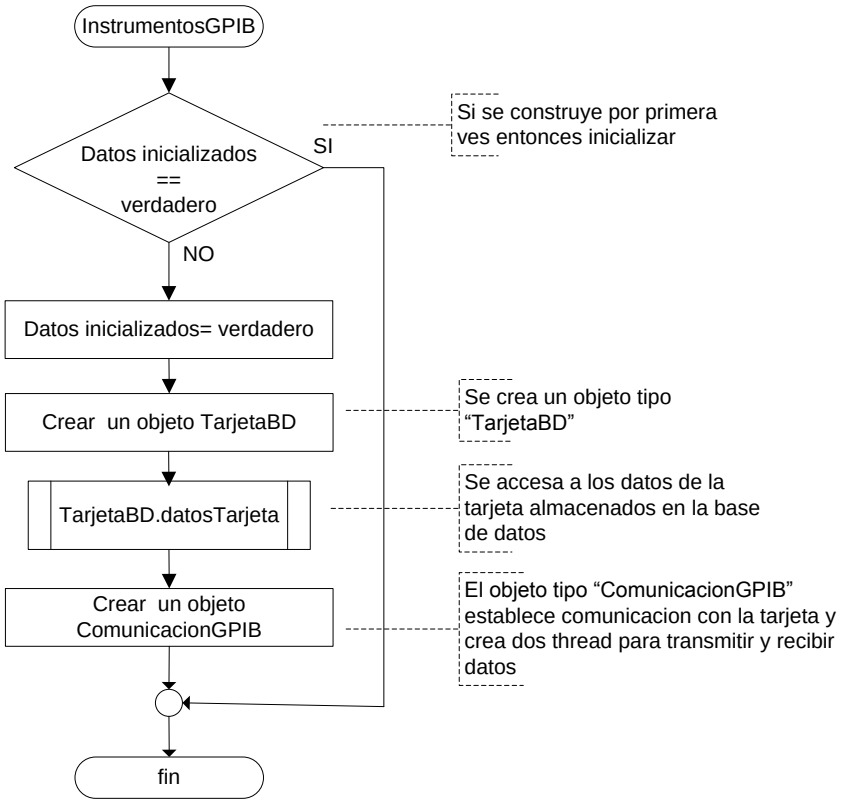


Figura 3.10. Método constructor de objetos que extienden de "InstrumentosGPIB"

- Se revisa si la configuración de la tarjeta controladora, ha sido leída desde la base de datos sino se finaliza la ejecución del método.

- Si es primera vez que se construirá un objeto que hereda de "Instrumento" entonces:

- Se indica datos inicializados.

- Se crea un objeto de tipo "TarjetaBD" y se accede a su método "Datos tarjeta" que obtiene los parámetros de configuración de la tarjeta controladora como la dirección IP y el numero de puerto TCP.

- Se crea un objeto tipo "ComunicacionGPIB" que se encarga de establecer un socket TCP-IP con la tarjeta y además se levantan dos thread que utilizan el mismo socket de comunicación uno para transmitir y otro para recibir datos de la tarjeta.

3.3.4.2 Inicialización del software controlador del Tektronix TDS640A

La inicialización del manejador del Osciloscopio Tektronix se muestra en el diagrama de flujo de la figura 3.11, donde se ejecuta el método constructor respectivo que se describe a continuación y es el llamado el método “inicializar” que carga la configuración del medidor y los respectivos parámetros para su adecuado control en forma remota.

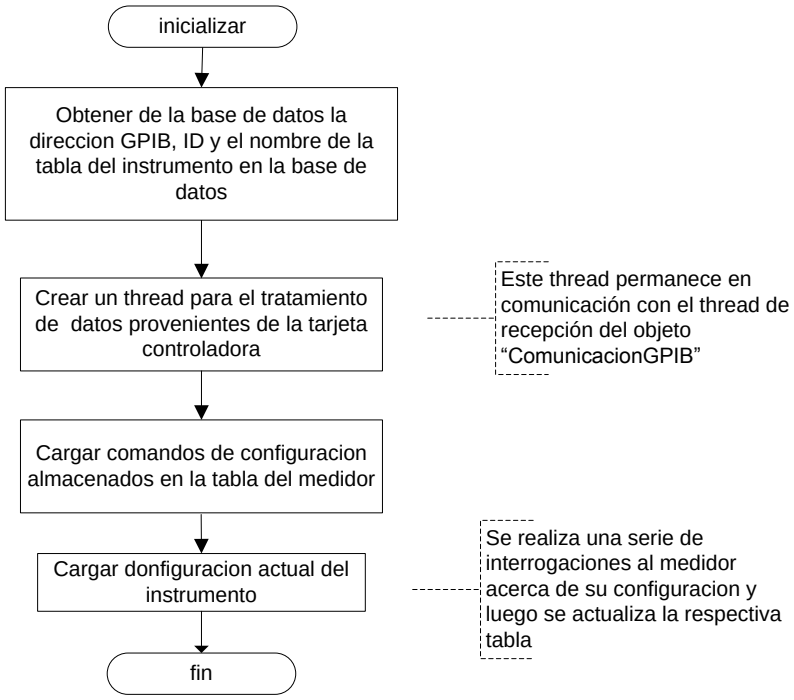


Figura 3.11 Diagrama de flujo del método “inicializar” del objeto “TektronixTDS640A”

Descripción del método Inicializar del objeto Tektronix :

- Establecer una conexión con la base de datos de la tarjeta controladora para obtener datos de configuración del instrumento Tektronix tales como dirección GPIB, identificador del instrumento (ID) y el nombre de la correspondiente tabla en la base de datos de la tarjeta controladora.

- Crea un thread para el instrumento con el fin de recibir los datos destinados para el y poder decidir el tratamiento de estos datos que pueden ser respuestas a peticiones del instrumento o petición de servicio por parte del mismo, este thread realmente no se comunica directamente con la tarjeta sino espera a que el thread de recepción indique que hay datos provenientes de la tarjeta GPIB.

- Se cargan de la tabla respectiva en la base de datos los conjuntos de comandos que pueden ser enviados al medidor.

- Se interroga al medidor por su configuración actual para actualizar su estado en sus correspondientes campos de la tabla en la base de datos.

3.4 Servicio de Aplicación Web para cliente

En esta sección se describe la aplicación Web mediante la cual un usuario puede acceder remotamente a la configuración de la tarjeta controladora del bus GPIB y al control del osciloscopio Tektronix TDS640A.

Esta aplicación esta programada utilizando la tecnología *Java Server Pages* (*JSP*) para el desarrollo de aplicaciones Web dinámicas, también se utiliza código HTML y *javasript* que aporta funcionalidades dinámicas a las páginas web. A continuación se describe brevemente cada uno de los elementos antes mencionados y finalmente se mostraran las figuras de las páginas que conforman la aplicación web.
Para mas detalles de estas tecnologías observe la documentación del disco compacto.

3.4.1 Java Server Pages

Tecnología desarrollada por *Sun Microsystem*, su funcionamiento se basa en *scripts*, que utilizan una variante del lenguaje *java*; esto permite generar contenido web dinámico, esto puede ser en forma de documentos HTML o XML. Las *JSP* permiten al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento web.
En las *JSP*, se describe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente. El código java se incluye dentro de las marcas de etiqueta <% y %>, a esto se le denomina *scriptlet*.

La principal ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles la aplicación web, almacenando en clases java las partes que consumen mas recursos así como las que requieren mas seguridad, y dejando la parte encargada de formatear el documento html en el archivo jsp.

3.4.2 HTML y javascript

El HTML, acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a internet y a los navegadores del tipo internet Explorer, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos y también de los más fáciles de aprender.

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, orientado a las páginas web, con una sintaxis semejante a la del lenguaje java y el lenguaje C.
Al contrario que Java, *JavaScript* no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien orientado a eventos. Estará listo para actuar en cuanto un evento (un click en un botón, por ejemplo) sea ejecutado. Aún así *javascript* implementa una sencilla interfaz de objetos/propiedades/métodos.
El lenguaje *Javascript* se integra dentro del código HTML de las páginas Web.
El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que fabricó los primeros navegadores de internet comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.
Tradicionalmente, se venía utilizando en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. *JavaScript* se ejecuta en el navegador al mismo tiempo que las sentencias van descargandose junto con el código HTML.

3.4.3 Descripción de la aplicación web

Anteriormente describimos la tecnología utilizada para el desarrollo de la aplicación web, ahora observaremos las páginas que componen la aplicación mediante algunas figuras.

La Figura 3.12 muestra la pagina principal que se muestra al acceder al sitio web mediante la url digitada en un navegador que para nuestro caso es:
http://localhost:8080/examples/jsp/controlDeInstrumentosGPIB.html.

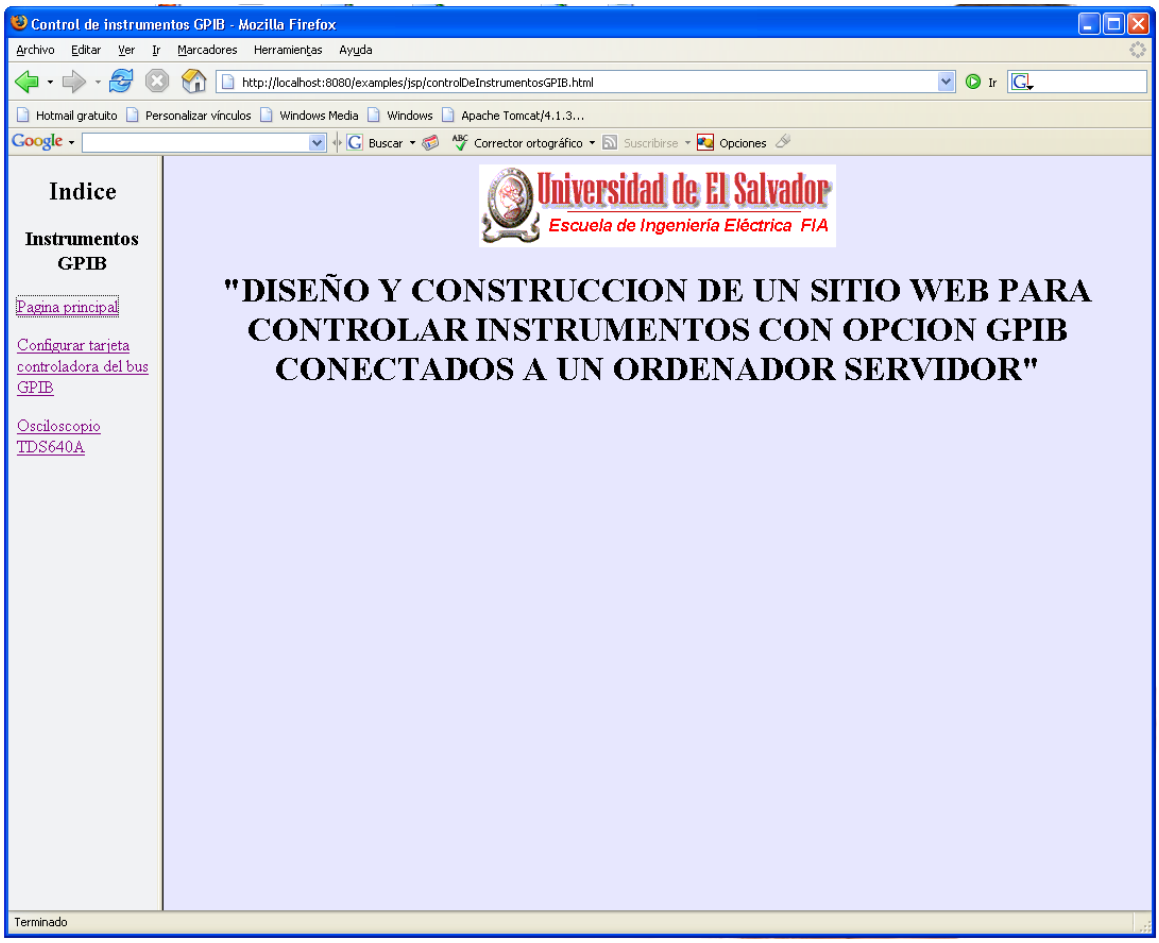


Figura 3.12 Pagina principal del sitio web para el control de instrumentos GPIB.

En la figura 3.12 se observa una sencilla página web compuesta por dos marcos, el izquierdo muestra un índice que permite acceder mediante links a: la pagina principal (la misma de la figura 3.12) y otras dos paginas *jsp*, una de las cuales es utilizada para configurar la tarjeta controladora y la otra para acceder remotamente al osciloscopio TDS640A.

El código de esta parte del sitio web puede ser observado en los archivos *controlDeInstrumentosGPIB.html, indice.html y paginaPrincipal.jsp;* estos se encuentran en el disco compacto que acompaña a este documento.

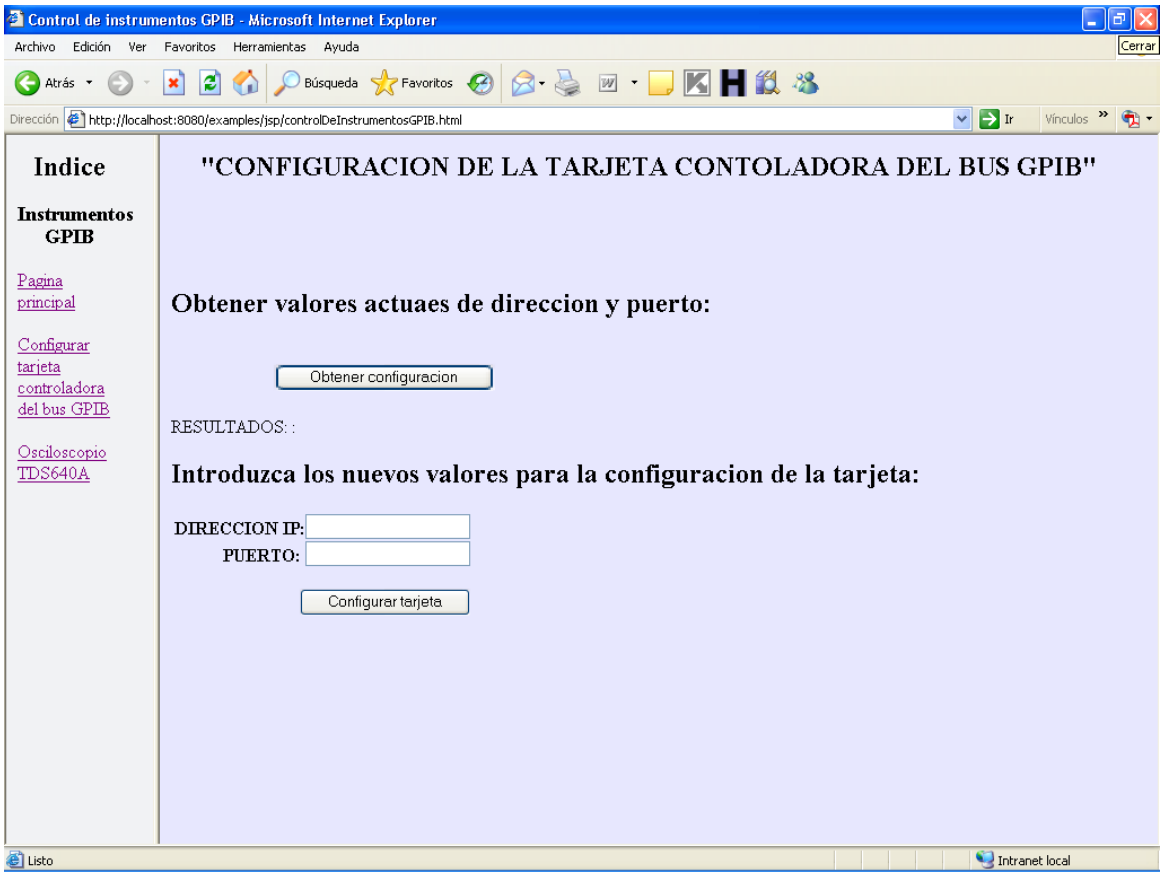


Figura 3.13 Pagina de configuración de tarjeta controladora del bus GPIB.

La figura 3.13 muestra la pagina jsp mostrada después de un click en el link del índice "*Configurar tarjeta Controladora del bus GPIB*", en esta se puede obtener la configuración actual de la tarjeta controladora (dirección IP y número de puerto) o también se puede asignar nuevos valores para dirección IP o puerto de la tarjeta. El código fuente de esta pagina jsp puede ser observado en el archivo configurarTarteja.jsp en el disco compacto que acompaña a este documento.

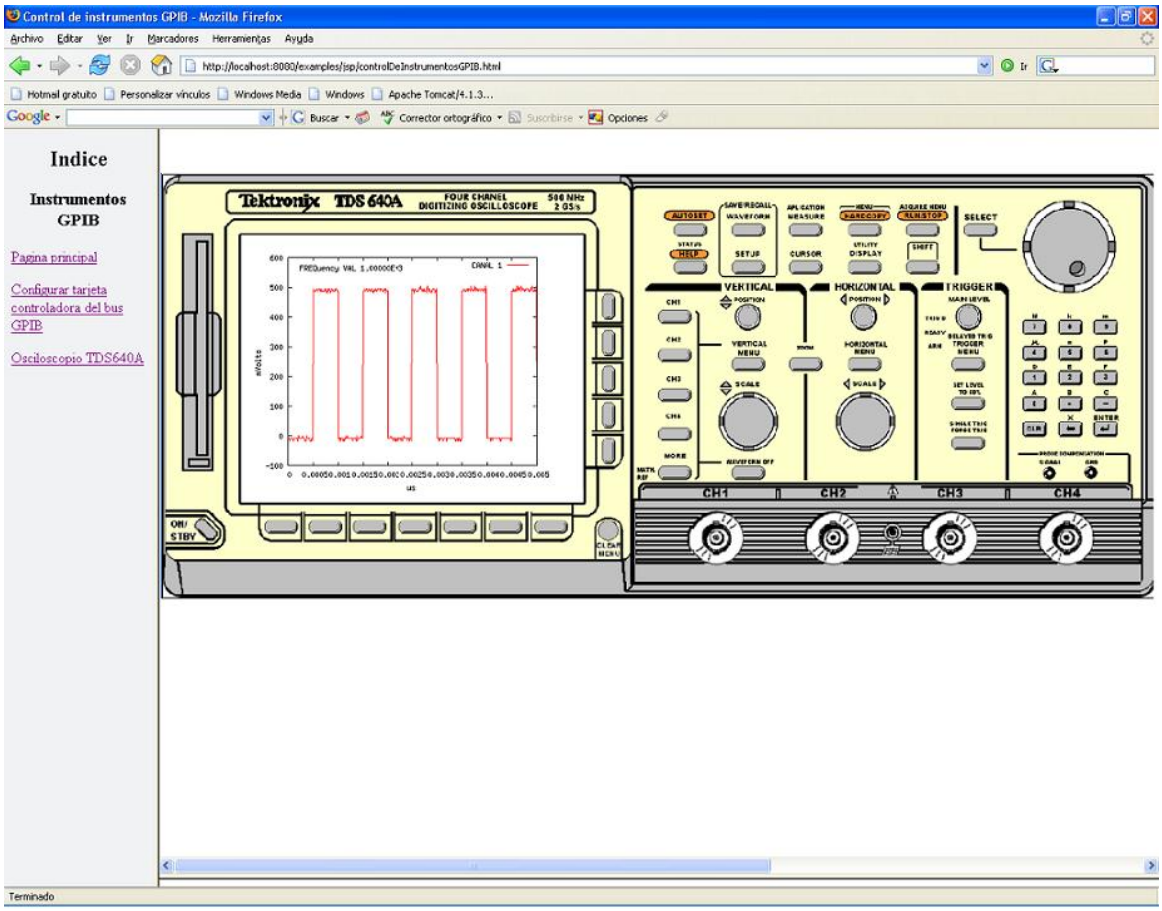


Figura 3.14 Pagina para el control remoto del osciloscopio Tektronix TDS640A.

La figura 3.14 muestra el panel frontal del osciloscopio TDS640A, este puede ser controlado remotamente desde esta página, el código puede ser observado en los archivos *osciloscopioFrame.html, osciloscopioTDS640AImage.html, osciloscopioTDS640AForm.jsp, OsciloscopioBean.java y scriptOsciloscopio.js* que pueden ser encontrados en el CD que acompaña a este documento.

Es en esta parte del sitio web en que se utiliza mayormente la tecnología jsp mediante la cual se procesa la información que el usuario envía en los formularios (estos son llenados con comandos según las opciones elegidas de los menús emergentes mostrados en el panel frontal del osciloscopio o mediante un click en algunos botones).
Es preciso hacer notar que cuando la página de la figura 3.14 es observada por el usuario en un navegador, solo en los botones que se observa un cambio de color son los que tienen acción en el control del osciloscopio (a excepción de los botones del teclado que también pueden ser utilizados y no muestran cambio de color).

En el código de javascript del archivo *scriptOsciloscopio.js* podremos encontrar múltiples funciones que hacen que el usuario observe un comportamiento similar al del panel frontal del osciloscopio, esto se hace mostrando un conjunto de menús emergentes como los que el osciloscopio mostraría si se estuviera controlando localmente, según las opciones elegidas de estos menús se enviara un formulario al servidor que contiene los comandos e la configuración que el usuario a elegido, luego estos comandos son enviados (ver código *OsciloscopioBean.java*) hacia la aplicación java que a través del puerto 1024 y posteriormente procesados y enviados hacia el instrumento por medio de la tarjeta controladora.

En esta sección se describe la página Web que se muestra al usuario para controlar el Osciloscopio Tektronix de forma remota, esta contiene algunos controles básicos

3.5 Diseño del servicio de almacenamiento en base de datos MySQL

Para la creación de este servicio se utilizo MySQL y la librería JDBC de java con el objeto de iniciar un proceso por el cual se envían declaraciones SQL.

3.5.1 Diseño de la Base de datos de la tarjeta controladora

Para poder almacenar la información de la tarjeta controladora y del Osciloscopio y que esta persista en todo momento se ha creado una base de datos en MySql llamada “BaseDatosTarjetaGPIB”. En la *figura 3.12* se muestra la estructura de las tablas que componen la base de datos de la tarjeta controladora.

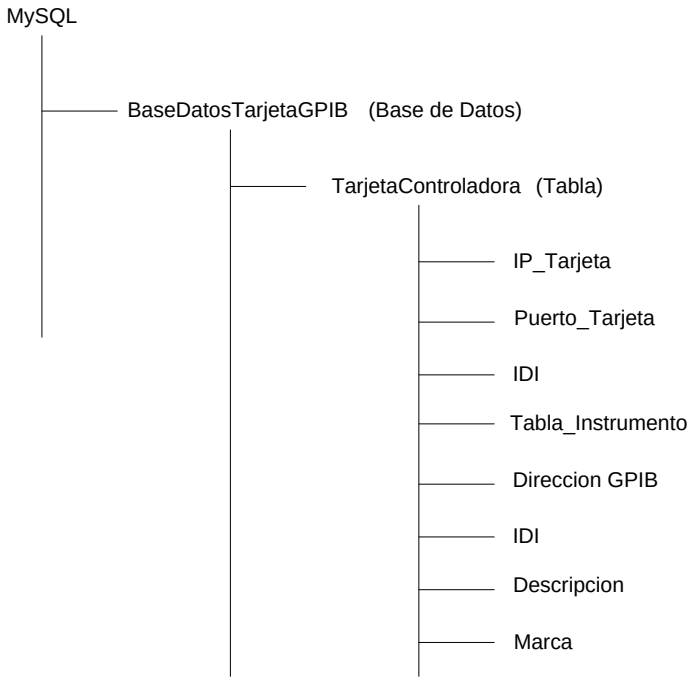


Figura 3.15: arquitectura del diseño de base de datos para la solucion

A continuación en la tabla 3.6 se muestra el contenido para la tabla TarjetaControladora.

IP_Tarjeta	Puerto_Tar	IDI	Tabla_Instrumento	DireccionGPIB	Descripcion	marca
192.168.111.12	80	Tektronix	TektronixTDS640A	1	Osciloscopio	Tektronix

Tabla 3.12. Tabla TarjetaControladora de la base de datos

El contenido de la tabla TektronixTDS640A del osciloscopio contiene los estados de las configuraciones de las distintas funciones del medidor.

CONCLUSIONES DEL CAPITULO III

- ✓ La tarjeta controladora esta compuesta por dos módulos, uno para establecer la comunicación TCP/IP con el servicio controlador y el otro para ejercer la función de controlador del bus GPIB.

- ✓ El microcontrolador PIC16F877A es el principal elemento de la tarjeta controladora y el software que se ejecuta dentro de este contiene las funciones para controlar el bus GPIB.

- ✓ Para la comunicación entre la tarjeta controladora y el servicio de control de la tarjeta se ha implementado un protocolo de comunicación propio

- ✓ El modulo GPIB esta formado esencialmente por componentes de *hardware* y *software* que combinados permiten el control del bus y de los instrumentos conectados a este.

- ✓ El manejador del modulo GPIB consta de múltiples rutinas que son las que proporcionan todas las funcionalidades básicas definidas por el estándar IEEE488.1 y permiten el control de dispositivos conectados al bus.

- ✓ El manejador que se ejecuta dentro del microcontrolador para controlar del bus GPIB necesita de cierto requerimiento de hardware para el acople de las líneas al bus GPIB y asi cumplir con la funcion de controlador del bus GPIB, estos circuitos son los *transceivers (74LS642, 74LS645)* y un match (74LS75) para la estabilidad de la línea ATN y dos inversores (7404) para complementar el control de los *transceivers*.

- ✓ El modulo TCP/IP esta compuesto fundamentalmente por la tarjeta EDTP que contiene el controlador ethernet RTL8019AS el cual implementa la capa física del modelo OSI delegando al microcontrolador la tarea de decodificar las cabeceras y datos de los segmentos TCP/IP.

- ✓ Las rutinas que manejan la comunicación ethernet forman parte del f*irmware* que esta implementado dentro del microcontrolador PIC.

- ✓ La función de los chips *transceivers* SN74LS645-1N es el multiplexar el puerto D (utilizado por los módulos TCP/IP y GPIB), logrando de este

modo que el puerto D maneje (uno a la ves) las señales DIO del bus GPIB y el bus de datos del RTL8019AS.

- ✓ Se mostró a través de diagramas de flujo la lógica del programa principal de la tarjeta controladora y se detallo sobre el proceso de ejecución del programa que se ejecuta en el PIC16F877A

- ✓ El MAX232 fue utilizado en este diseño para ayudar al desarrollo y depuración de los programas de la tarjeta controladora.

- ✓ En el desarrollo del servicio controlador de la tarjeta GPIB se utiliza tecnología Java para crear un software que permita manejar la tarjeta controladora y el osciloscopio Tektronix TDS640A.

- ✓ El código del software del servicio de control de la tarjeta GPIB se divide en dos áreas, una orientada a establecer comunicación con la tarjeta controladora y otra para manejar los controles y configuración del Tektronix en forma remota.

- ✓ La estructura de las tablas que componen la base de datos de la solución permite almacenar en forma persistente la configuración de la tarjeta.

- ✓ El uso de Java Script, Java Server Pages y código HTML en la aplicación Web ayudan a crear una interfase para el osciloscopio Tektronix amigable y fácil de usar.

REFERENCIAS BIBLIOGRAFICAS

R. ANSI/IEEE Std 488.1-1987. IEEE Standard Digital Interface for Programmable Instrumentation (The Institute of Electrical and Electronics Engineers, Inc. 345 East 47th Street, New Cork, NY 10017,USA).

R. Microchip. PICmicro Mid-Range MCU Family Reference Manual (1997 Microchip Technology Inc. USA.).

R. RTL 8019AS. Realtek Full – Duplex Ethernet Controller with Plug and Play Function (Real PNP). (REALTEK SEMICONDUCTOR CORP. HEAD OFFICE NO. 2, INDUSTRY E. RD. IX, SCIENCE – BASED INDUSTRIAL PARK, HSINCHU 300, TAIWAN, R.O.C. TEL: 886-3-5780211 FAX: 886-3-5776047).

R. RFC: 791. INTERNET PROTOCOL (Protocolo Internet) DARPA INTERNET PROGRAM ESPECIFICACION del PROTOCOLO. Septiembre 1981 (Traducción al castellano: mato 1999 Por: Pedro J. Ponce de León <pjleon@arrakis.es>)

R. RFC: 793. PROTOCOLO DE CONTROL DE TRASMISIÓN DARPA INTERNET PROGRAM ESPECIFICACION DE PORTOCOLOS. Septiembre de 1981 (Traducido al español por Pedro J. Ponce de León Pierre@mail.ono.es
Domingo Sánchez Ruiz domingo@toboso.fis.ucm.es
y José Ignacio Usera Estirado j_usera@lycos.es)

CAPITULO IV

COSTOS DE CONSTRUCCION DE LA TARJETA CONTROLADORA

Introducción

En este capitulo se desglosan los costos que se realizaron para construir la tarjeta controladora, los costos que se mencionan son solo de materiales no incluyen mano de obra ni valores agregados. Los costos se presentan en una lista en la que se detallan los elementos de la tarjeta controladora (ver figura 3.3) con una descripción técnica y su respectivo costo. Al final del capitulo se tiene un precio o costo de materiales de la tarjeta controladora GPIB.

4.1 Costos de materiales de la tarjeta controladora

En la siguiente tabla se listan los costos de construcción de la tarjeta controladora

ITEM	ELEMENTO	DESCRIPCION	CANTIDAD	COSTO U.$	COSTO T.$
1	PIC16F877A	microcontrolador de Microchip	1	18	18
2	RTL8019AS	microcontrolador ethernet	1	50	50
3	74645N		3	3	9
4	SN74LS642-1N		1	3	3
5	74LS75N	Latch	1	3	3
6	7404N	inversor	2	3	6
7	LM7805	Regulador de voltaje	1	1.03	1.03
8	XTAL	cristal de 20MHz	1	1.25	1.25
9	Puerto DB9	conector puerto serie hembra 9 pines	1	2	2
10	C 27pF	capacitores de 27 pF	2	0.25	0.5
11	C 1uF	capacitorres de 1uF	1	0.27	0.27
12	R 4.7KΩ	Resistencia de 4.7KΩ	1	0.15	0.15
13	R 300Ω	Resistencia de 300Ω	8	0.15	1.2
14	conector tipo Ribon	conector puerto GPIB hembra 25 pines	1	7	7
15	Tableta de cobre	Tableta de cobre de 18x22 cm	1	5.15	5.15
				Total	107.55

Tabla 4.1: costos de construcción de la tarjeta controladora

En la tabla 4.1 se obtiene un valor de 107.55 $ como costo total de materiales para construir la tarjeta controladora. Estos valores ya incluyen IVA.

CONCLUSIONES DEL CAPITULO IV

- ✓ El costo de construcción de la tarjeta controladora es menor al que se podría encontrar por una tarjeta con características equivalentes.

- ✓ El costo obtenido en esta sección no representa el costo total de la solución presentada en este documento.

CONCLUSIONES GENERALES Y RECOMENDACIONES

CONCLUSIONES

1. Con este trabajo se obtiene el control de instrumentos conectados al bus IEEE488 a través de un sitio WEB gestionado por un ordenador servidor, el desarrollo de esta solución esta dividida en tres partes (ver Figura 1.1); la tarjeta controladora (como centro del sistema) controla a los instrumentos conectados al bus IEEE488 y mantiene comunicación con el servidor el cual envía las ordenes que la tarjeta debe ejecutar.

2. La aplicación de usuario es programada usando J2EE. Esta mostrara una página WEB creada usando JSP la cual es despachada por un servidor en el cual se ejecuta la aplicación para el control de la tarjeta GPIB.

3. La tarjeta controladora tiene como componente central el PIC16F877A el cual maneja las funcionalidades del bus IEEE488 y mantiene la comunicación con la aplicación en el servidor asistido por el controlador RTL8019AS (que forma parte de la tarjeta EDTP) implementando se esta manera el protocolo de comunicación TCP/IP.

4. El estándar IEEE488 surge en el año de 1975 y después de esto a sido mejorado, convirtiéndose en un estándar de comunicación para instrumentos ampliamente utilizado.

5. La primera versión del estándar fue definida como IEEE488.1 y establece las características eléctricas, mecánicas y funcionalidades básicas para el control de instrumentos.

6. Luego de su primer versión, el estándar evoluciona y se crea el IEEE488.2, el cual define más precisamente la comunicación entre dispositivos controladores con los instrumentos; además el SCPI (Standar Commands for Programble Instruments) es construido tomando como base el IEEE488.2 y proporcionando un conjunto de comandos de programación comprensivos los cuales son utilizados por un gran numero de instrumentos de diferentes fabricantes.

7. Respecto a la tarjeta controladora, esta es compuesta por dos módulos, uno para mantener la comunicación con el servidor (usando TCP/IP) y otro para comunicarse con los dispositivos conectados al bus IEEE488; ambos módulos están compuestos a su vez por *software* y *hardware* asociados a ellos y encargados de complementar su funcionamiento.

8. Los chips *transceivers* SN74LS645-1N son utilizados para el multiplexado del puerto D (utilizado por los módulos TCP/IP y GPIB), logrando de este modo que el puerto D contenga las señales DIO del bus GPIB y el bus de datos del RTL8019AS; pero solo uno de los dos módulos puede utilizar el puerto D a la vez.

9. En el desarrollo del controlador servidor de la tarjeta GPIB se utiliza tecnología Java para manejar la tarjeta controladora GPIB y el Osciloscopio Tektronix TDS640A.

10. El código se divide en dos áreas una orientada a establecer comunicación entre la tarjeta controladora y el cliente; y otra orientada a manejar la información y eventos del osciloscopio Tektronix TDS640A.

RECOMENDACIONES

1. La comunicación RS232 fue utilizada en este diseño para ayudar al desarrollo y depuración de los programas de prueba de la tarjeta y por esto se ha incluido como parte del circuito final con el objeto de seguir aportando ayuda en una mejora futura.

2. Las conexiones para adicionar una memoria I2C 24LC1025 también forman parte del circuito final previniendo una posible necesidad de mas memoria para registros o datos en un posterior diseño.

ANEXOS

A 1. CODIGO DEL MANEJADOR DE LA TARJETA CONTROLADORA

El código fuente de estos programas también se encuentra en el disco compacto de este trabajo.

A 1.1 CODIGO DEL PROGRAMA PRINCIPAL

```
;*************************************************************

         include   "P16F877A.INC"
         include "const_eep.inc"
FIN_DE_CADENA               equ       '$'
physical_address0   equ 0x21  ;variables para el controlador del RTL
VARIABLES_RTL2              equ 0xA0  ;variables para el controlador del RTL
variables_GPIB              equ 0xB8
BUFFER_ENTRADA              equ 0x90  ;buffer de entrada paquetes TCP 16 bytes

BUFFER_SALIDA               equ 0x10  ;buffer de salida paquetes TCP a
                                                                        ;partir de dirección 0x110 (bank 2)
                                                                        ;IRP=1 direccionamiento imdirecto

         org                0x00
         goto      inicio            ;saltar al inicio del programa principal

         org                0x04
         interrupciones
;************************
         ;salvar el contexto del programa principal
         movwf     W_tmp                     ;salvamos el registro W
         swapf     STATUS,W          ;y el registro STATUS "girado" en W
         bsf                STATUS,RP0                ;banco 1
         movwf     STATUS_tmp                ;guardamos el registro STATUS
         movf      PCLATH,W
         movwf     PCLATH_tmp                ;guardamos el registro       PCLATCH
;        movf      FSR,W
;        movwf     FSR_tmp
;        movf      INDF,W
;        movwf     INDF_tmp
         bcf                STATUS,RP0                ;banco 0
         movf      TRISTATE,W
         bsf                STATUS,RP0                ;banco 1
         movwf     TRISTATE_tmp
         movf      DATOS,W                   ;guardar direccion de PORTD
         movwf     DATOS_dir
         bcf                STATUS,RP0                ;banco 0
         movf      DATOS,W
         bsf                STATUS,RP0                ;banco 1
         movwf     DATOS_tmp         ;guardar datos de PORTD
         bcf                STATUS,RP0                ;banco 0
         ;************************


probarINT_TMR0
         btfss     INTCON,T0IF
         goto      probarINTE

         bcf                STATUS,RP0
         bcf                INTCON,T0IF
         movlw     0xFF                      ;Le asigno FF al TMR0
         movwf     TMR0                      ;Al recibir un flanco de subida se activara la interrupción


         bcf                STATUS,RP0
         bcf                PCLATH,4
         bsf                PCLATH,3          ;PAGINA 1 DE PROGRAMA
         call      BUS_DATOS_RTL       ;asignar el bus de datos al modulo TCP-IP
         bcf                PCLATH,4
         bcf                PCLATH,3          ;PAGINA 0 DE PROGRAMA
         call      censar_nic
         goto      salirINT
probarINTE
         btfss     INTCON,INTF
```

```
          goto      salirINT            ;atender la interrupcion del PIC

          bcf                 INTCON,INTF                   ;limpiar bandera
;         CALL      BANDERA3
          ;**********asignamos un valor de NLINEAS
          movlw     0x01
          bsf                 STATUS,RP0                    ;banco 1
          movwf     NLINEAS
          bcf                 STATUS,RP0                    ;banco 0
          ;************************************
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      BUS_DATOS_GPIB
          call      responder_SRQ
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

salirINT

          ;************************
          ;recuperando el contexto del programa principal
          bsf                 STATUS,RP0                    ;banco 1
          movf      DATOS_dir,W                   ;recupero dieccion de PORTD
          movwf     DATOS
          movf      DATOS_dir,W
          bcf                 STATUS,RP0                    ;banco 0
          movwf     DATOS                         ;recupero los datos de PORTD
          bsf                 STATUS,RP0                    ;banco 1
          movf      TRISTATE_tmp,W
          bcf                 STATUS,RP0                    ;banco 0
          movwf     TRISTATE
          bsf                 STATUS,RP0                    ;banco 1
;         movf      FSR_tmp,W
;         movwf     FSR
;         movf      INDF_tmp,W
;         movwf     INDF
          movf      PCLATH_tmp,W
          movwf     PCLATH
          swapf     STATUS_tmp,W
          movwf     STATUS
          swapf     W_tmp,F
          swapf     W_tmp,W
          ;************************
          retfie



          org                 0x44

inicio

          bcf                 STATUS,RP0                    ;Selecciónа el banco 0
          ;INICIALIZACION DEL MODULO GPIB
          ;configurar al PIC para usar el MODULO GPIB
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      configurar_microcontrolador
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA


RESET_TARJETA
          bsf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA
          call ini_MODGPIB
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          ;Carga la dirección IP local desde la memoria EEPROM DEL PIC
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      cargar_IP           ;carga la IP desde la EEPROM
          call      cargar_PUERTO       ;carga el PUERTO desde la EEPROM
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      BUS_DATOS_RTL       ;asiganar el bus de datos al RTL
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          call      init_nic
          movlw     TCR
          movwf     REGISTRO
          movlw     0X00
          movwf     VALOR
          call      outport


          movlw     LISTEN
          bsf                 STATUS,RP0                    ;banco 1
          movwf     html_socket
          movwf     html_socket2
          ;indicar que no hay paquete completo
```

```
          bsf                 STATUS,RP0                    ;BANK1
          clrf      FIN_LINEA           ;indicamos por omision no fin de linea
          bcf                 STATUS,RP0                    ;BANK0

          ;inicializar buffer de entrada de paquetes TCP
          bsf                 STATUS,IRP                    ;banco 2,3(direccionamiento indirecto)
          movlw     BUFFER_ENTRADA
          movwf     FSR

;-------  configuracion de las interrupciones    -----------

          ;CONFIGURACION DE INTERRUPCION DEL GPIB
          bsf                 INTCON,INTE                   ;habilita la interrupcion externa INT0
          bsf       STATUS,RP0
          bcf                 OPTION_REG,INTEDG


          ;CONFIGURACION DE INTERRUPCION DEL MODULO TCP/IP
          bcf                 STATUS,RP0
          movlw     0xFF                        ;Le asigno FF al TMR0
          movwf     TMR0
          bsf       STATUS,RP0
          bsf                 OPTION_REG,T0CS
          bcf                 OPTION_REG,T0SE
          bsf                 OPTION_REG,PSA                ;preescalador asignado a WDT??
                    ;Al recibir un flanco de subida se activara la interrupción
          bcf                 STATUS,RP0
          bsf                 INTCON,T0IE
          ;HABILITAMOS INTERRUPCIONES
          bsf                 INTCON,GIE


;-----------------------------------------------------------

LAZO_PRINCIPAL

          ;revisar si en el BUFFER_ENTRADA hay paquete completo
          bsf                 STATUS,RP0                    ;BANK1
          btfss     FIN_LINEA,0
          goto      LAZO_PRINCIPAL
          bcf                 INTCON,T0IE                   ;deshabilitar interrupciones del timer 0
          bcf                 INTCON,GIE
;         CALL      BANDERA1
          bcf                 STATUS,RP0                    ;BANK0
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
;         call      BUS_DATOS_RTL       ;asignar el bus de datos al modulo TCP-IP
;         call      tratamiento_datos_TCP
          call      BUS_DATOS_GPIB
          call      analizar_PCmensajes
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          ;indicar buffer de entrada listo para otro paquete de datos
          bsf                 STATUS,RP0                    ;BANK1
          bcf                 FIN_LINEA,0
          bcf                 STATUS,RP0                    ;BANK0
;         CALL      BANDERA2
          bsf                 INTCON,T0IE                   ;habilitar interrupciones del timer 0
          bsf                 INTCON,GIE
          goto      LAZO_PRINCIPAL
;*****************************************************************




retardo_ms:
;17*99*0.2us*33=1000us=1ms
lazo_rms2
          bcf                 STATUS,RP0                    ;banco 0
          movlw     0x11                        ;17 decimal
          movwf     CONTADOR2
lazo_rms1
          movlw     0x63                        ;99 decimal
          movwf     CONTADOR1
lazo_rms0
          decfsz    CONTADOR1,F                 ;un ciclo
          goto                lazo_rms0
          decfsz    CONTADOR2,F
          goto                lazo_rms1
          bsf                 STATUS,RP0                    ;banco 1
          decfsz    m_segundos,F
          goto                lazo_rms2
          bcf                 STATUS,RP0                    ;banco 0
          return

;*****    RUTINA PARA INICIALIZAR EL MODULO GPIB


          include   "RTL8019AS.INC"
          include   "GPIB.INC"
```

```
          include   "controladorRTL.INC"
          include   "DRIVER_GPIB.INC"
          END
```

A 1.2 CODIGO DEL MANEJADOR DEL MODULO GPIB

```
;*****************************************************************
;*****************************************************************

;DEFINICION DE LAS RUTINAS CONTROLADORAS DEL MODULO GPIB

;*****************************************************************
;*****************************************************************

; MACROS DE TEMPORIZACION
;retardo de aproximadamente 2micro seg
T1
          NOP
          NOP
          return
;*****************************************************************
;retardo de aproximadamente 100 micro seg
microsegundos100:
;2*8*0.2us*33=1000us=1ms
lazo_rms2m
          bcf                 STATUS ,RP0         ;banco 0
          movlw     0x2                           ;2 decimal
          movwf     CONTADOR2
lazo_rms1m
          movlw     0x8                           ;8 decimal
          movwf     CONTADOR1
lazo_rms0m
          decfsz    CONTADOR1,F         ;un ciclo
          goto      lazo_rms0m
          decfsz    CONTADOR2,F
          goto      lazo_rms1m
          bsf                 STATUS,RP0          ;banco 1
          decfsz    micro_segundos,F
          goto      lazo_rms2m
          bcf                 STATUS,RP0          ;banco 0
          return
;*****************************************************************
;*****************************************************************
configurar_microcontrolador
          movlw     0x07                ;configurar el puerto A y E en modo digital
          bsf                 STATUS,RP0          ;banco 1
          movwf     ADCON1
          clrf      TRISTATE  ;como salida
          clrf      TRISA
          bsf                 TRISA,4
          ;configuracion de lineas del RTL
          movlw     b'00000001'         ;SRQ en RB0
          movwf     ADDR                ;lineas de direcciones RTL
          clrf      CONTROL_RTL         ;lineas de control RTL



;-------  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
;         movlw     .25                           ;Cristal de 4MHz
          movwf     SPBRG               ;9600 baudios con Fosc=4MHz

          ;-------  configuracion de USART en RX  -----------
          bcf                 STATUS,RP0          ;Seleccióna el banco 0
          movlw     b'10010000'         ;configuramos el RX y USART ponemos en On
          movwf     RCSTA

          bsf                 STATUS,RP0          ;banco 1
          bsf                 PIE1,RCIE ;Habilitamos la interuccion de RX

          bcf                 STATUS,RP0          ;Seleccióna el banco 0
          clrf      CONTROL_RTL
          clrf      ADDR
          bcf       CONTROL,RESET       ;output_low(RESET)Pull Reset Low to complete reset cycle
          bsf       CONTROL_RTL,IOW     ;output_high(IOW)Set IOW and IOR high (they are active low)
          bsf       CONTROL_RTL,IOR     ;output_high(IOR);
```

```
         call      configuracionT
;***************** inicializar variable IMPAR
         bsf               STATUS,RP0          ;banco 1
         clrf      IMPAR
         bcf               STATUS,RP0          ;banco 0
;*****************
         return
;*********************************************************************
;*********************************************************************
;configuracion_T:
;         se encarga de colocar las lineas del controlador en modo TALKER
;         SRQ DAV EOI NRFD NDAC ATN IFC REN
;         RB0 RA0 RC0 RA1  RA2  RB6 RA3 RB7
;de la siguiente forma
;DIO1-DIO8          salidas
;DAV, EOI           salidas
;NRFD, NDAC                 entradas
;ATN, IFC,REN       salidas
;SRQ                        entrada
;1 entrada
;0 salida

configuracionT
         ;configurar los tri-state
;*****************
         bsf       TRISTATE,1
         bcf       TRISTATE,2
;*****************

;         bsf       TRISTATE,1
;         bcf       TRISTATE,2
;         bcf       TRISTATE,3
;         bsf       TRISTATE,4
;         bcf       TRISTATE,5

         bsf               STATUS, RP0         ; Seleccionamos banco 1
         clrf      DIODATO                     ;DIO salida
         movlw     b'00000001'
         movwf     CONTROL
         movlw     b'00010110'
         movwf     HANDSHAKE
         bcf               PORT_EOI,EOI        ;configurar linea EOI como salida
         bcf               STATUS, RP0         ; Seleccionamos banco 0

         bsf               HANDSHAKE,DAV

         bsf               PORTLATCH_ATN,LATCH_ATN
         bsf               CONTROL,ATN                 ;ATN en falso
         bcf               PORTLATCH_ATN,LATCH_ATN
         bcf               PORT_EOI,EOI        ;EOI en falso (no fin de transmision)
         bsf               PORT_IFC,IFC        ;IFC en falso
         movlw     0x00
         movwf     DIODATO                     ;NUL, verdadero
         return
;*********************************************************************
;*********************************************************************
;configuracionL:
;                  se encarga de colocar las lineas del controlador modo LISTENER
;         SRQ DAV EOI NRFD NDAC ATN IFC REN
;         RB0 RA0 RC0 RA1  RA2  RB6 RA3 RB7
;
;DIO1-DIO8          entradas
;DAV, EOI           entradas
;NRFD, NDAC                 salidas
;ATN, IFC,REN       salidas
;SRQ                        entrada
;1 entrada
;0 salida

configuracionL
;configurar los tri-state
;*****************
         bsf       TRISTATE,1
         bsf       TRISTATE,2
;*****************

;         bsf       TRISTATE,1
;         bcf       TRISTATE,2
;         bsf       TRISTATE,3
;         bcf       TRISTATE,4
;         bcf       TRISTATE,5

         bsf               STATUS, RP0         ; Seleccionamos banco 1
         movlw     b'11111111'
         movwf     DIODATO                     ;configurar lineas DIO entrada
         movlw     b'00000001'
         movwf     CONTROL
         movlw     b'00010001'                 ;DAV y EOI entradas; NRFD, NDAC salidas
         movwf     HANDSHAKE
         bsf               PORT_EOI,EOI        ;configurar linea EOI como entrada
         bcf               STATUS, RP0                 ; Seleccionamos banco 0

         movlw     b'00000110'
```

```
          iorwf     HANDSHAKE,F                 ;SET NRFD y NDAC
          return
;*********************************************************************
;*********************************************************************
inicializar_GPIB
          ;almacenamos la configuracion de la tarjeta tal como:
          ;Direccion IP
          ;Direccion MAC
          ;Puerto TCP
          ;inicializamos la interfase GPIB
          call      configuracionT     ;poner al controlador como TALKER
          call      enviar_IFC                  ;limpiar la interfaz
          call      enviar_REN                  ;habilitar la configuracion remota
          call      enviar_LLO
          call      enviar_DCL
;         call      enviar_UNL

          ;incf      FSR,F                       ;apuntar a la direccion IP0
          ;movf      INDF,W
          ;movwf     my_ip0
          ;incf      FSR,F                       ;apuntar a la direccion IP1
          ;movf      INDF,W
          ;movwf     my_ip1
          ;incf      FSR,F                       ;apuntar a la direccion IP2
          ;movf      INDF,W
          ;movwf     my_ip2
          ;incf      FSR,F                       ;apuntar a la direccion IP3
          ;movf      INDF,W
          ;movwf     my_ip3

          ;incf      FSR,F                       ;apuntar a la direccion MAC0
          ;movf      INDF,W
          ;movwf     physical_address0
          ;incf      FSR,F                       ;apuntar a la direccion MAC0
          ;movf      INDF,W
          ;movwf     physical_address1
          ;incf      FSR,F                       ;apuntar a la direccion MAC0
          ;movf      INDF,W
          ;movwf     physical_address2
          ;incf      FSR,F                       ;apuntar a la direccion MAC0
          ;movf      INDF,W
          ;movwf     physical_address3
          ;incf      FSR,F                       ;apuntar a la direccion MAC0
          ;movf      INDF,W
          ;movwf     physical_address4
          ;incf      FSR,F                       ;apuntar a la direccion MAC0
          ;movf      INDF,W
          ;movwf     physical_address5

          ;incf      FSR,F                       ;apuntar a la direccion GPIB del Inst 1

;         call      enviar_UNL
          return
;*********************************************************************
enviar_IFC
          ;limpiamos la interfase GPIB
          bcf                 PORT_IFC,IFC        ;activar IFC (verdadero)
          movlw     0x3                                 ;numero de 100 veces uS
          bsf                 STATUS,RP0                  ;banco 1
          movwf     micro_segundos
          bcf                 STATUS,RP0                  ;banco 0
          call      microsegundos100;durante 100us o mas
          bsf                 PORT_IFC,IFC        ;desactivar IFC (falso)
          return
;*********************************************************************
enviar_LLO
          movlw     LLO                                 ;comando universal
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;ATN en falso
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          return
;*********************************************************************
enviar_REN
;         bsf                 PORTLATCH_REN,LATCH_REN
          bcf       CONTROL,REN
;         bcf                 PORTLATCH_REN,LATCH_REN
          return
;*********************************************************************
enviar_DCL
          ;limpiamos el dispositivo
          movlw     DCL                                 ;comando universal
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;ATN en falso
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          return
```

```
;*********************************************************************
enviar_UNL
          movlw     UNL                                     ;comando universal
          bsf                 STATUS,RP0                    ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                    ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                   ;ATN en falso
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          return
;*********************************************************************
enviar_UNT
          movlw     UNT                                     ;comando universal
          bsf                 STATUS,RP0                    ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                    ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                   ;ATN en falso
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          return

;*********************************************************************
; AH: función de interfaz Acceptor Handshake
AH:

          ;se establecen las lineas NRFD y NDAC verdaderas (nivel LOW)
          ;bsf      HANDSHAKE,NRFD      ;"recordar que el buffer invierte esta señal"
          ;bsf      HANDSHAKE,NDAC      ;"recordar que el buffer invierte esta señal"

          movlw     b'00000110'
          iorwf     HANDSHAKE,F

          ;se establece la linea NRFD en falso (nivel HIGH)
          bcf                 HANDSHAKE,NRFD      ;"recordar que el buffer invierte esta señal"

          ;revisar sí DAV está en verdadero (nivel LOW)
          ;movlw    0xff                           ;255 decimal
          ;movwf    CONTADOR2
lazoAH1:
          ;nop
          ;nop
          ;nop
          ;nop
          ;decf     CONTADOR2,F
          ;btfsc    STATUS,Z
          ;goto     salirAH1
          btfss     HANDSHAKE,DAV
          goto      lazoAH1                         ;regresar y esperar a que este en nivel LOW

          ;movlw    '@'
          ;call     RS232_Tx

          ;ahora se puede leer el dato presente en las lineas DIO
          ;primero avisamos que no estamos listo para un nuevo dato (LOW)
          bsf                 HANDSHAKE,NRFD      ;"recordar que el buffer invierte esta señal"
;**********************
          btfss     PORT_EOI,EOI        ;obtiene si es el final de la transferencia
          goto      seguir_RD1                      ;de una secuencia de multiples bytes
          movlw     TRUE
          bsf                 STATUS,RP0                    ;banco 1
          movwf     ESTADO_EOI
          bcf                 STATUS,RP0                    ;banco 0
seguir_RD1:
;**********************
          ;leemos el dato en las lineas DIO
          movf      DIODATO,W           ;leemos el puerto C
          bsf                 STATUS,RP0                    ;banco 1
          movwf     DATO_GPIB           ;y almacenamos su contenido
          comf      DATO_GPIB,F                     ;quitamos la negacion del standar
          ;indicamos dato aceptado con NDAC en falso (nivel HIGH)
          bcf                 STATUS,RP0                    ;banco 0
          bcf                 HANDSHAKE,NDAC      ;"recordar que el buffer invierte esta señal"

          ;esperamos a que retiren el dato de las lineas DIO
          ;DAV debe ser falso (nivel HIGH)
          ;movlw    0xff                           ;255 decimal
          ;movwf    CONTADOR2
lazoAH2:
          ;nop
          ;nop
          ;nop
          ;nop
          ;decf     CONTADOR2,F
          ;btfsc    STATUS,Z
          ;goto     salirAH1
          btfss     HANDSHAKE,DAV       ;esta DAV en estado HIGH
          goto      lazoAH2                         ;si no esperar

          ;movlw    '#'
          ;call     RS232_Tx
salirAH1
```

```
          ;si DAV esta en estado alto indicamos que esta listo para un nuevo ciclo
          ;colocando NDAC verdadero (nivel LOW)

          bsf                 HANDSHAKE,NDAC      ;"recordar que el buffer invierte esta señal"


          retlw     HECHO
;*******************************************************************************
;*******************************************************************
;*******************************************************************

; SH: función de interfaz Source Handshake

SH:

SIDS
          ;ponemos linea DAV en falso (nivel HIGH) indicando que no hay dato valido
          bcf                 HANDSHAKE,DAV       ;"recordar que el buffer invierte esta señal"
          ;se revisa si las lineas NRFD y NDAC son verdaderas(nivel LOW)

;         movf      HANDSHAKE,W
;         andlw     b'00011000'
;         sublw     b'00011000'
;         btfss     STATUS,Z
;         goto      SIDS




          btfss     HANDSHAKE,NRFD
          goto      SIDS
          btfss     HANDSHAKE,NDAC
          goto      SIDS

          ; colocar el dato a transmitir en las lineas DIO
lazoSH1:
          bsf                 STATUS,RP0                  ;banco 1
          comf      DATO_GPIB,F                 ;negar el byte
          movf      DATO_GPIB,W
          bcf                 STATUS,RP0                  ;banco 0
          movwf     DIODATO

          ;censar si NRFD es falso (nivel HIGH)
          ;movlw    0xff                        ;255 decimal
          ;movwf    CONTADOR2
lazoSH2:
          ;nop
          ;nop
          ;nop
          ;nop
          ;decf     CONTADOR2,F
          ;btfsc    STATUS,Z
          ;goto     salirSH1
          btfsc     HANDSHAKE,NRFD
          goto      lazoSH2

          ;establecer linea DAV verdadero (nivel LOW), indicando dato valido
          bsf                 HANDSHAKE,DAV       ;"recordar que el buffer invierte esta señal"

          ;censar la linea NDAC en falso (nivel HIGH)
          ;que indica que aceptor ya acepto el dato
          ;movlw    0xff                        ;255 decimal
          ;movwf    CONTADOR2
lazoSH3:
          ;nop
          ;nop
          ;nop
          ;nop
          ;decf     CONTADOR2,F
          ;btfsc    STATUS,Z
          ;goto     salirSH1
          btfsc     HANDSHAKE,NDAC      ;continua si NDAC es HIGH
          goto      lazoSH3

salirSH1
          return
;*******************************************************************
;*******************************************************************
;
;funcion_RL
;                   Entradas:DIRECCION:direccion del dispositivo
;                                       REMOTE=TRUE---dispositivo en "Remote"
;                                       REMOTE=FALSE--dispositivo en "Local

funcion_RL
          call      configuracionT      ;Controlador en modo TALKER
          ;dispositivo en "Remote"
          call      enviar_REN                  ;enviar mensaje REN
          movlw     UNL
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
```

```
          bcf                 CONTROL,ATN                 ;indicar mensaje de interfas
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          bsf                 STATUS,RP0                  ;banco 1
          movf      DIRECCION,W
          movwf     MLA
          movlw     0x20                        ;obtener direccion LISTEN
          addwf     MLA,F
          movf      MLA,W
          movwf     DATO_GPIB           ;enviar direccion LISTEN
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;pedir atencion
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          movlw     TRUE
          bsf                 STATUS,RP0                  ;banco 1
          subwf     REMOTE,W
          bcf                 STATUS,RP0                  ;banco 0
          btfss     STATUS,Z            ;en remoto o local?
          goto      ir_local            ;ir a local
          call      enviar_LLO

          goto      salir_RL
ir_local

          movlw     GTL                                   ;enviar mensaje GTL
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;pedir atencion
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
;         bsf                 PORTLATCH_REN,LATCH_REN
          bsf                 CONTROL,REN                 ;enviar REN false (LOW)
;         bcf                 PORTLATCH_REN,LATCH_REN
salir_RL
          return

;*************************************************
;FUNCION PARA TRANSMISION DE DATOS POR RS232        *
;Varables de entrada:                                                                    *
;         - W Conteniendo el byte a transmitir                    *
;*************************************************
RS232_Tx
          bcf                 STATUS,RP0                  ;Selecciòna el banco 0
          movwf     TXREG                       ;Almacena el byte a transmitir

          bsf                 STATUS,RP0                  ;Selecciòna el banco 1
Tx_Wait
          btfss     TXSTA,TRMT                  ;TXSTA,TRMT         ;Byte transmitido ??
          goto      Tx_Wait                     ;No, esperar
          bcf                 STATUS,RP0                  ;Selecciòna el banco 0

          return


;*************************************************
;         FUNCION PARA RECEPCIÒN DE DATOS POR RS232                *
;*************************************************
RS232_Rx
          movf      RCREG,W
          return


;*********************************************************************
;analizar_PCmensajes
;                   analiza el paquete de mensajes que la PC
;                   le envia a la tarjeta GPIB

analizar_PCmensajes
          ;mostrando por el puerto serie lo que se recibe
          movlw     BUFFER_ENTRADA      ;apuntar a la primera posicion del buffer
          movwf     FSR                                   ;para leer tipo de mensaje
          decf      FSR,F
mostrar1
          incf      FSR,F
          movf      INDF,W
          call      RS232_Tx
          movlw     '$'
          subwf     INDF,W
          btfss     STATUS,Z
          goto      mostrar1

          ;decodificando el Paquete de datos de llegada
          movlw     BUFFER_ENTRADA      ;apuntar a la primera posicion del buffer
          movwf     FSR
          ;comparar si es del tipo INIT "inicializar tarjeta"
          movlw     INIT
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoPCT1
```

```
          call      inicializar_GPIB
          goto      salirPCT
lazoPCT1

          ;comparar si es del tipo TEM "tarjeta enviar mensajes al bus"
          movlw     TEM
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoPCT2


          enviar_mensajes
          ;verificar que tipo de comando se va a enviar
          incf      FSR,F

          ;comparar si es COMANDO DE INTERFAZ
          movlw     COMANDO_I
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoED1
          ;enviar el comando de interfaz

          incf      FSR,F                       ;ignorar el campo TIPO
          incf      FSR,F                       ;ignorar el campo direccion del dispositivo
          incf      FSR,F                       ;apuntar al codigo del comando de interfaz
          call      configuracionT

          ;*******************
          ;enviar el comando UNL
          ;call     configuracionT      ;colocarse en estado TALKER
          ;anular todos los LISTENER
          movlw     UNL
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;indicar comando de interfaz
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          ;direccionar al dispositivo deseado como LISTENER
          ;sumandole          0x20
          bsf                 STATUS,RP0                  ;banco 1
          movf      DIRECCION,W
          movwf     MLA
          movlw     0x20
          addwf     MLA,F
          movf      MLA,W
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;indicar comando de interfaz
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          ;*******************

          ;*******************
          movlw     0X30
          subwf     INDF,F
          ;*******************
          movf      INDF,W
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;indicar que el dato es un comando de interfaz
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH
          bsf                 STATUS,RP0                  ;banco 1
          movwf     ERRORES                     ;guardar el tipo de error o si es HECHO
          bcf                 STATUS,RP0                  ;banco 0
          andlw     b'11111110'                 ;si no retorna HECHO entonces cargar ERROR
          btfsc     STATUS,Z            ;si resulta zero es HECHO
          goto      lazoEC1
          bsf                 STATUS,RP0                  ;banco 1
          bsf                 ESTADO_GPIB,ER      ;activamos bandera de error
          bcf                 STATUS,RP0                  ;banco 0
          goto      salirEC
lazoEC1
          bsf                 STATUS,RP0                  ;banco 1
          bcf                 ESTADO_GPIB,ER      ;no hay error
          bcf                 STATUS,RP0                  ;banco 0
salirEC

          goto      salirED                     ;salir, comando enviado

lazoED1

          ;comparar si es COMANDO DE DISPOSITIVO
          movlw     COMANDO_D
          subwf     INDF,W
          btfss     STATUS,Z
          goto      salirED
          ;enviar el comando dependiente de dispositivo
```

```
        incf    FSR,F                   ;almacenar el campo PETICION
        movf    INDF,W
        bsf             STATUS,RP0              ;banco 1
        movwf   PETICION
        bcf             STATUS,RP0              ;banco 0
        incf    FSR,F                   ;almacenar el campo direccion del dispositivo
        movf    INDF,W
        bsf             STATUS,RP0              ;banco 1
        movwf   DIRECCION
        movlw   0x30
        subwf   DIRECCION,F
        bcf             STATUS,RP0              ;banco 0
        incf    FSR,F                   ;almacenar el campo N° de lineas
        movf    INDF,W
        bsf             STATUS,RP0              ;banco 1
        movwf   NLINEAS
        movlw   0x30
        subwf   NLINEAS,F
        movf    NLINEAS,W
        movwf   NLINEAS_R       ;numero de lineas a recibir
        movwf   NLINEAS_E       ;numero de lineas a enviar al PC
        bcf             STATUS,RP0              ;banco 0

        call    configuracionT
        ;enviar el comando UNL
        ;call   configuracionT  ;colocarse en estado TALKER
        ;anular todos los LISTENER
        movlw   UNL
        bsf             STATUS,RP0              ;banco 1
        movwf   DATO_GPIB
        bcf             STATUS,RP0              ;banco 0
        bsf             PORTLATCH_ATN,LATCH_ATN
        bcf             CONTROL,ATN             ;indicar comando de interfaz
        bcf             PORTLATCH_ATN,LATCH_ATN
        call    SH
        ;direccionar al dispositivo deseado como LISTENER
        ;sumandole      0x20
        bsf             STATUS,RP0              ;banco 1
        movf    DIRECCION,W
        movwf   MLA
        movlw   0x20
        addwf   MLA,F
        movf    MLA,W
        movwf   DATO_GPIB
        bcf             STATUS,RP0              ;banco 0
        bsf             PORTLATCH_ATN,LATCH_ATN
        bcf             CONTROL,ATN             ;indicar comando de interfaz
        bcf             PORTLATCH_ATN,LATCH_ATN
        call    SH
        ;inicia la transferencia de los datos
lazoED3
        incf    FSR,F                   ;apuntamos al dato
        bsf             STATUS,RP0              ;banco 1
        movf    SEPARADOR_CONTROLADOR,W
        bcf             STATUS,RP0              ;banco 0
        subwf   INDF,W
        btfss   STATUS,Z        ;verificar si es el caracter de fin de transmision
        goto    lazoED4
        bsf             PORT_EOI,EOI    ;EOI en verdadero (fin de la transmision)
        bsf             STATUS,RP0              ;banco 1
        movf    SEPARADOR_ENTRADA,W
        bcf             STATUS,RP0              ;banco 0
        movwf   INDF
lazoED4

        ;***********************
        movf    INDF,W
        call    RS232_Tx
        ;***********************
        movf    INDF,W
        bsf             STATUS,RP0              ;banco 1
        movwf   DATO_GPIB
        bcf             STATUS,RP0              ;banco 0
        bsf             PORTLATCH_ATN,LATCH_ATN
        bsf             CONTROL,ATN             ;indicar mensaje dependiente de dispositivos
        bcf             PORTLATCH_ATN,LATCH_ATN
        call    SH
        bcf             PORT_EOI,EOI    ;EOI en falso (no fin de transmision)

        bsf             STATUS,RP0              ;banco 1
        movf    SEPARADOR_ENTRADA,W
        bcf             STATUS,RP0              ;banco 0
        subwf   INDF,W
        btfss   STATUS,Z        ;verificar si es el caracter de fin de transmision
        goto    lazoED3

        ;revisar si se espera respuesta del dispositivo
        movlw   0x30
        bsf             STATUS,RP0              ;banco 1
        subwf   PETICION,W
        bcf             STATUS,RP0              ;banco 0
        btfsc   STATUS,Z
        goto    no_respuesta
```

```
          ;si espera respuesta entonces llamar a recibir_datos
si_respuesta
          bsf                STATUS,RP0                  ;banco 1
          movf      PETICION,W                  ;indicar tipo de dato de retorno de peticion
          movwf     TIPODATO
          bcf                STATUS,RP0                  ;banco 0
          ;***hasta aqui he agregado
          call      recibir_datosGPIB
no_respuesta


salirED
          goto      salirPCT
lazoPCT2

          ;comparar si es  TEML "tarjeta enviar mensajes de linea al bus"
          movlw     TEML
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoPCT3
          call      enviar_mensajeL
          goto      salirPCT
lazoPCT3

          ;comparar si es del tipo TRD "tarjeta recibir datos del bus"
          movlw     TRD
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoPCT4

          incf      FSR,F                       ;almacenar el campo direccion del dispositivo
          movf      INDF,W
          bsf                STATUS,RP0                  ;banco 1
          movwf     DIRECCION
          ;*****************
          movlw     0x30
          subwf     DIRECCION,F
          ;*****************
          bcf                STATUS,RP0                  ;banco 0
          incf      FSR,F                       ;almacenar el campo N° de lineas
          movf      INDF,W
          bsf                STATUS,RP0                  ;banco 1
          movwf     NLINEAS
          ;*********************************
          movlw     0x30
          subwf     NLINEAS,F
          movf      NLINEAS,W
          ;*********************************

          movwf     NLINEAS_R           ;numero de lineas a recibir
          movwf     NLINEAS_E           ;numero de lineas a enviar al PC
          ;bcf                STATUS,RP0                  ;banco 0
          ;bsf                STATUS,RP0                  ;banco 1
          movlw     0x34                ;indicar 34
          movwf     TIPODATO
          bcf                STATUS,RP0                  ;banco 0
          call      recibir_datosGPIB
          goto      salirPCT
lazoPCT4

          ;comparar si es  PDR (Programar al dispositivo en modo "Remote")
          movlw     PDR
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoPCT5

          incf      FSR,F                       ;almacenar el campo direccion del dispositivo
          movf      INDF,W
          bsf                STATUS,RP0                  ;banco 1
          movwf     DIRECCION
          ;*****************
          movlw     0x30
          subwf     DIRECCION,F
          ;*****************
          bcf                STATUS,RP0                  ;banco 0
          incf      FSR,F                       ;almacenar el modo de operacion remoto o local
          movf      INDF,W
          bsf                STATUS,RP0                  ;banco 1
          movwf     REMOTE
          bcf                STATUS,RP0                  ;banco 0
          call      funcion_RL

          goto      salirPCT
lazoPCT5
;comparar si es del tipo CONFIGURAR TARJETA
          movlw     CONFIG_TAR                  ;valor=6
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoPCT6
          ;si se trata del campo CONFIGURAR TARJETA
          incf      FSR,F
          ;revisar que tipo de CONFIGURACION
          movlw     CONFIG_IP ;es un reset GPIB
```

```
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoCONFIG0
          call      configurar_IP
          goto      salirPCT
lazoCONFIG0
          movlw     CONFIG_MAC          ;es un reset GPIB
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoCONFIG1
          call      configurar_MAC
          goto      salirPCT
lazoCONFIG1
          movlw     CONFIG_PORT         ;es un reset GPIB
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoCONFIG2
          call      configurar_PORT
          goto      salirPCT
lazoCONFIG2
          movlw     RESET_GPIB          ;es un reset GPIB
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoCONFIG3
          bsf               PCLATH,4
          bcf               PCLATH,3            ;PAGINA 2 DE PROGRAMA
          call      ini_MODGPIB
          bcf               PCLATH,4
          bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA
          goto      salirPCT
lazoCONFIG3
          movlw     CONFIG_TCP          ;es un reset GPIB
          subwf     INDF,W
          btfss     STATUS,Z
          goto      lazoCONFIG4
          call      ini_MODTCP
          goto      salirPCT
lazoCONFIG4

lazoPCT6
salirPCT
          movlw     BUFFER_ENTRADA
          movwf     FSR
          return
;*********************************************************************
;*********************************************************************

;*********************************************************************
;*********************************************************************
enviar_mensajeL

          return
;*********************************************************************
;*********************************************************************
recibir_datosGPIB:  ;OK
;recibir datos provemientes del bus GPIB
          bsf               STATUS,RP0                  ;banco 1
          movf      NLINEAS,W
          movwf     NLINEAS_R
          bcf               STATUS,RP0                  ;banco 0
          call      configuracionT      ;colocarse en estado TALKER
          ;anular todos los TALKER
          movlw     UNT
          bsf               STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf               STATUS,RP0                  ;banco 0
          bsf               PORTLATCH_ATN,LATCH_ATN
          bcf               CONTROL,ATN                 ;indicar comando de interefaz
          bcf               PORTLATCH_ATN,LATCH_ATN
          call      SH

          ;direccionar al dispositivo deseado como TALKER
          ;sumandole 0x40
          bsf               STATUS,RP0                  ;banco 1
          movf      DIRECCION,W
          movwf     MTA
          movlw     0x40
          addwf     MTA,F
          movf      MTA,W
          movwf     DATO_GPIB
          bcf               STATUS,RP0                  ;banco 0
          call      SH
          call      configuracionL      ;colocar el controlador en LISTEN
          bsf               PORTLATCH_ATN,LATCH_ATN
          bsf               CONTROL,ATN                 ;indicar comando de dispositivos
          bcf               PORTLATCH_ATN,LATCH_ATN

          bsf               STATUS,IRP                  ;banco 2,3(direccionamiento indirecto)
;************************************
          movlw     FALSE
          bsf               STATUS,RP0                  ;banco 1
          movwf     ESTADO_EOI
          bcf               STATUS,RP0                  ;banco 0
```

```
;************************************
          movlw     BUFFER_SALIDA       ;inicializar el buffer para almacenarlos
          movwf     FSR
          bsf                 STATUS,RP0                  ;banco 1
          movf      DIRECCION,W
          movwf     INDF
          movlw     0x30
          addwf     INDF,F                      ;colocar direccion del instrumento
;***esto he agregado
          incf      FSR,F                       ;reservar un byte para el tipo de dato
          movf      TIPODATO,W
          movwf     INDF
          incf      FSR,F                       ;enviar el byte de estado STB
          movf      STB,W
          movwf     INDF
          incf      FSR,F                       ;reservar para indicar ultimo paquete
          movlw     FALSE
          movwf     INDF                        ;indicar FALSE = no es ultimo paquete
;         clrf      CONTADOR_BYTES
;         movlw     0x4C                        ;prueba cargar 4C hex=76 dec para que siempre
;         movwf     CONTADOR_BYTES      ;divida el paquete en dos partes
          clrf      CONTADOR_BYTES      ;inicializar el contador en cero
;***hasta aqui he agregado
lazo_RD1
          bcf                 STATUS,RP0                  ;banco 0

          ;movlw    '*'
          ;call     RS232_Tx

          call      AH
          incf      FSR,F                       ;apuntar al siguiente byte para los datos
          bsf                 STATUS,RP0                  ;banco 1
          movf      DATO_GPIB,W
          bcf                 STATUS,RP0                  ;banco 0
          movwf     INDF

          bsf                 STATUS,RP0                  ;banco 1
          movf      DATO_GPIB,W
          call      RS232_Tx

          ;movlw    0x00                        ;NULL
          ;subwf    INDF,W
          ;btfsc    STATUS,Z
          ;goto     salir_RD

          bsf                 STATUS,RP0                  ;banco 1
          incf      CONTADOR_BYTES,F

          movlw     TRUE
          subwf     ESTADO_EOI,W
          btfsc     STATUS,Z
          goto      SEGUIR                      ;se detecto la linea EOI verificar numero de
                                                          ;lineas y finalizar enviando los datos
;************************************
          ;comparacion
          movlw     0x5B                        ;91 dec =  5B hex
          subwf     CONTADOR_BYTES,W    ;compara si esta lleno el buffer de salida
          btfss     STATUS,Z            ;saltar si son diferentes
          goto      lazo_RD1            ;seguir con la ejecucion del programa
          bcf                 STATUS,RP0                  ;banco 0
          incf      FSR,F                       ;apuntar al siguiente byte para los datos
          movlw     0x0a                        ;indicar que es el ultimo paquete colocando
          movwf     INDF                        ;0x0a (avance de linea) al final del buffer
                                                          ;y luego enviar el paquete

          movlw     BUFFER_SALIDA       ;inicializar el buffer para almacenarlos
          movwf     FSR
          movlw     0x03                        ;sumarle 3 al FSR para que apunte al
          addwf     FSR,F                       ;registro que indica si es ultimo paquete
          movlw     FALSE
          movwf     INDF                        ;indicar FALSE = no es ultimo paquete

          bcf                 STATUS,RP0                  ;banco 0
          call      BUS_DATOS_RTL       ;ceder bus de datos al modulo RTL

          bsf                 INTCON,T0IE                 ;habilitar interrupciones del timer 0
          bsf                 INTCON,GIE
nop
nop
nop
nop
nop
nop
nop
nop
nop

          bcf                 INTCON,T0IE                 ;habilitar interrupciones del timer 0
          bcf                 INTCON,GIE


          call      enviar_datos_TCP
```

```
         bsf                INTCON,T0IE                ;habilitar interrupciones del timer 0
         bsf                INTCON,GIE
nop
nop
nop
nop
nop
nop
nop
nop
nop

         bcf                INTCON,T0IE                ;habilitar interrupciones del timer 0
         bcf                INTCON,GIE

         bsf                TRISTATE,1                 ;ceder el bus de datos al modulo GPIB

         ;seleccionar denuevo banco 2,3 por que en la rutina
         ;enviar_datos_TCP fue seleccionado en bancos 0,1...
         ;apuntar denuevo el buffer de salida en la posicion
         ;4 para seguir escribiendo los datos restantes.
         bsf                STATUS,IRP                 ;banco 2,3(direccionamiento indirecto)
         movlw     BUFFER_SALIDA       ;inicializar el buffer para almacenarlos
         movwf     FSR
         movlw     0x03                        ;sumarle 3 al FSR para que apunte al
         addwf     FSR,F                       ;registro que indica si es ultimo paquete
         bsf                STATUS,RP0                 ;banco 1
;        clrf      CONTADOR_BYTES      ;iniciar conteo para los datos restantes
;        movlw     0x4C                        ;prueba cargar 4C hex=76 dec para que siempre
;        movwf     CONTADOR_BYTES      ;divida el paquete en dos partes

         clrf      CONTADOR_BYTES      ;inicializar el contador en cero
         goto      lazo_RD1
;************************************
SEGUIR
         decf      NLINEAS_R,F                ;decremenar la variable NLINEAS_R
         clrw
         subwf     NLINEAS_R,W                ;si no es cero entonces sustitulle
         btfsc     STATUS,Z            ;el avance de linea por una coma.
         goto      fin_de_transmision
         movlw     0x2c
         movwf     INDF
         movlw     FALSE
         movwf     ESTADO_EOI
;************************************
         goto      lazo_RD1


fin_de_transmision:
;        bsf                STATUS,RP0                 ;banco 1
;        movf      ESTADO_EOI,W
;        call      RS232_Tx
         incf      FSR,F
         movlw     0x0a                        ;el avance de linea es el codigo que detecta
         movwf     INDF                        ;el manejador en java y le indica a este
                                                           ;cuando terminar la lectura de datos
         ;indicar ultimo paquete
         bcf                STATUS,RP0                 ;banco 0
         movlw     BUFFER_SALIDA       ;inicializar el buffer para almacenarlos
         movwf     FSR
         movlw     0x03                        ;sumarle 3 al FSR para que apunte al
         addwf     FSR,F                       ;registro que indica si es ultimo paquete
         movlw     TRUE
         movwf     INDF                        ;indicar TRUE = si es ultimo paquete


;******************;******************;******************
         bcf                STATUS,RP0                 ;banco 0
salir_RD
         ;Enviar los datos recibidos al PC

;*************************
         call      BUS_DATOS_RTL
         call      enviar_datos_TCP
         call      BUS_DATOS_GPIB
;*************************
;        bcf                STATUS,IRP                 ;banco 0,1(direccionamiento indirecto)
         return
;***************************************************************
;***************************************************************
;***************************************************************
;***************************************************************
responder_SRQ

         bsf                STATUS,RP0                 ;banco 1
         ;revisar tipo de encuesta a implementar
         btfss     ESTADO_GPIB,PP      ;si P es verdadera indica Paralela
         goto      viñetaRS1
         bcf                STATUS,RP0                 ;banco 0
         call      encuesta_paralela
         goto      viñetaRS2
viñetaRS1
```

```
          bcf                STATUS,RP0                    ;banco 0
          call      encuesta_serie

          movlw     0x01
          movwf     NLINEAS_E
;         bcf                STATUS,RP0                    ;banco 0
;         movwf     INDF
;         call      enviar_PCmensajes

          movlw     0x40
          bsf                STATUS,RP0                    ;banco 1
          subwf     DIRECCION_CENSO,W
          movwf     DIRECCION
          movf      NLINEAS,W
          movwf     NLINEAS_R
          bcf                STATUS,RP0                    ;banco 0
;***esto he agregado
          bsf                STATUS,RP0                    ;banco 1
          movlw     0x36                          ;indicar dato tipo retorno de SRQ
          movwf     TIPODATO
          bcf                STATUS,RP0                    ;banco 0
;***hasta aqui he agregado
;         call      recibir_datosGPIB

;*****************************************
          movlw     BUFFER_SALIDA       ;inicializar el buffer para almacenarlos
          movwf     FSR
          bsf                STATUS,RP0                    ;banco 1
          movf      DIRECCION,W
          movwf     INDF
          movlw     0x30
          addwf     INDF,F                        ;colocar direccion del instrumento

          incf      FSR,F                         ;tipo de dato
          movf      TIPODATO,W
          movwf     INDF
          incf      FSR,F                         ;enviar el byte de estado STB
          movf      STB,W
          movwf     INDF
          incf      FSR,F                         ;indicar ultimo paquete
          movlw     TRUE
          movwf     INDF
          incf      FSR,F
          incf      FSR,F
          incf      FSR,F
          incf      FSR,F
          movlw     0x0a
          movwf     INDF
          bcf                STATUS,RP0                    ;banco 0
          call      BUS_DATOS_RTL       ;ceder bus de datos al modulo RTL
          call      enviar_datos_TCP
          call      BUS_DATOS_GPIB      ;ceder el bus de datos al modulo GPIB
;*****************************************

viñetaRS2
          return
;*****************************************************************
encuesta_serie

          call      configuracionT      ;colocarse en estado TALKER
          ;anular todos los LISTENER
          movlw     UNL
          bsf                STATUS,RP0                    ;banco 1
          movwf     DATO_GPIB
          bcf                STATUS,RP0                    ;banco 0
          bsf                PORTLATCH_ATN,LATCH_ATN
          bcf                CONTROL,ATN                   ;indicar comando de interfaz
          bcf                PORTLATCH_ATN,LATCH_ATN
          call      SH
          ;habilitar encuesta serie
          movlw     SPE
          bsf                STATUS,RP0                    ;banco 1
          movwf     DATO_GPIB
          bcf                STATUS,RP0                    ;banco 0
          bsf                PORTLATCH_ATN,LATCH_ATN
          bcf                CONTROL,ATN                   ;indicar comando de interfaz
          bcf                PORTLATCH_ATN,LATCH_ATN
          call      SH

          movlw     0x41
          bsf                STATUS,RP0                    ;banco 1
          movwf     DIRECCION_CENSO
          bcf                STATUS,RP0                    ;banco 0

lazo_ES1
          bsf                STATUS,RP0                    ;banco 1
          incf      DIRECCION_CENSO,F
          ;enviar direccion MTAX
          movf      DIRECCION_CENSO,W
          movwf     DATO_GPIB
          bcf                STATUS,RP0                    ;banco 0
          bsf                PORTLATCH_ATN,LATCH_ATN
          bcf                CONTROL,ATN                   ;indicar comando de interfaz
```

```
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH

          ;recibir el byte de estado
          call      configuracionL
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bsf                 CONTROL,ATN
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      AH
          bsf                 STATUS,RP0                  ;banco 1
          movf      DATO_GPIB,W
          movwf     STB
          bcf                 STATUS,RP0                  ;banco 0

          call      configuracionT
          movlw     UNT
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN                 ;indicar comando de interfaz
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH

          bsf                 STATUS,RP0                  ;banco 1
          btfsc     STB,6                       ;revisar si este pidio servicio
          goto      dispositivo_encontrado
          ;pero si no es ese seguir revisando los demas dispositivos
          movlw     0x5f
          subwf     DIRECCION_CENSO,W
          bcf                 STATUS,RP0                  ;banco 0
          btfss     STATUS,Z
          goto      lazo_ES1
dispositivo_no_encontrado
          ;indicar de alguna forma dispositivo NO encontrado
          goto      saliES
dispositivo_encontrado
          bcf                 STATUS,RP0                  ;banco 0
          ;indicar de alguna forma dispositivo encontrado
          call      configuracionT
          movlw     SPD
          bsf                 STATUS,RP0                  ;banco 1
          movwf     DATO_GPIB
          bcf                 STATUS,RP0                  ;banco 0
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 CONTROL,ATN
          bcf                 PORTLATCH_ATN,LATCH_ATN
          call      SH

saliES
          return
;*****************************************************************

;*****************************************************************
BUS_DATOS_RTL               ;bus de datos dederlo al RTL

          bcf                 TRISTATE,1
          ;configurar PORTD como entrada por defecto
          bsf                 STATUS,RP0
          movlw     0xFF
          movwf     DATOS
          bcf                 STATUS,RP0
          return

;*****************************************************************
BUS_DATOS_GPIB              ;bus de datos cederlo al GPIB

          bsf       TRISTATE,1
          call      configuracionT      ;modulo GPIB como Talker por defecto
          return
;*****************************************************************
cargar_IP

          movlw     0x00                        ; Numero IP mas significativo
          bsf                 STATUS,RP0                            ; banco 1
          movwf     direccion_EEPROM
          call      leer_dato
          bcf                 STATUS,RP0                            ; banco 0
          movwf     my_ip0

          movlw     0x01
          bsf                 STATUS,RP0                            ; banco 1
          movwf     direccion_EEPROM
          call      leer_dato
          bcf                 STATUS,RP0                            ; banco 0
          movwf     my_ip1

          movlw     0x02
          bsf                 STATUS,RP0                            ; banco 1
          movwf     direccion_EEPROM
          call      leer_dato
          bcf                 STATUS,RP0                            ; banco 0
          movwf     my_ip2
```

```
         movlw     0x03
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     my_ip3
         return
;*****************************************************************

;*****************************************************************
cargar_MAC
         movlw     0x04                        ; Numero IP mas significativo
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     physical_address0

         movlw     0x05
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     physical_address1

         movlw     0x06
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     physical_address2

         movlw     0x07
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     physical_address3

         movlw     0x08
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     physical_address4

         movlw     0x09
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         bcf               STATUS,RP0                          ; banco 0
         movwf     physical_address5
         return
;*****************************************************************
cargar_PUERTO

         movlw     0x0a                        ; Numero IP mas significativo
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         movwf     my_port0

         movlw     0x0b
         bsf               STATUS,RP0                          ; banco 1
         movwf     direccion_EEPROM
         call      leer_dato
         movwf     my_port1
         bcf               STATUS,RP0                          ; banco 0
         return
;*****************************************************************
guardar_IP
         bcf               STATUS,RP0                          ; banco 0
         ;averiguar cuantos pares vienen
         incf      FSR,F               ;apunto al segundo byte
         movlw     '.'                 ;caracter ascii "."
         subwf     INDF,W
         btfss     STATUS,Z
         goto      nopunto1
espunto1:
         decf      FSR,F
         clrf      numero1_1
         movf      INDF,W
         movwf     numero1_0
         goto      continuarGIP1
nopunto1:
         decf      FSR,F
         movf      INDF,W
         movwf     numero1_1
         incf      FSR,F
         movf      INDF,W
         movwf     numero1_0
continuarGIP1
```

```
          bsf               PCLATH,4
          bcf               PCLATH,3            ;PAGINA 2 DE PROGRAMA
          call      ascii_Hexa
          bcf               PCLATH,4
          bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA

          movwf     my_ip0
          bsf               STATUS,RP0                          ; banco 1
          movwf     dato_EEPROM
          movlw     0x00
          movwf     direccion_EEPROM
          call      almacenar_dato

          bcf               STATUS,RP0                          ; banco 0
          incf      FSR,F                                 ;apunto al punto
          incf      FSR,F                                 ;apunto al primer byte
          incf      FSR,F                                 ;apunto al segundo byte
          movlw     '.'               ;caracter ascii "."
          subwf     INDF,W
          btfss     STATUS,Z
          goto      nopunto2
espunto2:
          decf      FSR,F
          clrf      numero1_1
          movf      INDF,W
          movwf     numero1_0
          goto      continuarGIP2
nopunto2:
          decf      FSR,F
          movf      INDF,W
          movwf     numero1_1
          incf      FSR,F
          movf      INDF,W
          movwf     numero1_0
continuarGIP2
          bsf               PCLATH,4
          bcf               PCLATH,3            ;PAGINA 2 DE PROGRAMA
          call      ascii_Hexa
          bcf               PCLATH,4
          bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA
          movwf     my_ip1
          bsf               STATUS,RP0                          ; banco 1
          movwf     dato_EEPROM
          movlw     0x01
          movwf     direccion_EEPROM
          call      almacenar_dato

          bcf               STATUS,RP0                          ; banco 0
          incf      FSR,F                                 ;apunto al punto
          incf      FSR,F                                 ;apunto al primer byte
          incf      FSR,F                                 ;apunto al segundo byte
          movlw     '.'               ;caracter ascii "."
          subwf     INDF,W
          btfss     STATUS,Z
          goto      nopunto3
espunto3:
          decf      FSR,F
          clrf      numero1_1
          movf      INDF,W
          movwf     numero1_0
          goto      continuarGIP3
nopunto3:
          decf      FSR,F
          movf      INDF,W
          movwf     numero1_1
          incf      FSR,F
          movf      INDF,W
          movwf     numero1_0
continuarGIP3
          bsf               PCLATH,4
          bcf               PCLATH,3            ;PAGINA 2 DE PROGRAMA
          call      ascii_Hexa
          bcf               PCLATH,4
          bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA

          movwf     my_ip2
          bsf               STATUS,RP0                          ; banco 1
          movwf     dato_EEPROM
          movlw     0x02
          movwf     direccion_EEPROM
          call      almacenar_dato


          bcf               STATUS,RP0                          ; banco 0
          incf      FSR,F                                 ;apunto al punto
          incf      FSR,F                                 ;apunto al primer byte
          incf      FSR,F                                 ;apunto al segundo byte
          movlw     '.'               ;caracter ascii "."
          subwf     INDF,W
          btfss     STATUS,Z
          goto      nopunto4
espunto4:
          decf      FSR,F
```

```
          clrf      numero1_1
          movf      INDF,W
          movwf     numero1_0
          goto      continuarGIP4
nopunto4:
          decf      FSR,F
          movf      INDF,W
          movwf     numero1_1
          incf      FSR,F
          movf      INDF,W
          movwf     numero1_0
continuarGIP4
          bsf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 2 DE PROGRAMA
          call      ascii_Hexa
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA

          movwf     my_ip3
          bsf                 STATUS,RP0                              ; banco 1
          movwf     dato_EEPROM
          movlw     0x03
          movwf     direccion_EEPROM
          call      almacenar_dato

          return
;*****************************************************************
guardar_MAC
          return
;*****************************************************************
guardar_PUERTO
          movlw     0x0a
          bsf                 STATUS,RP0                              ; banco 1
          movwf     direccion_EEPROM
          movf      INDF,W
          movwf     dato_EEPROM
          call      almacenar_dato

          incf      FSR,F
          movlw     0x0b
          bsf                 STATUS,RP0                              ; banco 1
          movwf     direccion_EEPROM
          movf      INDF,W
          movwf     dato_EEPROM
          call      almacenar_dato
          return
;*****************************************************************
leer_dato

          bcf                 STATUS,RP1                              ; banco 1
          bsf                 STATUS,RP0                              ; banco 1
          movf      direccion_EEPROM,W            ;direccion a leer
          bsf                 STATUS,RP1                              ; banco 2
          bcf                 STATUS,RP0                              ; banco 2
          movwf     EEADR
          bsf                 STATUS,RP1                              ; banco 3
          bsf                 STATUS,RP0                              ; banco 3
          bcf                 EECON1,EEPGD        ;seleccionamos la memoria de datos
          bsf                 EECON1,RD ;iniciamos la lectura
          btfsc     EECON1, RD          ;esperemos que la lectura termine
          goto      $-1
          bsf                 STATUS,RP1                              ; banco 2
          bcf                 STATUS,RP0                              ; banco 2
          movf      EEDATA,W  ;ponemos el dato leido en W
          ;retorna en el banco 1
          bcf                 STATUS,RP1                              ; banco 1
          bsf                 STATUS,RP0                              ; banco 1
          return
;*****************************************************************
almacenar_dato:

almacenar

          bsf                 STATUS,RP1                              ; banco 3
          bsf                 STATUS,RP0                              ; banco 3
esperarALMD1
          btfsc     EECON1, WR                              ;esperar si hay un proceso de escritura en pogreso
          goto      esperarALMD1
          bcf                 EECON1,WREN                             ;desactivar la funcion de escritura.
          bcf                 STATUS,RP1                              ; banco 1
          bsf                 STATUS,RP0                              ; banco 1
          movf      direccion_EEPROM,W

          bsf                 STATUS,RP1                              ; banco 2
          bcf                 STATUS,RP0                              ; banco 2
          movwf     EEADR                                   ;cargar la direccion

          bcf                 STATUS,RP1                              ; banco 1
          bsf                 STATUS,RP0                              ; banco 1
          movf      dato_EEPROM,W
          bsf                 STATUS,RP1                              ; banco 2
          bcf                 STATUS,RP0                              ; banco 2
          movwf     EEDATA
```

```
          bsf                 STATUS,RP1                          ; banco 3
          bsf                 STATUS,RP0                          ; banco 3
          bcf                 EECON1, EEPGD               ;seleccionar la memoria de datos
          bsf                 EECON1, WREN                ;activar funcion de escritura
          movlw     0x55
          movwf     EECON2                                ;escribir 0x55 en EECON2
          movlw     0xaa
          movwf     EECON2                                ;escribir 0xaa en EECON2
          bsf                 EECON1,WR                   ;iniciar la operacion de escritura.
          bsf                 STATUS,RP1                          ; banco 3
          bsf                 STATUS,RP0                          ; banco 3
esperarALMD2
          btfsc     EECON1, WR                            ;esperar a que el proceso de escritura termine
          goto      esperarALMD2
          bcf                 EECON1,WREN                         ;desactivar la funcion de escritura.

          ;call     leer_dato                   ;leer el byte escrito en la EEPROM para verificar
          ;subwf    dato_EEPROM,W               ;que se guardo correctamente
          ;btfss    STATUS,Z
          ;goto     almacenar
          bcf                 STATUS,RP1                          ; banco 0
          bcf                 STATUS,RP0                          ; banco 0

          return
;*******************************************************************************
;*****    RUTINA PARA INICIALIZAR EL MODULO TCP/IP
ini_MODTCP
          ;inicializar la configuracion del RTL
          ;inicializar direccion ip por defecto
          bcf                 STATUS,RP0                  ;banco 0
          bcf                 INTCON,GIE


          ;inicializacion de estado del socket
          movlw     LISTEN
          bsf                 STATUS,RP0                  ;banco 1
          movwf     html_socket
          movwf     html_socket2
          ;indicar que no hay paquete completo
          bsf                 STATUS,RP0                  ;BANK1
          clrf      FIN_LINEA           ;indicamos por omision no fin de linea
          bcf                 STATUS,RP0                  ;BANK0

          ;guargar la nueva direccion IP y puerto TCP
          incf      FSR,F                       ;apuntar a la direccion IP
          call      guardar_IP
          ;incf     FSR,F                       ;apuntar al valor del PUERTO TCP
          ;call     guardar_PUERTO

          bsf                 INTCON,GIE
          return
;*******************************************************************************
configurar_IP
          bcf                 STATUS,RP0                  ;banco 0
          bcf                 INTCON,GIE
          ;inicializacion de estado del socket
          movlw     LISTEN
          bsf                 STATUS,RP0                  ;banco 1
          movwf     html_socket
          movwf     html_socket2
          ;indicar que no hay paquete completo
          bsf                 STATUS,RP0                  ;BANK1
          clrf      FIN_LINEA           ;indicamos por omision no fin de linea
          bcf                 STATUS,RP0                  ;BANK0
          ;guargar la nueva direccion IP
          incf      FSR,F                       ;apuntar a la direccion IP
          call      guardar_IP
;         bsf                 INTCON,GIE
          return
;*******************************************************************************
configurar_MAC

          return
;*******************************************************************************
configurar_PORT
          bcf                 STATUS,RP0                  ;banco 0
          bcf                 INTCON,GIE
          ;inicializacion de estado del socket
          movlw     LISTEN
          bsf                 STATUS,RP0                  ;banco 1
          movwf     html_socket
          movwf     html_socket2
          ;indicar que no hay paquete completo
          bsf                 STATUS,RP0                  ;BANK1
          clrf      FIN_LINEA           ;indicamos por omision no fin de linea
          bcf                 STATUS,RP0                  ;BANK0
          incf      FSR,F                       ;apuntar al valor del PUERTO TCP
          call      guardar_PUERTO
          bsf                 INTCON,GIE
          return
;*******************************************************************************
```

```
;pagina 2 de programa
          org       0x1000

ini_MODGPIB
          bcf                 STATUS,RP0                    ;banco 0

          bcf                 INTCON,GIE

          ;colocar en un estado inicial la interfase GPIB
          bsf                 PORTLATCH_ATN,LATCH_ATN
          bsf                 CONTROL,ATN                             ;ATN en falso
          bcf                 PORTLATCH_ATN,LATCH_ATN
          bcf                 PORT_EOI,EOI                  ;EOI en falso
          bsf                 PORT_IFC,IFC                  ;IFC en falso
;         bsf                 PORTLATCH_REN,LATCH_REN
          bsf                 CONTROL,REN
;         bcf                 PORTLATCH_REN,LATCH_REN
          movlw     0x00
          movwf     DIODATO                       ;NUL, verdadero
          ;inicializar la interface
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      inicializar_GPIB
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          ;DIRECCION DEL DISPOSITIVO
          bsf                 STATUS,RP0                    ;banco 1
          movlw     0x01                        ;direccion del dispositivo
          movwf     DIRECCION           ;entre 0 y 30
          ;mascara DEL DISPOSITIVO
          movlw     0x31                        ;mascara del dispositivo
          movwf     MASCARA_SRQ                 ;entre 0 y 30
          ;Habilitar encesta serie
          bcf                 ESTADO_GPIB,PP
          ;direccionar en modo remoto al dispositivo
          movlw     TRUE
          movwf     REMOTE
          bcf                 STATUS,RP0                    ;banco 0
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      funcion_RL
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          bsf                 STATUS,RP0                    ;banco 1
          movlw     0x0a      ; CR
          movwf     SEPARADOR_SALIDA
          movlw     0x0a      ; LF
          movwf     SEPARADOR_ENTRADA
          movlw     '$'                 ; $
          movwf     SEPARADOR_CONTROLADOR
          bcf                 STATUS,RP0                    ;banco 0
          bsf                 INTCON,GIE
          return

encuesta_paralela

          return
;*********************************************************************
;ascii_Hex          convierte de ascii a Hexadecimal
;entradas
;                             numero1_0
;                             numero2_0
;salidas
;                             reg W
ascii_Hexa:
          bcf                 STATUS,RP0                    ;banco 0
          movlw     0x57
          subwf     numero1_1,W
          btfss     STATUS,C
          goto      esnumero1
esletra1:
          movlw     0x57
          subwf     numero1_1,F
          goto      continuarACH1
esnumero1:
          movlw     0x30
          subwf     numero1_1,F
continuarACH1:

          movlw     0x57
          subwf     numero1_0,W
          btfss     STATUS,C
          goto      esnumero2
esletra2:
          movlw     0x57
          subwf     numero1_0,F
          goto      continuarACH2
esnumero2:
          movlw     0x30
          subwf     numero1_0,F
continuarACH2:
          bcf                 STATUS,C
```

```
         movlw     0x0f
         andwf     numero1_1,F
         swapf     numero1_1,F
         movf      numero1_1,W
         addwf     numero1_0,W
         bcf               STATUS,RP0                  ;banco 0
         return
```

A 1.3 CODIGO DEL MANEJADOR DEL MODULO TCP/IP

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;*************************************************************
;*************************************************************
;**       CONTROLADOR TCP/IP basado en PIC y RTL8019AS              **
;**       AUTORES:  LUIS LEON                                                                  **
;**                                             OSBALDO CACERES
         **
;**       UNIVERSIDAD DE EL SALVADOR, AÑO 2005                                  **
;**
                           **
;*************************************************************
;*************************************************************

;*************************************************************
;**       CODIGO DE LAS MACROS DEL CONTROLADOR
;*************************************************************

;*************************************************************
;*************************************************************
;*************************************************************
;init_nic:inicializa la tarjeta NIC
;                  entradas: ninguna
;                  salidas: ninguna

init_nic
         PAGE0
         movlw     CR
         movwf     REGISTRO
         movlw     0x21                        ;Page 0, Abort DMA, Stop NIC
         movwf     VALOR
         call      outport                     ;outport(CR, 0x21);

         ;delay_ms(25) esperar para que se apague
         movlw     0x19                        ;25 decimal
         bsf               STATUS,RP0                  ;banco 1
         movwf     m_segundos
         bcf               STATUS,RP0                  ;banco 0
;        bcf               PCLATH,4
;        bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA
         call      retardo_ms
;        bcf               PCLATH,4
;        bcf               PCLATH,3            ;PAGINA 0 DE PROGRAMA
         movlw     NIC_RESET
         movwf     REGISTRO
         movlw     0xFF
         movwf     VALOR
         call      outport             ;outport(NIC_RESET, 0xFF) Reset la NIC

         ;delay_ms(25);     // esperar para el reset
         movlw     0x19                        ;25 decimal
         bsf               STATUS,RP0                  ;banco 1
         movwf     m_segundos
         bcf               STATUS,RP0                  ;banco 0
;        bcf               PCLATH,4
;        bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA
         call      retardo_ms
;        bcf               PCLATH,4
;        bcf               PCLATH,3            ;PAGINA 0 DE PROGRAMA
         movlw     DCR
         movwf     REGISTRO
         movlw     0x48
         movwf     VALOR                       ;byte DMA, Byte order 8086,
         call      outport                     ;outport(DCR, 0x48)
                                               ;Dual 16-bit DMA mode, Normal Operation
                                               ;Send CMD not executed, FIFO threshold 8 bytes

         movlw     RBCR0
         movwf     REGISTRO
         movlw     0x00                        ;limpiar el registro remoto contador de bytes
         movwf     VALOR
         call      outport                     ;outport(RBCR0, 0x00)

         movlw     RBCR1
         movwf     REGISTRO
         movlw     0x00
```

```
          movwf     VALOR
          call      outport                     ;f(RBCR1, 0x00)

          movlw     RCR
          movwf     REGISTRO
          movlw     0x0C                        ;Monitor off, Promiscuo off
          movwf     VALOR
          call      outport                     ;Acceptar Multicast, Acceptar Broadcast
                                                          ;desechar paquetes menores de 64 bytes
                                                                    ;desechar paquetes con error

          movlw     TCR
          movwf     REGISTRO
          movlw     0x02                        ;Internal Loop Back
          movwf     VALOR
          call      outport                     ;outport(TCR, 0x02)

          movlw     BNDRY
          movwf     REGISTRO
          movlw     RCV_BUF_START       ;apunta a inicio del Buffer RAM
          movwf     VALOR
          call      outport                     ;outport(BNDRY,  RCV_BUF_START)

          movlw     PSTART
          movwf     REGISTRO
          movlw     RCV_BUF_START       ;inicio del Buffer RAM
          movwf     VALOR
          call      outport                     ;outport(PSTART, RCV_BUF_START)

          movlw     PSTOP
          movwf     REGISTRO
          movlw     XMT_BUF_START       ;establecemos el tope de la pagina de recepcion
          movwf     VALOR                               ;8-bit mode para el buffer de transmicion
          call      outport                     ;outport(PSTOP,  XMT_BUF_START);

          movlw     ISR
          movwf     REGISTRO
          movlw     0xFF                        ;Limpiamos el registro de estado de interupciones
          movwf     VALOR
          call      outport                     ;outport(ISR, 0xFF)

          movlw     IMR
          movwf     REGISTRO
          movlw     0x01                        ;configuramos la mascara para habilitar interrupciones
          movwf     VALOR                       ;
          call      outport                     ;outport(IMR, 0x05)

          movlw     CR
          movwf     REGISTRO
          movlw     0x61                        ;  Page 1
          movwf     VALOR
          call      outport                     ;outport(CR, 0x61)

;cargar la direccion fisica

          ;Carga la MAC local desde la memoria EEPROM DEL PIC
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      cargar_MAC                    ;cargar MAC desde la EEPROM
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          ;(for i=0;i=5)outport(PAR[i], physical_address[i])
          movlw     PAR0
          movwf     REGISTRO
          movlw     physical_address0
          movwf     FSR
continuar_inip2
          movf      INDF,W
          movwf     VALOR
          call      outport
          movlw     0x06
          subwf     REGISTRO,W
          btfsc     STATUS,Z
          goto      continuar_inip1
          incf      REGISTRO,F
          incf      FSR,F
          goto      continuar_inip2

continuar_inip1
          ;outport(CURR, 0x40);
          movlw     CURR                        ;apuntamos a la pagina del primer buffer de recepcion
          movwf     REGISTRO
          movlw     0x40
          movwf     VALOR
          call      outport

          movlw     CR                                  ;encendemos la Tarjeta
          movwf     REGISTRO
          movlw     0x22
          movwf     VALOR
          call      outport                     ; outport(CR, 0x22)
```

```
    ; We're still in Loop Back so nothing should happen

          movlw     TCR
          movwf     REGISTRO
          movlw     0x00
          movwf     VALOR
          call      outport                     ;outport(TCR, 0x00)save this for later in main
          return
;*****************************************************************************
;*****************************************************************************
;inport:
;         entrada: REGISTRO, tiene la direccion del registro a leer
;         SALIDA:   VALOR tiene el valor o contenido del registro

inport
          PAGE0
          ;****************************
          bcf                 STATUS,0            ;limpiar el bit de acarreo
          rlf                 REGISTRO,W
          ;****************************
;         movf      REGISTRO,W
          movwf     ADDR                        ;PORTB=ADDR direccion del registro a leer
          bcf                 CONTROL_RTL,IOR     ;output_low(IOR) activamos para peticion de lectura
          ;previamente el PORTD se configura como entrada
          movf      DATOS,W                     ;leemos el dato en el puerto D
          movwf     VALOR                       ;VALOR = DATOS=PORTD y lo almacenamos
          bsf                 CONTROL_RTL,IOR     ;output_high(IOR)Desactivamos la peticion de lectura
          PAGE0
          return

;*****************************************************************************
;outport:coloca un byte de datos a la direccion specifica
;         entrada:- REGISTRO, contiene la direccion del registro a escribir
;                   - VALOR, contiene el dato a escribir en el registro
;         salida: ninguna
outport
          PAGE0
          ;****************************
          bcf                 STATUS,0            ;limpiar el bit de acarreo
          rlf                 REGISTRO,W
          ;****************************
;         movf      REGISTRO,W
          movwf     ADDR                        ;poner la direccion del registro en los pines de direccion

          PAGE1
          clrf      DATOS                       ;set_tris_d(0x00)direccionar el PORTD como salida
          PAGE0
          movf      VALOR,W
          movwf     DATOS                       ;VALOR = DATOS=PORTD Poner el dato en el puerto D
          bcf                 CONTROL_RTL,IOW     ;Activar la peticion de escritura
          NOP
          NOP
          bsf                 CONTROL_RTL,IOW     ;Desactivar la peticion de escritura
          movlw     0xFF
          PAGE1
          movwf     DATOS                       ;direccionar el PORTD como entrada
          PAGE0
          return
;*******************************************************************************
;remote_dma-setup: configura el DMA de forma remota
;         entradas:- OPERACION, accion a realizarze, escritura o lectura
;                           - DIRECCION0, 8 bits LSB de direccion
;                           - DIRECCION1, 8 bits MSB de direccion
remote_dma_setup

          ;abortar
          movlw     CR
          movwf     REGISTRO
          movlw     0x22
          movwf     VALOR
          call      outport

          movlw     RBCR1
          movwf     REGISTRO
          movlw     0x0F
          movwf     VALOR
          call      outport

;  if(OPERACION=WRITE) outport(CR, 0x12);
;  else    outport(CR, 0x0A);

          movf      OPERACION,W
          andlw     ESCRITURA
          btfsc     STATUS,Z
          goto      saltar_lectura

          ;establecemos numero maximo de bytes lectura/escritura
          movlw     RBCR0
          movwf     REGISTRO
          movlw     0xFF
          movwf     VALOR
          call      outport
```

```
          movlw     RBCR1
          movwf     REGISTRO
          movlw     0xFF
          movwf     VALOR
          call      outport

;AQUI CAMBIA EL VALOR DEL REGISTRO BNDRY

          movlw     RSAR0                       ;enviamos los bits LSB
          movwf     REGISTRO
          movf      DIRECCION0,W
          movwf     VALOR
          call      outport

          movlw     RSAR1                       ;enviamos los bits MSB
          movwf     REGISTRO
          movf      DIRECCION1,W
          movwf     VALOR
          call      outport

;         outport(CR, 0x12) escritura
          movlw     CR
          movwf     REGISTRO
          movlw     0x12
          movwf     VALOR
          call      outport
          goto salirDMA
saltar_lectura
;         outport(CR, 0x0A)lectura
          movlw     CR
          movwf     REGISTRO
          movlw     0x1A
          movwf     VALOR
          call      outport
salirDMA
          return
;*************************************************************************************************

;*************************************************************************************************
;censar_nic: censa el estado de la NIC,, si se ha recibido algun paquete.

censar_nic

;***********************
          movlw     ISR
          movwf     REGISTRO
          movlw     0xFF                        ;Limpiamos el registro de estado de interupciones
          movwf     VALOR
          call      outport                     ;outport(ISR, 0xFF)
;***********************
          movlw     BNDRY
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          bsf                 STATUS,RP0                  ;PAGINA 1
          movwf     OUT
          bcf                 STATUS,RP0                  ;PAGINA 0

          ;outport(CR, 0x62) Page 1
          movlw     CR
          movwf     REGISTRO
          movlw     0x62
          movwf     VALOR
          call      outport

          ;in = inport(CURR) Get the current input pointer
          movlw     CURR
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          bsf                 STATUS,RP0                  ;PAGINA 1
          movwf     IN
          bcf                 STATUS,RP0                  ;PAGINA 0

          ;outport(CR, 0x22) Page 0
          movlw     CR
          movwf     REGISTRO
          movlw     0x22
          movwf     VALOR
          call      outport

          ;if(in != out)
          bsf                 STATUS,RP0                  ;PAGINA 1
          movf      IN,W
          subwf     OUT,W
          bcf                 STATUS,RP0                  ;PAGINA 0
          btfsc     STATUS,Z
          goto      salirCN             ;salir si son distintos
   ;Hay datos  presentes, leerlos entonces.
          ;call     leer_paquete

leer_paquete:
```

```
          ;remote_dma_setup(READ, (long int)out<<8)
          movlw     LECTURA
          movwf     OPERACION
          bsf               STATUS,RP0                  ;PAGINA 1
          movf      OUT,W
          bcf               STATUS,RP0                  ;PAGINA 0
          movwf     DIRECCION1
          clrf      DIRECCION0
          call      remote_dma_setup

          ;estado = inport(NIC_DATA)
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
;         movwf     ESTADO

    ;next_ptr = inport(NIC_DATA)
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          bsf               STATUS,RP0                  ;PAGINA 1
          movwf     NEXT_PTR
          bcf               STATUS,RP0                  ;PAGINA 0

          ;Clear out the byte count and dest address
    ;for (i=0 ; i<8 ; i++)
    ;inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     8
          movwf     CONTADOR
lazoCN1

          call      inport
          decfsz    CONTADOR,F
          goto      lazoCN1

          ;Retrieve source hardware address
   ;for (i=0 ; i<6 ; i++)
   ;source_address[i] = inport(NIC_DATA);

          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     source_address0
          call      inport
          movf      VALOR,W
          movwf     source_address1
          call      inport
          movf      VALOR,W
          movwf     source_address2
          call      inport
          movf      VALOR,W
          movwf     source_address3
          call      inport
          movf      VALOR,W
          movwf     source_address4
          call      inport
          movf      VALOR,W
          movwf     source_address5

          ;Retrieve ethernet message type                                      TYPE1,TYPE0
    ;type = (long int)inport(NIC_DATA)<<8;       MSB  ,LSB
    ;type = type + inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     TYPE1                       ;primero se lee el byte MSB segun la trama
          call      inport
          movf      VALOR,W
          movwf     TYPE0                       ;segundo se lee el byte LSB

          ;switch(type)
    ;{
caseARP:
          movlw     ARP0
          subwf     TYPE0,W
          btfss     STATUS,Z
          goto      caseIP                      ;si no es ARP revisar si es IP

          movlw     ARP1
          subwf     TYPE1,W
          btfss     STATUS,Z
          goto      caseIP                      ;si no es ARP revisar si es IP


          ;CASE ARP
          ;discard unneeded bytes (address space, protocol address space,
    ;length of hardware address, length of protocol address, opcode)
```

```
          ;for (i=0; i<8 ; i++)
    ;inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     8
          movwf     CONTADOR
lazoCN2

          call      inport
          decfsz    CONTADOR,F
          goto      lazoCN2

          ;discard source hardware address (we already have it)
          ;for (i=0; i<6 ; i++)
    ;inport(NIC_DATA);
          movlw     6
          movwf     CONTADOR
lazoCN3

          call      inport
          decfsz    CONTADOR,F
          goto      lazoCN3

          ;get ip address of packet source IP0:IP1:IP2:IP3
          ;for (i=0; i<4 ; i++)
    ;source_ip[i] = inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO

          call      inport
          movf      VALOR,W
          movwf     source_ip0

          call      inport
          movf      VALOR,W
          movwf     source_ip1

          call      inport
          movf      VALOR,W
          movwf     source_ip2

          call      inport
          movf      VALOR,W
          movwf     source_ip3

          ;discard target hardware address (probably not valid anyway)
          ;for (i=0; i<6 ; i++)
    ;inport(NIC_DATA);
          movlw     6
          movwf     CONTADOR
lazoCN4

          call      inport
          decfsz    CONTADOR,F
          goto      lazoCN4

          ;get ip address of destination  IP0:IP1:IP2:IP3
          ;for (i=0; i<4 ; i++)
    ;dest_ip[i] = inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO

          call      inport
          movf      VALOR,W
          movwf     dest_ip0

          call      inport
          movf      VALOR,W
          movwf     dest_ip1

          call      inport
          movf      VALOR,W
          movwf     dest_ip2

          call      inport
          movf      VALOR,W
          movwf     dest_ip3

          ;if (dest_ip[0] == my_ip[0] && dest_ip[1] == my_ip[1] &&
    ;         dest_ip[2] == my_ip[2] && dest_ip[3] == my_ip[3])
          movf      dest_ip0,W
          subwf     my_ip0,W
          btfss     STATUS,Z
          goto      salirBREAK

          movf      dest_ip1,W
          subwf     my_ip1,W
          btfss     STATUS,Z
          goto      salirBREAK

          movf      dest_ip2,W
          subwf     my_ip2,W
          btfss     STATUS,Z
```

```
          goto      salirBREAK

          movf      dest_ip3,W
          subwf     my_ip3,W
          btfss     STATUS,Z
          goto      salirBREAK

          call      arp_response        ;The ARP was for us, so let's respond
salirBREAK          ;BREAK
          goto      salirSWITCHtype

caseIP:
          movlw     IP0
          subwf     TYPE0,W
          btfss     STATUS,Z
          goto      salirSWITCHtype1    ;si no es IP entonces salir

          movlw     IP1
          subwf     TYPE1,W
          btfss     STATUS,Z
          goto      salirSWITCHtype1    ;si no es IP entonces salir



          ;CASE IP  Tratamiento del paquete IP
          ;*****Recibimos paquete IP
          ;version = inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          ;movwf    version
          ;hdr_len = (version & 0x0F) << 2;
    ;opt_len = hdr_len - 20;
          andlw     0x0F
          movwf     opt_len
          bcf               STATUS,C
          RLF               opt_len,F           ; LA ROTACION INCLUYE EL ACARREO
          RLF               opt_len,F
          movf      opt_len,W
          bsf               STATUS,RP0
          movwf     hdr_len
          bcf               STATUS,RP0
          movlw     0x14                        ;20 decimal
          subwf     opt_len,F           ;le restamos 20

          ;discard type of service
          ;inport(NIC_DATA);
          call      inport

          ;IP packet length                                                        IPL1,IPL0
          ;IP_length = inport(NIC_DATA) << 8; MSB , LSB
          call      inport
          movf      VALOR,W
          movwf     IP_length1

          call      inport
          movf      VALOR,W
          movwf     IP_length0

          ;Identification
     ;identification = inport(NIC_DATA) << 8;
     ;identification += inport(NIC_DATA);
          call      inport
          movf      VALOR,W
          movwf     identification1

          call      inport
          movf      VALOR,W
          movwf     identification0

          ;Fragment
          ;fragment = inport(NIC_DATA) << 8;
     ;fragment += inport(NIC_DATA);
          call      inport
          movf      VALOR,W
          movwf     fragment1

          call      inport
          movf      VALOR,W
          movwf     fragment0

          ;dicard time to live (the packets already here)
          ;inport(NIC_DATA);
          call      inport

          ;IP Protocol
          ;IP_protocol = inport(NIC_DATA);
          call      inport
          movf      VALOR,W
          movwf     IP_protocol

          ;get header checksum
```

```
          ;chksum = inport(NIC_DATA)<<8;
    ;chksum += inport(NIC_DATA);
          call      inport
          movf      VALOR,W
          movwf     chksum1
          call      inport
          movf      VALOR,W
          movwf     chksum0

          ;get ip address of packet source                IP0:IP2:IP2:IP3
          ;for (i=0; i<4 ; i++)
    ;source_ip[i] = inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     source_ip0
          call      inport
          movf      VALOR,W
          movwf     source_ip1
          call      inport
          movf      VALOR,W
          movwf     source_ip2
          call      inport
          movf      VALOR,W
          movwf     source_ip3

          ;get ip address of destination                  IP0:IP2:IP2:IP3
          ;for (i=0; i<4 ; i++)
    ;dest_ip[i] = inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     dest_ip0
          call      inport
          movf      VALOR,W
          movwf     dest_ip1
          call      inport
          movf      VALOR,W
          movwf     dest_ip2
          call      inport
          movf      VALOR,W
          movwf     dest_ip3

          movlw     0x00
          subwf     opt_len,W
          btfsc     STATUS,Z
          goto      no_opciones_ip

          ;discard options if any
          ;for (i=0; i<opt_len ; i++)
    ;inport(NIC_DATA);
          movf      opt_len,W
          movwf     CONTADOR
lazoCN5
          call      inport
          decfsz    CONTADOR,F
          goto      lazoCN5


no_opciones_ip
          ;switch (IP_protocol)
          ;caseICMP
          movlw     ICMP
          subwf     IP_protocol,W
          btfss     STATUS,Z
          goto      caseTCP
          call      case_ICMP
          goto salirSWITCHip_protocol

caseTCP
          movlw     TCP
          subwf     IP_protocol,W
          btfss     STATUS,Z
          goto      caseUDP
          ;case TCP
          call      case_TCP
          goto      salirSWITCHip_protocol

caseUDP
          movlw     UDP
          subwf     IP_protocol,W
          btfss     STATUS,Z
          goto      salirSWITCHip_protocol
          ;aqui codigo para paquete UDP
salirSWITCHip_protocol


          goto      salirSWITCHtype
salirSWITCHtype1

salirSWITCHtype
```

```
          ;outport(CR, 0x22);
          movlw     CR
          movwf     REGISTRO
          movlw     0x22
          movwf     VALOR
          call      outport

          ;return




          ;outport(BNDRY, next_ptr);
          movlw     BNDRY
          movwf     REGISTRO
          bsf                 STATUS,RP0                          ;PAGINA 1
          movf      NEXT_PTR,W
          bcf                 STATUS,RP0                          ;PAGINA 0
          movwf     VALOR
          call      outport

;*******ESTO AGREGADO

;**********************
          movlw     BNDRY
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          bsf                 STATUS,RP0                  ;PAGINA 1
          movwf     OUT
          bcf                 STATUS,RP0                  ;PAGINA 0

          ;outport(CR, 0x62) Page 1
          movlw     CR
          movwf     REGISTRO
          movlw     0x62
          movwf     VALOR
          call      outport

          ;in = inport(CURR) Get the current input pointer
          movlw     CURR
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          bsf                 STATUS,RP0                  ;PAGINA 1
          movwf     IN
          bcf                 STATUS,RP0                  ;PAGINA 0

          ;outport(CR, 0x22) Page 0
          movlw     CR
          movwf     REGISTRO
          movlw     0x22
          movwf     VALOR
          call      outport

          ;if(in != out)
          bsf                 STATUS,RP0                  ;PAGINA 1
          movf      IN,W
          subwf     OUT,W
          bcf                 STATUS,RP0                  ;PAGINA 0
          btfsc     STATUS,Z
          goto      salirCN0            ;salir si son distintos
    ;ay datos  presentes, leerlos entonces.
          ;call     leer_paquete
          goto      leer_paquete


salirCN0

;**********************
          movlw     ISR
          movwf     REGISTRO
          movlw     0xFF                        ;Limpiamos el registro de estado de interupciones
          movwf     VALOR
          call      outport                     ;outport(ISR, 0xFF)
salirCN

          return
;*********************************************************************
;*********************************************************************


;*********************************************************************
;*********************************************************************

arp_response
          ;prepare ethernet packet
          ;load_ethernet_header()
          call      load_ethernet_header        ;MAC destino y fuente

          ;packet type                          ARP=0x0806
```

```
      ;outport(NIC_DATA, ARP >> 8)
      ;outport(NIC_DATA, ARP )
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     ARP1
      movwf     VALOR
      call      outport

      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     ARP0
      movwf     VALOR
      call      outport

      ;hardware type (0x0001 = ethernet)
      ;outport(NIC_DATA, 0x00);
      ;outport(NIC_DATA, 0x01);
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x00
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x01
      movwf     VALOR
      call      outport

      ;protocol type (0x0800 = IP)
      ;outport(NIC_DATA, 0x08);
      ;outport(NIC_DATA, 0x00);
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x08
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x00
      movwf     VALOR
      call      outport

      ;hardware address length (6 bytes)
      ;outport(NIC_DATA, 0x06);
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x06
      movwf     VALOR
      call      outport

      ;protocol address length (4 bytes)
      ;outport(NIC_DATA, 0x04);
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x04
      movwf     VALOR
      call      outport

      ;opcode (ARP reply)
      ;outport(NIC_DATA, 0x00);
      ;outport(NIC_DATA, 0x02);
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x00
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x02
      movwf     VALOR
      call      outport

      ;my hardware address
      ;for (i=0 ; i<6 ; i++)
;outport(NIC_DATA, physical_address[i]);
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      physical_address0,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      physical_address1,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      physical_address2,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      physical_address3,W
```

```
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      physical_address4,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      physical_address5,W
      movwf     VALOR
      call      outport

      ;my ip address
      ;for (i=0 ; i<4 ; i++)
;outport(NIC_DATA, my_ip[i]);
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      my_ip0,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      my_ip1,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      my_ip2,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      my_ip3,W
      movwf     VALOR
      call      outport

      ;target hardware address
      ;for (i=0 ; i<6 ; i++)
;outport(NIC_DATA, source_address[i]);
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_address0,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_address1,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_address2,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_address3,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_address4,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_address5,W
      movwf     VALOR
      call      outport

      ;source ip address
      ;for (i=0 ; i<4 ; i++)
;outport(NIC_DATA, source_ip[i])
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_ip0,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_ip1,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_ip2,W
      movwf     VALOR
      call      outport
      movlw     NIC_DATA
      movwf     REGISTRO
      movf      source_ip3,W
      movwf     VALOR
```

```
          call      outport

          ;pad to 46 bytes (46-28=18)
          ;for (i=0 ; i<18 ;i++)
    ;outport(NIC_DATA, 0x00)
          movlw     18
          movwf     CONTADOR

lazoARP_RESPONSE1

          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     0x00
          movwf     VALOR
          call      outport

          decfsz    CONTADOR,F
          goto      lazoARP_RESPONSE1

          ;packet assmebled so let's send it
          ;send_packet(60=3C);
          bsf       STATUS,RP0
          movlw     0
          movwf     LEN1
          movlw     0x3C
          movwf     LEN0
          bcf                 STATUS,RP0
          call      send_packet
          return
;*****************************************************************
;*************************************************************************************************
load_ethernet_header:

          ; (WRITE, XMT_BUF_START << 8)
          movlw     ESCRITURA
          movwf     OPERACION
          movlw     XMT_BUF_START
          movwf     DIRECCION1
          movlw     0x00
          movwf     DIRECCION0
          call      remote_dma_setup

          ;write hardware addresses
          ;for(i=0 ; i<6 ; i++)
    ;outport(NIC_DATA, source_address[i]);
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      source_address0,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      source_address1,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      source_address2,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      source_address3,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      source_address4,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      source_address5,W
          movwf     VALOR
          call      outport

          ;for(i=0 ; i<6 ; i++)
    ;outport(NIC_DATA, physical_address[i]);
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      physical_address0,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      physical_address1,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      physical_address2,W
          movwf     VALOR
          call      outport
```

```
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      physical_address3,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      physical_address4,W
          movwf     VALOR
          call      outport
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      physical_address5,W
          movwf     VALOR
          call      outport

          return
;*************************************************
;*****************************************************************************************
;send_packet:
;                   entradas: LEN1:LEN0
send_packet

          ;outport(CR, 0x22);
          movlw     CR
          movwf     REGISTRO
          movlw     0x22
          movwf     VALOR
          call      outport

          ;outport(TBCR0, len );
          movlw     TBCR0
          movwf     REGISTRO
          bsf                 STATUS,RP0
          movf      LEN0,W
          bcf                 STATUS,RP0
          movwf     VALOR
          call      outport

          ;outport(TBCR1, len >> 8);
          movlw     TBCR1
          movwf     REGISTRO
          bsf                 STATUS,RP0
          movf      LEN1,W
          bcf                 STATUS,RP0
          movwf     VALOR
          call      outport

          ;outport(TPSR, XMT_BUF_START);
          movlw     TPSR
          movwf     REGISTRO
          movlw     XMT_BUF_START
          movwf     VALOR
          call      outport

          ;outport(CR, 0x26);
          movlw     CR
          movwf     REGISTRO
          movlw     0x26
          movwf     VALOR
          call      outport
          return
;*****************************************************************************
case_ICMP
          ;protocolo_icmp=inport(NIC_DATA)
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          bsf                 STATUS,RP0                          ;banco 1
          movwf     protocolo_icmp
          bcf                 STATUS,RP0                          ;banco 0



          ;if(protocolo_icmp==ECHO)
          movlw     ECHO
          bsf                 STATUS,RP0                          ;banco 1
          subwf     protocolo_icmp,W
          btfss     STATUS,Z
          goto      no_echo
si_echo
          bcf                 STATUS,RP0                          ;banco 0
          ;esto es un ping
          ;recibir el mensaje datos_icmp=inport(NIC_DATA)
          call      inport
          movf      VALOR,W
          bsf                 STATUS,RP0                          ;banco 1
          movwf     datos_icmp
          bcf                 STATUS,RP0                          ;banco 0
          ; recibir el cheksum de cabecera ICMP
          ;chksum_icmp=inport(NIC_DATA)
          call      inport
```

```
      movf      VALOR,W
      bsf               STATUS,RP0                          ;banco 1
      movwf     chksum_icmp1
      bcf               STATUS,RP0                          ;banco 0
      call      inport
      movf      VALOR,W
      bsf               STATUS,RP0                          ;banco 1
      movwf     chksum_icmp0
      bcf               STATUS,RP0                          ;banco 0
      ; recibir el identificador de cabecera ICMP
      ;identificador_icmp=inport(NIC_DATA)
      call      inport
      movf      VALOR,W
      bsf               STATUS,RP0                          ;banco 1
      movwf     identificador_icmp1
      bcf               STATUS,RP0                          ;banco 0
      call      inport
      movf      VALOR,W
      bsf               STATUS,RP0                          ;banco 1
      movwf     identificador_icmp0
      bcf               STATUS,RP0                          ;banco 0
      ; recibir secuencia de cabecera ICMP
      ;secuencia_icmp=inport(NIC_DATA)
      call      inport
      movf      VALOR,W
      bsf               STATUS,RP0                          ;banco 1
      movwf     secuencia_icmp1
      bcf               STATUS,RP0                          ;banco 0
      call      inport
      movf      VALOR,W
      bsf               STATUS,RP0                          ;banco 1
      movwf     secuencia_icmp0
      bcf               STATUS,RP0                          ;banco 0

      ;si IP destino == my_ip
      movf      dest_ip0,W
      subwf     my_ip0,W
      btfss     STATUS,Z
      goto      no_echo

      movf      dest_ip1,W
      subwf     my_ip1,W
      btfss     STATUS,Z
      goto      no_echo

      movf      dest_ip2,W
      subwf     my_ip2,W
      btfss     STATUS,Z
      goto      no_echo

      movf      dest_ip3,W
      subwf     my_ip3,W
      btfss     STATUS,Z
      goto      no_echo

      ;****respuesta al ping*******

      ;preparar cabecera ethernet
      call      load_ethernet_header          ;cargar MAC destino y fuente
      ;load_IP_header(28,ICMP) Number of bytes included in call (add 2 for protocol)
      ;20 for IP header
;8 for Options
      bsf               STATUS,RP0
      clrf      IP_packet_length1
      movlw     0x1c      ;28 decimal
      movwf     IP_packet_length0
      movlw     ICMP
      movwf     IP_send_protocol
      bcf               STATUS,RP0
      bcf               PCLATH,4
      bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA
      call      load_IP_header
      bcf               PCLATH,4
      bcf               PCLATH,3            ;PAGINA 0 DE PROGRAMA
      ;chksum=0
      bcf               STATUS,RP0                          ;banco 0
      clrf      chksum0
      clrf      chksum1
      ;outport(NICDATA,datos_icmp)
      movlw     NIC_DATA
      movwf     REGISTRO
      movlw     0x00
      movwf     VALOR
      call      outport
      bsf               STATUS,RP0                          ;banco 1
      movf      datos_icmp,W
      bcf               STATUS,RP0                          ;banco 0
      movwf     VALOR
      call      outport
      ;prev_chksum=chksum
      movf      chksum0,W
      movwf     prev_chksum0
      movf      chksum1,W
```

```
          movwf     prev_chksum1
          ;chksum += identificador_icmp
          bsf                 STATUS,RP0                              ;banco 1
          movf      identificador_icmp0,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     numero1_0
          bsf                 STATUS,RP0                              ;banco 1
          movf      identificador_icmp1,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     numero1_1           ;numero1 = TCP
          movf      chksum0,W
          movwf     numero2_0
          movf      chksum1,W
          movwf     numero2_1           ;numero2 = chksum
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      calcular_check
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          ;prev_chksum=chksum
          movf      chksum0,W
          movwf     prev_chksum0
          movf      chksum1,W
          movwf     prev_chksum1
          ;chksum += secuencia_icmp
          bsf                 STATUS,RP0                              ;banco 1
          movf      secuencia_icmp0,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     numero1_0
          bsf                 STATUS,RP0                              ;banco 1
          movf      secuencia_icmp1,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     numero1_1           ;numero1 = TCP
          movf      chksum0,W
          movwf     numero2_0
          movf      chksum1,W
          movwf     numero2_1           ;numero2 = chksum
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      calcular_check
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          ;outportw(NIC_DATA, ~chksum);
          comf      chksum0,F
          comf      chksum1,F
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      chksum1,W
          movwf     VALOR
          call      outport
          movf      chksum0,W
          movwf     VALOR
          call      outport
          ;outport(NICDATA,identificador_icmp)
          movlw     NIC_DATA
          movwf     REGISTRO
          bsf                 STATUS,RP0                              ;banco 1
          movf      identificador_icmp1,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     VALOR
          call      outport
          bsf                 STATUS,RP0                              ;banco 1
          movf      identificador_icmp0,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     VALOR
          call      outport
          ;outport(NICDATA,secuencia_icmp)
          movlw     NIC_DATA
          movwf     REGISTRO
          bsf                 STATUS,RP0                              ;banco 1
          movf      secuencia_icmp1,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     VALOR
          call      outport
          bsf                 STATUS,RP0                              ;banco 1
          movf      secuencia_icmp0,W
          bcf                 STATUS,RP0                              ;banco 0
          movwf     VALOR
          call      outport
          ;for(i=0;i<18;i++)
          ;         outport(NICDATA,0x00
          movlw     18
          movwf     CONTADOR
lazo_respuesta_ping
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     0x00
          movwf     VALOR
          call      outport
          decfsz    CONTADOR,F
          goto      lazo_respuesta_ping
          ;enviar paquete ensamblado
          ;send_packet(60=3C);
```

```
        bsf     STATUS,RP0
        movlw   0
        movwf   LEN1
        movlw   0x3C
        movwf   LEN0
        bcf             STATUS,RP0
        call    send_packet

no_echo
        bcf             STATUS,RP0                          ;banco 0
        return
;*********************************************************************************
case_TCP


        ;retrieve 'source port'                          port1,port0
    ;tcp_source_port = inport(NIC_DATA)<<8
    ;tcp_source_port += inport(NIC_DATA)
        movlw   NIC_DATA
        movwf   REGISTRO
        call    inport
        movf    VALOR,W
        movwf   tcp_source_port1
        call    inport
        movf    VALOR,W
        movwf   tcp_source_port0

        ;retrieve 'destination port'           port1,port0
    ;tcp_dest_port = inport(NIC_DATA)<<8
    ;tcp_dest_port += inport(NIC_DATA)
        movlw   NIC_DATA
        movwf   REGISTRO
        call    inport
        movf    VALOR,W
        movwf   tcp_dest_port1
        call    inport
        movf    VALOR,W
        movwf   tcp_dest_port0

        ;We only respond to port 80 (HTML requests)
    ;if (tcp_dest_port != 80 = 0x50) break;
        bsf             STATUS,RP0                          ;banco 1
        movf    my_port0,W
        bcf             STATUS,RP0                          ;banco 0
        subwf   tcp_dest_port1,W
        btfss   STATUS,Z
        goto    salirCaseTcp1
        bsf             STATUS,RP0                          ;banco 1
        movf    my_port1,W
        bcf             STATUS,RP0                          ;banco 0
        subwf   tcp_dest_port0,W
        btfss   STATUS,Z
        goto    salirCaseTcp1
        ;si es port 80 entonces hacer esto

        ;retrieve sequence number
    ;seq[0]=inport(NIC_DATA)<<8;
    ;seq[1]+=inport(NIC_DATA);
    ;seq[2]=inport(NIC_DATA)<<8;
    ;seq[3]+=inport(NIC_DATA);
        movlw   NIC_DATA
        movwf   REGISTRO
        call    inport
        movf    VALOR,W
        movwf   seq0
        call    inport
        movf    VALOR,W
        movwf   seq1
        call    inport
        movf    VALOR,W
        movwf   seq2
        call    inport
        movf    VALOR,W
        movwf   seq3

        ;retrieve acknowlegment number
    ;ack[0]=inport(NIC_DATA)<<8;
    ;ack[1]+=inport(NIC_DATA);
    ;ack[2]=inport(NIC_DATA)<<8;
    ;ack[3]+=inport(NIC_DATA);
        movlw   NIC_DATA
        movwf   REGISTRO
        call    inport
        movf    VALOR,W
        movwf   ack0
        call    inport
        movf    VALOR,W
        movwf   ack1
        call    inport
        movf    VALOR,W
        movwf   ack2
        call    inport
        movf    VALOR,W
```

```
          movwf     ack3

          ;retrieve offset
    ;offset=(inport(NIC_DATA) & 0xF0) >> 2;
    ;tcp_options = (offset - 20);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          andlw     0xF0
          bsf                 STATUS,RP0
          movwf     offset
          bcf                 STATUS,C
          rrf                 offset,F
          rrf                 offset,F
          movf      offset,W
          bcf                 STATUS,RP0
          movwf     tcp_options
          movlw     0x14                                ;20 DECIMAL
          subwf     tcp_options,F


          ;retrieve flags
    ;flags=inport(NIC_DATA) & 0x3F;
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          andlw     0x3F
          movwf     flags


          ;if ((flags & TCP_SYN) == TCP_SYN) max_seg=0;
          movlw     TCP_SYN
          andwf     flags,W
          btfsc     STATUS,Z
          goto      continuarTCP1
          clrf      max_seg0
          clrf      max_seg1
continuarTCP1
          ;retrieve windows
    ;window=inport(NIC_DATA) << 8;
    ;window+=inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     window1
          call      inport
          movf      VALOR,W
          movwf     window0

          ;discard checksum
    ;inport(NIC_DATA);
    ;inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          call      inport

          ;discard urgent pointer
    ;inport(NIC_DATA);
    ;inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          call      inport

          ;Hay opciones TCP
          movlw     0x00
          subwf     tcp_options,W
          btfsc     STATUS,Z
          goto      no_opciones_tcp

          ;Handle options
    ;for (i=0 ; i < tcp_options ; i++)
          clrf      CONTADOR
lazoTCP1
          ;kind = inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     kind

          ;switch(kind)
          movlw     2
          subwf     kind,W
          btfss     STATUS,Z
          goto      caseother
    ;case 2:
case2:
          ;inport(NIC_DATA);
```

```
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport    ;logitud de las opciones
    ;if ((flags & TCP_SYN) == TCP_SYN) {
          movlw     TCP_SYN
          andwf     flags,W
          btfsc     STATUS,Z
          goto      elseTCP1
          ;max_seg = inport(NIC_DATA) << 8;
   ;max_seg += inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     max_seg1
          call      inport
          movf      VALOR,W
          movwf     max_seg0
          goto      salirSWITCHkind
elseTCP1
    ;inport(NIC_DATA);
    ;inport(NIC_DATA);
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          call      inport
    goto  salirSWITCHkind
caseother
          ;otros casos
salirSWITCHkind
          movlw     0x04
          addwf     CONTADOR,F
          movf      CONTADOR,W
          subwf     tcp_options,W
          btfss     STATUS,Z
          goto      lazoTCP1
no_opciones_tcp

;*** RESPUESTA TCP
          ;switch(html_socket)
          ;case LISTEN:
          movlw     LISTEN
          bsf               STATUS,RP0
          subwf     html_socket,W
          bcf               STATUS,RP0
          btfss     STATUS,Z
          goto      caseSYNRCVD
caseLISTEN:
          call      case_LISTEN

          goto      salirSWITCH_html_socket      ;break

caseSYNRCVD:
          movlw     SYN_RCVD
          bsf               STATUS,RP0
          subwf     html_socket,W
          bcf               STATUS,RP0
          btfss     STATUS,Z
          goto      caseESTAB
          call      case_RCVD
          goto      salirSWITCH_html_socket

caseESTAB:
          movlw     ESTAB
          bsf               STATUS,RP0
          subwf     html_socket,W
          bcf               STATUS,RP0
          btfss     STATUS,Z
          goto      salirSWITCH_html_socket
          case_ESTAB:

;We are expecting an ACK without or SYN,We could receive a FIN
          ;if (flags & TCP_ACK)
          movlw     TCP_ACK
          andwf     flags,W
          btfsc     STATUS,Z
          goto      salir_CE
          ;si es ACK hacer esto

;**** revisamos si es

          ;if(xm_ack[0] != seq[0])
          movf      xm_ack0,W
          subwf     seq0,W
          btfss     STATUS,Z
          goto      salir_CE           ;break

          movf      xm_ack1,W
          subwf     seq1,W
          btfss     STATUS,Z
          goto      salir_CE           ;break
```

```
          ;if(xm_ack[1] != seq[1])
          movf      xm_ack2,W
          subwf     seq2,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          movf      xm_ack3,W
          subwf     seq3,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          ;if(ack[0] != xm_seq[0])
          movf      ack0,W
          subwf     xm_seq0,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          movf      ack1,W
          subwf     xm_seq1,W
          btfss     STATUS,Z
          goto      salir_CE            ;break


          ;if(ack[1] != xm_seq[1])
          movf      ack2,W
          subwf     xm_seq2,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          movf      ack3,W
          subwf     xm_seq3,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          ;Check if addressed for this socket
          ;if (port!=tcp_source_port) break
          movf      port1,W
          subwf     tcp_source_port1,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          movf      port0,W
          subwf     tcp_source_port0,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          ;if(ip[0]!=source_ip[0])
          movf      ip0,W
          subwf     source_ip0,W
          btfss     STATUS,Z
          goto      salir_CE            ;break


          ;if(ip[1]!=source_ip[1])
          movf      ip1,W
          subwf     source_ip1,W
          btfss     STATUS,Z
          goto      salir_CE            ;break


          ;if(ip[2]!=source_ip[2])
          movf      ip2,W
          subwf     source_ip2,W
          btfss     STATUS,Z
          goto      salir_CE            ;break


;if(ip[3]!=source_ip[3])
          movf      ip3,W
          subwf     source_ip3,W
          btfss     STATUS,Z
          goto      salir_CE            ;break

          ;tcp_estab();
          ;******************** SOCKET TCP ESTABLECIDO
          tcp_estab
;         call      BANDERA0

;Calculate data length
          ;tcp_data_len = IP_length - (hdr_len + offset);
          bsf                 STATUS,RP0
          movf      offset,W
          addwf     hdr_len,W           ;hdr_len + offset
          bcf                 STATUS,RP0
          movwf     numero2_0
          clrf      numero2_1           ;numero2=(hdr_len + offset)
          movf      IP_length0,W
          movwf     numero1_0
          movf      IP_length1,W
          movwf     numero1_1
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      restar                        ; restar numero1 - numero2=resultado
```

```
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        movf    resultado0,W
        bsf             STATUS,RP0
        movwf   tcp_data_len0
        bcf             STATUS,RP0
        movf    resultado1,W
        bsf             STATUS,RP0
        movwf   tcp_data_len1
        bcf             STATUS,RP0

        ;Increment sequence to account for data
        ;xm_ack[1] += tcp_data_len
        bsf             STATUS,RP0
        movf    tcp_data_len0,W
        bcf             STATUS,RP0
        movwf   numero1_0
        bsf             STATUS,RP0
        movf    tcp_data_len1,W
        bcf             STATUS,RP0
        movwf   numero1_1
        movf    xm_ack2,W
        movwf   numero2_1
        movf    xm_ack3,W
        movwf   numero2_0           ;numero2=
        bcf             PCLATH,4
        BSF             PCLATH,3            ;PAGINA 1 DE PROGRAMA
        call    sumar                       ; restar numero1 - numero2=resultado
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        movf    resultado0,W
        movwf   xm_ack3
        movf    resultado1,W
        movwf   xm_ack2

        ;if (xm_ack[1] < seq[1] )xm_ack[0]++
        ;restar xm_ack[1] - seq[1], si da positiva la resta es mayor
        ; si da negativa es menor
        movf    xm_ack2,W
        movwf   numero1_1
        movf    xm_ack3,W
        movwf   numero1_0           ;numero1=xm_ack[1]
        movf    seq2,W
        movwf   numero2_1
        movf    seq3,W
        movwf   numero2_0           ;numero2=seq[1]
        bcf             PCLATH,4
        BSF             PCLATH,3            ;PAGINA 1 DE PROGRAMA
        call    restar                      ; restar numero1 - numero2=resultado
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        btfsc   POSITIVA,0
        goto    continuarTCPE1
        ;xm_ack[0]++
        movf    xm_ack0,W
        movwf   numero1_1
        movf    xm_ack1,W
        movwf   numero1_0           ;numero1=xm_ack[0]
        movlw   0x00
        movwf   numero2_1
        movlw   0x01
        movwf   numero2_0           ;numero2=1
        bcf             PCLATH,4
        BSF             PCLATH,3            ;PAGINA 1 DE PROGRAMA
        call    sumar                       ; restar numero1 - numero2=resultado
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0 DE PROGRAMA
        movf    resultado0,W
        movwf   xm_ack1
        movf    resultado1,W
        movwf   xm_ack0
continuarTCPE1


;*********          ALMACENAREMOS LOS DATOS EN EL BUFFER DE ENTRADA*******
; revisamos si hay datos
        bsf             STATUS,RP0                          ;BANK 1
        movf    tcp_data_len1,W
        btfss   STATUS,Z
        goto    hay_datos1                          ;saltar a tratar los datos
        movf    tcp_data_len0,W
        btfsc   STATUS,Z
        goto    cero_datosFIN               ;si no hay datos es un ACK o FIN
        goto    hay_datos1                          ;saltar a tratar los datos

cero_datosFIN
        ;revisar si es un TCP_FIN
        bcf             STATUS,RP0                          ;BANK 0
        movlw   TCP_FIN
        andwf   flags,W
        btfsc   STATUS,Z
        goto    cero_datosACK
        goto    enviarFIN
```

```
cero_datosACK

          movlw     CLOSING
          bsf               STATUS,RP0                          ;BANK 1
          subwf     html_socket2,W
          bcf               STATUS,RP0                          ;BANK 0
          btfss     STATUS,Z
          goto      cero_datos1
          movlw     LISTEN
          bsf               STATUS,RP0                          ;BANK 1
          movwf     html_socket
          movwf     html_socket2
          bcf               STATUS,RP0                          ;BANK 0

          goto      enviarACK

hay_datos1
          bcf               STATUS,RP0                          ;BANK 0

          ;si hay datos almacenarlos entonces
          ;CONTADOR1:CONTADOR2 = numero de bytes a leer
          ;MSB:LSB
          bsf               STATUS,RP0                          ;BANK 1
          movf      tcp_data_len1,W
          bcf               STATUS,RP0
          movwf     CONTADOR1
          bsf               STATUS,RP0
          movf      tcp_data_len0,W
          bcf               STATUS,RP0                          ;BANK 0
          movwf     CONTADOR2


          ;Apuntamos al inicio de buffer de entrada
almacenar_datos
          movlw     NIC_DATA
          movwf     REGISTRO
          call      inport
          movf      VALOR,W
          movwf     INDF
          incf      FSR,F
          ;decrementamos contador de bytes
          movlw     0x01
          subwf     CONTADOR2,f
          btfsc     STATUS,Z
          goto      revisar_cero1
          btfsc     STATUS,C
          goto      almacenar_datos
          decfsz    CONTADOR1,f
          btfss     STATUS,Z
          goto      almacenar_datos
revisar_cero1
          movf      CONTADOR1,W
          btfss     STATUS,Z
          goto      almacenar_datos


;******** TERMINAMOS DE ALMACENAR LOS DATOS EN EL BUFFER *********
;         revisamos si el ultimo dato es el fin de cadena

          bsf               STATUS,RP0
          bcf               FIN_LINEA,0                         ;indicamos por omision no fin de linea
          bcf               STATUS,RP0
          decf      FSR,F
          movf      INDF,W
          sublw     '$'
          btfss     STATUS,Z
          goto      no_fin_linea1
          bsf               STATUS,RP0
          bsf               FIN_LINEA,0                         ;indicamos fin de linea
          bcf               STATUS,RP0
          movlw     BUFFER_ENTRADA
          movwf     FSR


;si se esta estableciendo el socket guardar datos del host
          ;almacenar la direccion MAC
          bcf               STATUS,RP0                  ;banco 0
          movf      source_address0,W
          bsf               STATUS,RP0                  ;banco 1
          movwf     Bsource_address0
          bcf               STATUS,RP0                  ;banco 0

          movf      source_address1,W
          bsf               STATUS,RP0                  ;banco 1
          movwf     Bsource_address1
          bcf               STATUS,RP0                  ;banco 0

          movf      source_address2,W
          bsf               STATUS,RP0                  ;banco 1
          movwf     Bsource_address2
          bcf               STATUS,RP0                  ;banco 0
```

```
         movf      source_address3,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_address3
         bcf               STATUS,RP0                  ;banco 0

         movf      source_address4,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_address4
         bcf               STATUS,RP0                  ;banco 0

         movf      source_address5,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_address5
         bcf               STATUS,RP0                  ;banco 0

;si se esta estableciendo el socket guardar datos del host
         ;almacenar la direccion IP
         movf      source_ip0,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_ip0
         bcf               STATUS,RP0                  ;banco 0

         movf      source_ip1,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_ip1
         bcf               STATUS,RP0                  ;banco 0

         movf      source_ip2,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_ip2
         bcf               STATUS,RP0                  ;banco 0

         movf      source_ip3,W
         bsf               STATUS,RP0                  ;banco 1
         movwf     Bsource_ip3
         bcf               STATUS,RP0                  ;banco 0

         goto      no_fin_linea2
no_fin_linea1
         incf      FSR,F
no_fin_linea2

;******* SE ENVIA EL RESPECTIVO ACK    *****************************
enviarFIN
enviarACK
         ;CARGAR CABECERA ETHERNET
         call      load_ethernet_header

         ;CARGAR CABECERA IP
         ;load_IP_header(48,TCP) Number of bytes included in call (add 2 for protocol)
         ;20 for IP header
   ;20 for TCP header
   ;8 for Options
         bsf               STATUS,RP0                        ;BANK 1
         clrf      IP_packet_length1
         movlw     0x30      ;48 decimal
         movwf     IP_packet_length0
         movlw     TCP
         movwf     IP_send_protocol
         bcf               STATUS,RP0                        ;BANK 0
         bcf               PCLATH,4
         bsf               PCLATH,3                  ;PAGINA 1 DE PROGRAMA
         call      load_IP_header
         bcf               PCLATH,4
         bcf               PCLATH,3                  ;PAGINA 0 DE PROGRAMA

         ;CARGAR CABECERA TCP
         ;load_TCP_header( 0x7000 | TCP_SYN | TCP_ACK, 0x07B8, 28);
   ;Number of bytes include in load_IP_header call
         movlw     0x70
         bsf               STATUS,RP0                        ;BANK 1
         movwf     data_flags1
         movlw     0x00
         movwf     data_flags0
         bcf               data_flags0,1             ;TCP_SYN
         bsf               data_flags0,4             ;TCP_ACK

         movlw     TCP_FIN
         bcf               STATUS,RP0                        ;BANK 0
         andwf     flags,W
         bsf               STATUS,RP0                        ;BANK 1
         btfsc     STATUS,Z
         goto      no_banderaFIN

         bsf               data_flags0,0             ;TCP_FIN
         movlw     CLOSING
         movwf     html_socket2
         bcf               STATUS,RP0                        ;BANK 0

         bsf               STATUS,RP0                        ;BANK 1
```

```
          ;Increment sequence to account for data
          ;xm_ack[1] += 1
          bcf                 STATUS,RP0                              ;BANK 0
          movlw     0x01
          movwf     numero1_0
          movlw     0x00
          movwf     numero1_1
          movf      xm_ack2,W
          movwf     numero2_1
          movf      xm_ack3,W
          movwf     numero2_0                   ;numero2=
          bcf                 PCLATH,4
          BSF                 PCLATH,3                    ;PAGINA 1 DE PROGRAMA
          call      sumar                                 ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3                    ;PAGINA 0 DE PROGRAMA
          movf      resultado0,W
          movwf     xm_ack3
          movf      resultado1,W
          movwf     xm_ack2


          bsf                 STATUS,RP0                              ;BANK 1

no_banderaFIN
          movlw     0x07
          movwf     tcp_chksum1
          movlw     0xb8
          movwf     tcp_chksum0
          clrf      tcp_length1
          movlw     0x1c                                  ;28 decimal
          movwf     tcp_length0
          bcf                 STATUS,RP0                              ;BANK 0
          bcf                 PCLATH,4
          bsf                 PCLATH,3
          call      load_tcp_header
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          ;Send options
          ;outport(NIC_DATA,0x02)
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     0x02
          movwf     VALOR
          call      outport

          ;outdata(0x04);
          movlw     0x04
          movwf     VALOR
          call      outport
          ;outdata(0x05)
          movlw     0x05
          movwf     VALOR
          call      outport
          ;outdata(0xB4)
          movlw     0xb4
          movwf     VALOR
          call      outport
          ;outdata(0x00)
          movlw     0x00
          movwf     VALOR
          call      outport
          ;outdata(0x00)
          movlw     0x00
          movwf     VALOR
          call      outport
          ;outdata(0x00)
          movlw     0x00
          movwf     VALOR
          call      outport
          ;outdata(0x00)
          movlw     0x00
          movwf     VALOR
          call      outport

          ;send_packet(62)
          bsf                 STATUS,RP0
          clrf      LEN1
          movlw     0x3e                                  ;62 decimal
          movwf     LEN0
          bcf                 STATUS,RP0
          call      send_packet

          movlw     TCP_FIN
          bcf                 STATUS,RP0                              ;BANK 0
          andwf     flags,W
          bsf                 STATUS,RP0                              ;BANK 1
          btfsc     STATUS,Z
          goto      cero_datos1


          ;Calculate the next sequence number to send
          ;In this case we add 1 since we only sent a SYN
```

```
          ;xm_seq[1]++
          bcf                 STATUS,RP0                          ;BANK 0

          movf      xm_seq2,W
          movwf     numero1_1
          movf      xm_seq3,W
          movwf     numero1_0                   ;numero1=xm_ack[1]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0                   ;numero2=1
          bcf                 PCLATH,4
          BSF                 PCLATH,3                    ;PAGINA 1 DE PROGRAMA
          call      sumar                                 ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3                    ;PAGINA 0 DE PROGRAMA
          movf      resultado1,W
          movwf     xm_seq2
          movf      resultado0,W
          movwf     xm_seq3
          goto      salir123

          ;cae con un simple ACK
cero_datos1                                                 ;notificar que era un ACK el paquete recibido

salir123
          goto      salir_CE2
salir_CE
salir_CE2
salirSWITCH_html_socket

salirCaseTcp1

          return
;*****************************************************************
;*****************************************************************



;*****************************************************************
;*****************************************************************
case_LISTEN:


;if (flags & TCP_SYN)
          movlw     TCP_SYN
          andwf     flags,W
          btfsc     STATUS,Z
          goto      salir_CL            ;break
          ;si es TCP_SYN entonces hacer esto
          ;if (flags & TCP_ACK) break;
          movlw     TCP_ACK
          andwf     flags,W
          btfss     STATUS,Z
          goto      salir_CL            ;break
          ;if (flags & TCP_RST || flags & TCP_FIN) break;
          movlw     TCP_RST
          andwf     flags,W
          btfss     STATUS,Z
          goto      salir_CL            ;break

          movlw     TCP_FIN
          andwf     flags,W
          btfss     STATUS,Z
          goto      salir_CL            ;break

          ;tcp_listen()
          call      tcp_listen


salir_CL
          return
;*********************************************************************************
case_RCVD:

          ;Check if addressed for this socket
          ;if (port!=tcp_source_port) break
          movf      port1,W
          subwf     tcp_source_port1,W
          btfss     STATUS,Z
          goto      salirCR                     ;break
          movf      port0,W
          subwf     tcp_source_port0,W
          btfss     STATUS,Z
          goto      salirCR                     ;break

          ;if(ip[0]!=source_ip[0]);break
          movf      ip0,W
          subwf     source_ip0,W
          btfss     STATUS,Z
          goto      salirCR

          ;if(ip[1]!=source_ip[1]);break
```

```
          movf      ip1,W
          subwf     source_ip1,W
          btfss     STATUS,Z
          goto      salirCR

          ;if(ip[2]!=source_ip[2]);break
          movf      ip2,W
          subwf     source_ip2,W
          btfss     STATUS,Z
          goto      salirCR

;if(ip[3]!=source_ip[3]);break
          movf      ip3,W
          subwf     source_ip3,W
          btfss     STATUS,Z
          goto      salirCR



          ;We are expecting an ACK without a SYN
    ;if (flags & TCP_ACK)
          movlw     TCP_ACK
          andwf     flags,W
          btfsc     STATUS,Z
          goto      salirCR   ;si no es ACK salir
          ;si era ACK entonces hacer esto



;if (flags & TCP_SYN) break;
          movlw     TCP_SYN
          andwf     flags,W
          btfss     STATUS,Z
          goto      salirCR                     ;break

          ;if (flags & TCP_RST || flags & TCP_FIN) break;
          movlw     TCP_RST
          andwf     flags,W
          btfss     STATUS,Z
          goto      salirCR                     ;break

          movlw     TCP_FIN
          andwf     flags,W
          btfss     STATUS,Z
          goto      salirCR                     ;break

          call      tcp_syn_rcvd

salirCR
          return
;*********************************************************************************



;****************************************************************
tcp_listen

          ;xm_ack[0] = seq[0]
          ;xm_ack[1] = seq[1]
          movf      seq0,W
          movwf     xm_ack0
          movf      seq1,W
          movwf     xm_ack1
          movf      seq2,W
          movwf     xm_ack2
          movf      seq3,W
          movwf     xm_ack3

          ;port = tcp_source_port
          movf      tcp_source_port1,W
          movwf     port1
          movf      tcp_source_port0,W
          movwf     port0

          ;Save the source IP address
          ;for (i=0 ; i<4 ; i++)
    ;ip[i] = source_ip[i];
          movf      source_ip0,W
          movwf     ip0
          movf      source_ip1,W
          movwf     ip1
          movf      source_ip2,W
          movwf     ip2
          movf      source_ip3,W
          movwf     ip3

          ;Calculate data length
          ;tcp_data_len = IP_length - (hdr_len + offset);
          bsf               STATUS,RP0
          movf      offset,W
          addwf     hdr_len,W             ;hdr_len + offset
          bcf               STATUS,RP0
```

```
          movwf     numero2_0
          clrf      numero2_1           ;numero2=(hdr_len + offset)
          movf      IP_length0,W
          movwf     numero1_0
          movf      IP_length1,W
          movwf     numero1_1
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      restar                        ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          movf      resultado0,W
          bsf                 STATUS,RP0
          movwf     tcp_data_len0
          bcf                 STATUS,RP0
          movf      resultado1,W
          bsf                 STATUS,RP0
          movwf     tcp_data_len1
          bcf                 STATUS,RP0

          ;Increment sequence to account for SYN and add data
          ;xm_ack[1] += tcp_data_len + 1;
          bsf                 STATUS,RP0
          incf      tcp_data_len0,F
          btfss     STATUS,Z
          goto      continuarSS
          incf      tcp_data_len1,F
          bcf                 STATUS,RP0
continuarSS
          bsf                 STATUS,RP0
          movf      tcp_data_len0,W
          bcf                 STATUS,RP0
          movwf     numero1_0
          bsf                 STATUS,RP0
          movf      tcp_data_len1,W
          bcf                 STATUS,RP0
          movwf     numero1_1
          movf      xm_ack2,W
          movwf     numero2_1
          movf      xm_ack3,W
          movwf     numero2_0           ;numero2=
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      sumar                         ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          movf      resultado0,W
          movwf     xm_ack3
          movf      resultado1,W
          movwf     xm_ack2

          ;if (xm_ack[1] < seq[1] )xm_ack[0]++
          ;restar xm_ack[1] - seq[1], si da positiva la resta es mayor
          ; si da negativa es menor que cero
          movf      xm_ack3,W
          movwf     numero1_0
          movf      xm_ack2,W
          movwf     numero1_1           ;numero1=xm_ack[1]
          movf      seq3,W
          movwf     numero2_0
          movf      seq2,W
          movwf     numero2_1           ;numero2=seq[1]

          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      restar                        ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          btfsc     POSITIVA,0
          goto      continuarTCPL1
          ;xm_ack[0]++
          movf      xm_ack0,W
          movwf     numero1_1
          movf      xm_ack1,W
          movwf     numero1_0           ;numero1=xm_ack[0]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0           ;numero2=1

          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      sumar                         ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          movf      resultado0,W
          movwf     xm_ack1
          movf      resultado1,W
          movwf     xm_ack0
continuarTCPL1

          ;Need to handle data if any here.
```

```
          ;Let's reply with our on SYN and an ACK
          ;We should include a Max Recieve Window option
          ;xm_seq[1]++;
          movf      xm_seq3,W
          movwf     numero1_0
          movf      xm_seq2,W
          movwf     numero1_1           ;numero1=xm_ack[0]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0           ;numero2=1

          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      sumar                         ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          movf      resultado1,W
          movwf     xm_seq2
          movf      resultado0,W
          movwf     xm_seq3

          ;if (xm_seq[1] == 0)
          ;xm_seq[0]++
          movlw     0x00
          subwf     xm_seq2,W
          btfss     STATUS,Z
          goto      salirIFTCPL1
          movlw     0x00
          subwf     xm_seq3,W
          btfss     STATUS,Z
          goto      salirIFTCPL1

          movf      xm_seq0,W           ;xm_seq[0]++
          movwf     numero1_1
          movf      xm_seq1,W
          movwf     numero1_0           ;numero1=xm_ack[0]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0           ;numero2=1

          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      sumar                         ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          movf      resultado1,W
          movwf     xm_seq0
          movf      resultado0,W
          movwf     xm_seq1

salirIFTCPL1
          ;load_ethernet_header()
          call      load_ethernet_header ;12 bytes (2 bytes for protocol included in load_IP..)


          ;load_IP_header(48,TCP) Number of bytes included in call (add 2 for protocol)
          ;20 for IP header
     ;20 for TCP header
     ;8 for Options
          bsf                 STATUS,RP0
          clrf      IP_packet_length1
          movlw     0x30      ;48 decimal
          movwf     IP_packet_length0
          movlw     TCP
          movwf     IP_send_protocol
          bcf                 STATUS,RP0
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      load_IP_header
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          ;load_TCP_header( 0x7000 | TCP_SYN | TCP_ACK, 0x07B8, 28);
     ;Number of bytes include in load_IP_header call
          movlw     0x70
          bsf                 STATUS,RP0
          movwf     data_flags1
          movlw     0x00
          movwf     data_flags0
          bsf                 data_flags0,1       ;TCP_SYN
          bsf                 data_flags0,4       ;TCP_ACK
          movlw     0x07
          movwf     tcp_chksum1
          movlw     0xb8
          movwf     tcp_chksum0
          clrf      tcp_length1
          movlw     0x1c                          ;28 decimal
          movwf     tcp_length0
          bcf                 STATUS,RP0
```

```
        bcf             PCLATH,4
        bsf             PCLATH,3
        call    load_tcp_header
        bcf             PCLATH,4
        bcf             PCLATH,3
        ;Send options
        ;outport(NIC_DATA,0x02)
        movlw   NIC_DATA
        movwf   REGISTRO
        movlw   0x02
        movwf   VALOR
        call    outport

        ;outdata(0x04);
        movlw   0x04
        movwf   VALOR
        call    outport
        ;outdata(0x05)
        movlw   0x05
        movwf   VALOR
        call    outport
        ;outdata(0xB4)
        movlw   0xb4
        movwf   VALOR
        call    outport
        ;outdata(0x00)
        movlw   0x00
        movwf   VALOR
        call    outport
        ;outdata(0x00)
        movlw   0x00
        movwf   VALOR
        call    outport
        ;outdata(0x00)
        movlw   0x00
        movwf   VALOR
        call    outport
        ;outdata(0x00)
        movlw   0x00
        movwf   VALOR
        call    outport

        ;send_packet(62)
        bsf             STATUS,RP0
        clrf    LEN1
        movlw   0x3e                    ;62 decimal
        movwf   LEN0
        bcf             STATUS,RP0
        call    send_packet

        ;Calculate the next sequence number to send
        ;In this case we add 1 since we only sent a SYN
        ;xm_seq[1]++
        movf    xm_seq2,W
        movwf   numero1_1
        movf    xm_seq3,W
        movwf   numero1_0       ;numero1=xm_ack[1]
        movlw   0x00
        movwf   numero2_1
        movlw   0x01
        movwf   numero2_0       ;numero2=1
        bcf             PCLATH,4
        BSF             PCLATH,3        ;PAGINA 1 DE PROGRAMA
        call    sumar                   ; restar numero1 - numero2=resultado
        bcf             PCLATH,4
        bcf             PCLATH,3        ;PAGINA 0 DE PROGRAMA
        movf    resultado1,W
        movwf   xm_seq2
        movf    resultado0,W
        movwf   xm_seq3

        ;if (xm_seq[1] == 0)
;xm_seq[0]++
        movlw   0x00
        subwf   xm_seq2,W
        btfss   STATUS,Z
        goto    salirIFTCPL2
        movlw   0x00
        subwf   xm_seq3,W
        btfss   STATUS,Z
        goto    salirIFTCPL2

        movf    xm_seq0,W       ;xm_seq[0]++
        movwf   numero1_1
        movf    xm_seq1,W
        movwf   numero1_0       ;numero1=xm_ack[0]
        movlw   0x00
        movwf   numero2_1
        movlw   0x01
        movwf   numero2_0       ;numero2=1
        bcf             PCLATH,4
        BSF             PCLATH,3        ;PAGINA 1 DE PROGRAMA
        call    sumar                   ; restar numero1 - numero2=resultado
```

```
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA

          movf      resultado1,W
          movwf     xm_seq0
          movf      resultado0,W
          movwf     xm_seq1

salirIFTCPL2

          ;html_socket = SYN_RCVD
          movlw     SYN_RCVD
          bsf                 STATUS,RP0
          movwf     html_socket
          bcf                 STATUS,RP0
          return
;*****************************************************************

;*****************************************************************
tcp_syn_rcvd

          ;Calculate data length
          ;tcp_data_len = IP_length - (hdr_len + offset);
          bsf                 STATUS,RP0
          movf      offset,W
          addwf     hdr_len,W   ;hdr_len + offset
          bcf                 STATUS,RP0
          movwf     numero2_0
          clrf      numero2_1   ;numero2=(hdr_len + offset)
          movf      IP_length0,W
          movwf     numero1_0
          movf      IP_length1,W
          movwf     numero1_1
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      restar                        ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          movf      resultado0,W
          bsf                 STATUS,RP0
          movwf     tcp_data_len0
          bcf                 STATUS,RP0
          movf      resultado1,W
          bsf                 STATUS,RP0
          movwf     tcp_data_len1
          bcf                 STATUS,RP0

          ;Increment sequence to account for data
          ;xm_ack[1] += tcp_data_len
          bsf                 STATUS,RP0
          movf      tcp_data_len0,W
          bcf                 STATUS,RP0
          movwf     numero1_0
          bsf                 STATUS,RP0
          movf      tcp_data_len1,W
          bcf                 STATUS,RP0
          movwf     numero1_1
          movf      xm_ack2,W
          movwf     numero2_1
          movf      xm_ack3,W
          movwf     numero2_0           ;numero2=
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      sumar                         ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          movf      resultado0,W
          movwf     xm_ack3
          movf      resultado1,W
          movwf     xm_ack2

          ;if (xm_ack[1] < seq[1] )xm_ack[0]++
          ;restar xm_ack[1] - seq[1], si da positiva la resta es mayor
          ; si da negativa es menor
          movf      xm_ack2,W
          movwf     numero1_1
          movf      xm_ack3,W
          movwf     numero1_0           ;numero1=xm_ack[1]
          movf      seq2,W
          movwf     numero2_1
          movf      seq3,W
          movwf     numero2_0           ;numero2=seq[1]
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      restar                        ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          btfsc     POSITIVA,0
          goto      continuarTCPSR1
          ;xm_ack[0]++
          movf      xm_ack0,W
          movwf     numero1_1
          movf      xm_ack1,W
```

```
          movwf     numero1_0           ;numero1=xm_ack[0]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0           ;numero2=1
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          call      sumar                         ; restar numero1 - numero2=resultado
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          movf      resultado0,W
          movwf     xm_ack1
          movf      resultado1,W
          movwf     xm_ack0
continuarTCPSR1

          ;Need to handle data if any here.
          ;We'll assume there isn't any for now so no response is needed
          ;html_socket = ESTAB
          movlw     ESTAB
          bsf                 STATUS,RP0
          movwf     html_socket
          bcf                 STATUS,RP0
          return
;*******************************************************************************


;*****************************************************************

;*****************************************************************
;         HASTA AQUI ESTAMOS EN LA PAGINA CERO DE PROGRAMA
;*****************************************************************

          org       0x800               ;PAGINA UNO DE PROGRAMA
;*****************************************************************
;*****************************************************************
;load_IP_header
;                   entradas:IP_packet_length0, LSB
;                                             IP_packet_length1, MSB
;                                             IP_send_protocol
load_IP_header

          ;outport(NIC_DATA, IP >> 8);   // IP packet
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     IP1
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1
          ;outport(NIC_DATA, IP);
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     IP0
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1

          ;outportw(NIC_DATA, 0x4500);
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     0x45
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1
          movlw     0x00
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1

          ;outportw(NIC_DATA, IP_packet_length)
          movlw     NIC_DATA
          movwf     REGISTRO
          bsf                 STATUS,RP0
          movf      IP_packet_length1,W
          bcf                 STATUS,RP0
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3            ;PAGINA 1
```

```
        bsf             STATUS,RP0
        movf    IP_packet_length0,W
        bcf             STATUS,RP0
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1

        ;outportw(NIC_DATA, identification)
        movlw   NIC_DATA
        movwf   REGISTRO
        movf    identification1,W
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1
        movf    identification0,W
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1

        ;outportw(NIC_DATA, 0x0000);   // Okay to fragment
        movlw   NIC_DATA
        movwf   REGISTRO
        movlw   0x00
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1
        movlw   0x00
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1

        ;outportw(NIC_DATA, 0xFF00 + IP_send_protocol);
        movlw   0x00
        movwf   numero1_0
        movlw   0xff
        movwf   numero1_1                   ;numero1=0xFF00

        bsf             STATUS,RP0
        movf    IP_send_protocol,W
        bcf             STATUS,RP0
        movwf   numero2_0
        movlw   0x00
        movwf   numero2_1                   ;numero2=IP_send_protocol


        call    sumar                       ; sumar numero1 + numero2=resultado


        movlw   NIC_DATA
        movwf   REGISTRO
        movf    resultado1,W
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1
        movf    resultado0,W
        movwf   VALOR
        bcf             PCLATH,4
        bcf             PCLATH,3            ;PAGINA 0
        call    outport
        bcf             PCLATH,4
        bsf             PCLATH,3            ;PAGINA 1

;*** CALCULAMOS EL CHECKSUM****
        ;chksum = 0
        clrf    chksum0
        clrf    chksum1
        ;prev_chksum = chksum;
        movf    chksum0,W
        movwf   prev_chksum0
        movf    chksum1,W
        movwf   prev_chksum1

        ;chksum += 0x4500

        movlw   0x45
```

```
movwf     numero1_1           ;numero1 = 0x4500
movlw     0x00
movwf     numero1_0
movf      chksum1,W
movwf     numero2_1
movf      chksum0,W
movwf     numero2_0                     ;numero2 = chksum

call      calcular_check

;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += IP_packet_length
bsf                 STATUS,RP0
movf      IP_packet_length1,W
bcf                 STATUS,RP0
movwf     numero1_1                     ;numero1 = IP_packet_length
bsf                 STATUS,RP0
movf      IP_packet_length0,W
bcf                 STATUS,RP0
movwf     numero1_0
movf      chksum1,W
movwf     numero2_1
movf      chksum0,W
movwf     numero2_0                     ;numero2 = chksum

call      calcular_check

;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += identification
movf      identification1,W
movwf     numero1_1                     ;numero1 =
movf      identification0,W
movwf     numero1_0
movf      chksum1,W
movwf     numero2_1
movf      chksum0,W
movwf     numero2_0                     ;numero2 = chksum

call      calcular_check

;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += 0x0000
movlw     0x00
movwf     numero1_1                     ;numero1 =
movlw     0x00
movwf     numero1_0
movf      chksum1,W
movwf     numero2_1
movf      chksum0,W
movwf     numero2_0                     ;numero2 = chksum

call      calcular_check

;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1


;chksum += 0xff+IP_send_protocol
movlw     0xff
movwf     numero1_1                     ;numero1 =
bsf                 STATUS,RP0                          ;banco 1
movf      IP_send_protocol,W
bcf                 STATUS,RP0                          ;banco 0
movwf     numero1_0
movf      chksum1,W
movwf     numero2_1
movf      chksum0,W
movwf     numero2_0                     ;numero2 = chksum

call      calcular_check

;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
```

```
          movwf     prev_chksum1

          ;chksum += (my_ip[0]<<8) + my_ip[1]
          movf      my_ip0,W
          movwf     numero1_1                     ;numero1 =
          movf      my_ip1,W
          movwf     numero1_0
          movf      chksum1,W
          movwf     numero2_1
          movf      chksum0,W
          movwf     numero2_0                     ;numero2 = chksum

          call      calcular_check

          ;prev_chksum = chksum
          movf      chksum0,W
          movwf     prev_chksum0
          movf      chksum1,W
          movwf     prev_chksum1


;chksum += (my_ip[2]<<8) + my_ip[3]
          movf      my_ip2,W
          movwf     numero1_1                     ;numero1 =
          movf      my_ip3,W
          movwf     numero1_0
          movf      chksum1,W
          movwf     numero2_1
          movf      chksum0,W
          movwf     numero2_0                     ;numero2 = chksum

          call      calcular_check

          ;prev_chksum = chksum
          movf      chksum0,W
          movwf     prev_chksum0
          movf      chksum1,W
          movwf     prev_chksum1


          ;chksum += ((long)source_ip[0]<<8) + source_ip[1]
          movf      source_ip0,W
          movwf     numero1_1                     ;numero1 = source_ip0
          movf      source_ip1,W
          movwf     numero1_0
          movf      chksum1,W
          movwf     numero2_1
          movf      chksum0,W
          movwf     numero2_0                     ;numero2 = chksum

          call      calcular_check

          ;prev_chksum = chksum
          movf      chksum0,W
          movwf     prev_chksum0
          movf      chksum1,W
          movwf     prev_chksum1

          ;chksum += ((long)source_ip[2]<<8) + source_ip[3]
          movf      source_ip2,W
          movwf     numero1_1                     ;numero1 = source_ip2
          movf      source_ip3,W
          movwf     numero1_0                     ;numero2 = source_ip3
          movf      chksum1,W
          movwf     numero2_1
          movf      chksum0,W
          movwf     numero2_0                     ;numero2 = chksum

          call      calcular_check

          ;outportw(NIC_DATA, ~chksum);  // Header Checksum
          comf      chksum0,F
          comf      chksum1,F
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      chksum1,W
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3                    ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3                    ;PAGINA 1
          movf      chksum0,W
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3                    ;PAGINA 0
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3                    ;PAGINA 1


          ;for (i=0 ; i<4 ; i++)   // source IP address for this packet
     ;outport(NIC_DATA, my_ip[i])
```

```
        movlw     NIC_DATA
        movwf     REGISTRO
        movf      my_ip0,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1
        movf      my_ip1,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1
        movf      my_ip2,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1
        movf      my_ip3,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1


        ;for (i=0 ; i<4 ; i++)   // destination IP address for this packet
    ;outport(NIC_DATA, source_ip[i]);
        movlw     NIC_DATA
        movwf     REGISTRO
        movf      source_ip0,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1
        movf      source_ip1,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1
        movf      source_ip2,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1
        movf      source_ip3,W
        movwf     VALOR
        bcf               PCLATH,4
        bcf               PCLATH,3                    ;PAGINA 0
        call      outport
        bcf               PCLATH,4
        bsf               PCLATH,3                    ;PAGINA 1

        return

;*****************************************************************
;load_tcp_header:
;         entradas: data_flags1
;                                                 data_flags0
;                                                 tcp_chksum1
;                                                 tcp_chksum0
;                                                 tcp_length1
;                                                 tcp_length0
load_tcp_header

         ;chksum = tcp_chksum;
         bsf               STATUS,RP0
         movf      tcp_chksum1,W
         bcf               STATUS,RP0
         movwf     chksum1
         bsf               STATUS,RP0
         movf      tcp_chksum0,W
         bcf               STATUS,RP0
         movwf     chksum0

         ;calculate checksum of pseudo header
         ;prev_chksum = chksum;
         movf      chksum1,W
         movwf     prev_chksum1
         movf      chksum0,W
         movwf     prev_chksum0
```

```
;chksum += ((long)my_ip[0]<<8) + my_ip[1];
movf      my_ip0,W
movwf     numero1_1
movf      my_ip1,W
movwf     numero1_0           ;numero1 = my_ip0
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check


;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += ((long)my_ip[2]<<8) + my_ip[3];
movf      my_ip2,W
movwf     numero1_1
movf      my_ip3,W
movwf     numero1_0           ;numero1 = my_ip2
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += ((long)source_ip[0]<<8) + source_ip[1];
movf      source_ip0,W
movwf     numero1_1
movf      source_ip1,W
movwf     numero1_0           ;numero1 = source_ip0
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += ((long)source_ip[2]<<8) + source_ip[3];
movf      source_ip2,W
movwf     numero1_1
movf      source_ip3,W
movwf     numero1_0           ;numero1 = source_ip2
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += TCP
movlw     TCP
movwf     numero1_0
clrf      numero1_1           ;numero1 = TCP
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1

;chksum += tcp_length
bsf               STATUS,RP0
movf      tcp_length0,W
bcf               STATUS,RP0
movwf     numero1_0
bsf               STATUS,RP0
movf      tcp_length1,W
bcf               STATUS,RP0
movwf     numero1_1           ;numero1 = tcp_length
movf      chksum0,W
movwf     numero2_0
```

```
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1


;chksum += SOURCE PORT
movf      tcp_dest_port1,W
movwf     numero1_1
movf      tcp_dest_port0,W
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1


;chksum += DESTINO PORT
movf      tcp_source_port1,W
movwf     numero1_1
movf      tcp_source_port0,W
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1


;chksum += NUMERO DE SECUENCIA
movf      xm_seq0,W
movwf     numero1_1
movf      xm_seq1,W
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1


;chksum += NUMERO DE SECUENCIA
movf      xm_seq2,W
movwf     numero1_1
movf      xm_seq3,W
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1


;chksum += NUMERO DE ACUSE
movf      xm_ack0,W
movwf     numero1_1
movf      xm_ack1,W
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
```

```
movf      chksum1,W
movwf     prev_chksum1


;chksum += NUMERO DE ACUSE
movf      xm_ack2,W
movwf     numero1_1
movf      xm_ack3,W
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1



;chksum += BANDERAS
bsf                 STATUS,RP0
movf      data_flags1,W
bcf                 STATUS,RP0
movwf     numero1_1
bsf                 STATUS,RP0
movf      data_flags0,W
bcf                 STATUS,RP0
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1



;chksum += 0X05B4
movlw     0x05
movwf     numero1_1
movlw     0xb4
movwf     numero1_0           ;numero1 =
movf      chksum0,W
movwf     numero2_0
movf      chksum1,W
movwf     numero2_1           ;numero2 = chksum
call      calcular_check
;prev_chksum = chksum;
movf      chksum0,W
movwf     prev_chksum0
movf      chksum1,W
movwf     prev_chksum1




;source port
;outportw(NIC_DATA, tcp_dest_port);
movlw     NIC_DATA
movwf     REGISTRO
movf      tcp_dest_port1,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movf      tcp_dest_port0,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3



;destination port
;outportw(NIC_DATA, tcp_source_port);
movlw     NIC_DATA
movwf     REGISTRO
movf      tcp_source_port1,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
```

```
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movf      tcp_source_port0,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3

 ;sequence number
;outportw(NIC_DATA, xm_seq[0]);
;outportw(NIC_DATA, xm_seq[1]);
movlw     NIC_DATA
movwf     REGISTRO
movf      xm_seq0,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movf      xm_seq1,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movlw     NIC_DATA
movwf     REGISTRO
movf      xm_seq2,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movf      xm_seq3,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3

;acknowledment number
;outportw(NIC_DATA, xm_ack[0])
;outportw(NIC_DATA, xm_ack[1])
movlw     NIC_DATA
movwf     REGISTRO
movf      xm_ack0,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movf      xm_ack1,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movlw     NIC_DATA
movwf     REGISTRO
movf      xm_ack2,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3
movf      xm_ack3,W
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
call      outport
bcf                 PCLATH,4
bsf                 PCLATH,3

;data offset and flags
;outportw(NIC_DATA, data_flags)
movlw     NIC_DATA
movwf     REGISTRO
bsf                 STATUS,RP0
movf      data_flags1,W
bcf                 STATUS,RP0
movwf     VALOR
bcf                 PCLATH,4
bcf                 PCLATH,3
```

```
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3
          bsf                 STATUS,RP0
          movf      data_flags0,W
          bcf                 STATUS,RP0
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3

          ;window
          ;outportw(NIC_DATA, 1460) 1460 decimal = 0X05B4
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     0x05
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3
          movlw     0xb4
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3

          ;checksum
          ;outportw(NIC_DATA, ~chksum);
          comf      chksum0,F
          comf      chksum1,F
          movlw     NIC_DATA
          movwf     REGISTRO
          movf      chksum1,W
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3
          movf      chksum0,W
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3

          ;urgent pointer
          ;outportw(NIC_DATA, 0);
          movlw     NIC_DATA
          movwf     REGISTRO
          movlw     0x00
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3
          movlw     0x00
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3
          call      outport
          bcf                 PCLATH,4
          bsf                 PCLATH,3
          return
;*******************************************************************************


;*******************************************************************************
;suma: suma dos numeros de dos bytes cada uno
;numnero1_1 numero1_0 + numnero2_1 numero2_0 = resultado1 resultado0
sumar
          clrf      POSITIVA
          movf      numero1_0,W
          addwf     numero2_0,W
          movwf     resultado0
          btfss     STATUS,C
          goto      continuarSumar
          movlw     1
          addwf     numero1_1,F
          btfss     STATUS,C
          goto      continuarSumar
          movlw     0x01
          movwf     POSITIVA

continuarSumar
```

```
          movf      numero1_1,W
          addwf     numero2_1,W
          movwf     resultado1
          btfss     STATUS,C
          goto      continuarSalir
          movlw     0x01
          movwf     POSITIVA

continuarSalir
          return
;*********************************************************************************
;restar
;restamos numero1 - numero2
restar
          MOVLW     1
          MOVWF     POSITIVA
          movf      numero2_0,W
          subwf     numero1_0,W
          movwf     resultado0
          btfsc     STATUS,C
          goto      continuarRestar
          movlw     1
          subwf     numero1_1,F
          btfss     STATUS,C
          goto      era_cero
          goto      continuarRestar
era_cero
          clrf      POSITIVA
          goto      salir_restar
continuarRestar
          movf      numero2_1,W
          subwf     numero1_1,W
          movwf     resultado1
          btfsc     STATUS,C
          goto      salir_restar
          clrf      POSITIVA
salir_restar
          return
;*********************************************************************************
calcular_check:
          ;bcf      PCLATH,4
          ;BSF      PCLATH,3            ;PAGINA 1 DE PROGRAMA
          ;call     sumar                         ; sumar numero1 + numero2=resultado


          movf      numero1_0,W
          addwf     numero2_0,W
          movwf     resultado0
          btfss     STATUS,C
          goto      continuarSumar1
          movlw     1
          addwf     numero1_1,F

continuarSumar1
          movf      numero1_1,W
          addwf     numero2_1,W
          movwf     resultado1


          ;bcf      PCLATH,4
          ;bcf      PCLATH,3            ;PAGINA 0 DE PROGRAMA
          movf      resultado1,W
          movwf     chksum1
          movf      resultado0,W
          movwf     chksum0                       ;el resultado de la suma guardarla en chksum
          ;if (chksum<prev_chksum) chksum++;
          ;chksum-prev_chksum, si el resultado es positivo C=1 chksum>prevchksum
          movf      chksum0,W
          movwf     numero1_0
          movf      chksum1,W
          movwf     numero1_1                      ;numero1=chksum
          movf      prev_chksum0,W
          movwf     numero2_0
          movf      prev_chksum1,W
          movwf     numero2_1                     ;numero2=
          ;bcf      PCLATH,4
          ;BSF      PCLATH,3            ;PAGINA 1 DE PROGRAMA
          ;call     restar                        ; restar numero1 - numero2=resultado

          MOVLW     1
          MOVWF     POSITIVA
          movf      numero2_0,W
          subwf     numero1_0,W
          movwf     resultado0
          btfsc     STATUS,C
          goto      continuarRestar1
          movlw     1
          subwf     numero1_1,F
          btfss     STATUS,C
          goto      era_cero1
          goto      continuarRestar1
era_cero1
          clrf      POSITIVA
```

```
          goto      salir_restar1
continuarRestar1
          movf      numero2_1,W
          subwf     numero1_1,W
          movwf     resultado1
          btfsc     STATUS,C
          goto      salir_restar1
          clrf      POSITIVA
salir_restar1


          ;bcf      PCLATH,4
          ;bcf      PCLATH,3                    ;PAGINA 0 DE PROGRAMA
          btfsc     POSITIVA,0                  ;si es 0 es negativa, si es 1 es positiva
          goto      continuar_LIPH1
          ;chksum++
          incf      chksum0,F
          btfsc     STATUS,Z
          incf      chksum1,F
continuar_LIPH1
          return
;*****************************************************************

;*******************************************************
;*******************************************************
;*******************************************************

enviar_datos_TCP


          ;cargar direccion MAC del host
          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_address0,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_address0

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_address1,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_address1

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_address2,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_address2

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_address3,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_address3

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_address4,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_address4

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_address5,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_address5


          ;cargar direccion IP del host
          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_ip0,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_ip0

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_ip1,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_ip1

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_ip2,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_ip2

          bsf                STATUS,RP0                 ;banco 1
          movf      Bsource_ip3,W
          bcf                STATUS,RP0                 ;banco 0
          movwf     source_ip3


          ;NDATOS_TCP: contador de datos a enviar por el modulo TCP/IP
          ;calcular el numero de datos a enviar
    bsf             STATUS,RP0                  ;banco 1
          clrf       NDATOS_TCP                 ;iniciar a cero el contador de bytes
          bcf                STATUS,RP0                 ;banco 0
          movlw     BUFFER_SALIDA       ;apuntas al inicio de buffer de direcciones
          movwf     FSR
          decf      FSR,F                       ;solo para despues incrementarlo
LazoEDT
```

```
          incf      FSR,F                       ;incrementamos el puntero del buffer de salida
    bsf             STATUS,RP0                  ;banco 1
          incf      NDATOS_TCP,F        ;incrementamos para indicar un dato mas enviado
;*********************************************************************

;         movf      SEPARADOR_CONTROLADOR,W       ;revisamos si el dato enviado es el fin
          movlw     0x0a      ;revisamos si el dato enviado es el fin

;*********************************************************************
          bcf               STATUS,RP0                  ;banco 0
          subwf     INDF,W                      ;comparar con el fin de transmicion
          btfss     STATUS,Z            ;si es asi continuar
          goto      LazoEDT                     ;si no regresar a seguir contando


;******************
    bsf             STATUS,RP0                  ;banco 1
          clrf      IMPAR                       ;indicar que es par
          btfss     NDATOS_TCP,0        ;verificar si NDATOS_TCP es par
          goto      NDATOS_TCP_es_par
;NDATOS_TCP_es_impar
          bsf               IMPAR,0                     ;indicar que es impar
          incf      NDATOS_TCP,F        ;incrementamos para que sea par
          bcf               STATUS,RP0                  ;banco 0
          incf      FSR,F                       ;incrementamos el puntero del buffer de salida
          movlw     0x00
          movwf     INDF
NDATOS_TCP_es_par:

;******************
    bsf             STATUS,RP0                  ;banco 1
          movf      NDATOS_TCP,W
          call      RS232_Tx
          bcf               STATUS,RP0                  ;banco 0
;******************
          ;CARGAR CABECERA ETHERNET
          bcf               PCLATH,4
          bcf               PCLATH,3            ;PAGINA 0 DE PROGRAMA
          call      load_ethernet_header
          bcf               PCLATH,4
          bsf               PCLATH,3            ;PAGINA 1 DE PROGRAMA

;load_IP_header(40+ longitud mensaje,TCP)
          bsf               STATUS,RP0                          ;banco 1
          clrf      IP_packet_length1
          movlw     0x28                                ;40 decimal= 20 for IP header + 20 for TCP header
          movwf     IP_packet_length0   ;longitud del paquete IP

;***esto he agregado
          bsf               STATUS,RP0                  ;banco 1
          btfss     IMPAR,0   ;verificar si NDATOS_TCP es par
          goto      NDATOS_TCP_es_par2
;NDATOS_TCP_es_impar
          clrf      IMPAR

          decf      NDATOS_TCP,F        ;incrementamos para que sea par
          bcf               STATUS,RP0                  ;banco 0
          incf      FSR,F                       ;incrementamos el puntero del buffer de salida

NDATOS_TCP_es_par2:
;***hasta aqui he agregado







          bsf               STATUS,RP0                          ;banco 1
          movf      NDATOS_TCP,W                ;sumarle el numero de datos a enviar
          addwf     IP_packet_length0,F
          movlw     TCP                                         ;indicar tipo de procolo que trae el paquete
          movwf     IP_send_protocol
          bcf               STATUS,RP0                          ;banco 0
          call      load_IP_header              ;cargamos el paquete IP en el RTL

          ;load_TCP_header( 0x5000 | TCP_ACK, chksum_buffer_in, 8+20);
    ;Number of bytes include in load_IP_header call
          movlw     0x50
          bsf               STATUS,RP0
          movwf     data_flags1
          movlw     0x00
          movwf     data_flags0
          bcf               data_flags0,1               ;TCP_SYN
          bsf               data_flags0,4               ;TCP_ACK
          bcf               STATUS,RP0



          call      chksum_buffer_salida ;calcular checksum de datos a enviar
          movf      chksum1,W
          bsf               STATUS,RP0
```

```
          movwf     tcp_chksum1
          bcf               STATUS,RP0
          movf      chksum0,W
          bsf               STATUS,RP0
          movwf     tcp_chksum0
          clrf      tcp_length1
          movlw     0x14                    ;20 decimal CABECERA + N° DE DATOS= 0x14 + NDATOS_TCP
          movwf     tcp_length0









          movf      NDATOS_TCP,W            ;sumarle el numero de datos a enviar
          addwf     tcp_length0,F
          bcf               STATUS,RP0
          call      load_tcp_header

          ;agregamos los datos del paquete TCP
          call      cargar_mensaje

          ; Y enviamos el paquete
          ;send_packet(n°datos+40+12+2)
          bsf               STATUS,RP0
          clrf      LEN1
          movlw     0x36                    ;54 decimal
          movwf     LEN0
          movf      NDATOS_TCP,W
          addwf     LEN0,F
          bcf               STATUS,RP0
          bcf               PCLATH,4
          bcf               PCLATH,3        ;PAGINA 0 DE PROGRAMA
          call      send_packet
          bcf               PCLATH,4
          bsf               PCLATH,3        ;PAGINA 1 DE PROGRAMA


                    ;Calcular el siguiente numero de secuencia a enviar

          ;xm_seq[1]++
          movf      xm_seq2,W
          movwf     numero1_1
          movf      xm_seq3,W
          movwf     numero1_0           ;numero1=xm_ack[1]
          movlw     0x00
          movwf     numero2_1
          bsf               STATUS,RP0
          movf      NDATOS_TCP,W        ;numero de datos enviados
          bcf               STATUS,RP0
          movwf     numero2_0           ;numero2=1
          call      sumar
          movf      resultado1,W
          movwf     xm_seq2
          movf      resultado0,W
          movwf     xm_seq3


;*******************
          btfss     POSITIVA,0
          goto      continuar10
          movf      xm_seq0,W
          movwf     numero1_1
          movf      xm_seq1,W
          movwf     numero1_0           ;numero1=xm_ack[1]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0           ;numero2=1
          call      sumar
          movf      resultado1,W
          movwf     xm_seq0
          movf      resultado0,W
          movwf     xm_seq1

          btfss     POSITIVA,0
          goto      continuar10

          movf      xm_seq2,W
          movwf     numero1_1
          movf      xm_seq3,W
          movwf     numero1_0           ;numero1=xm_ack[1]
          movlw     0x00
          movwf     numero2_1
          movlw     0x01
          movwf     numero2_0           ;numero2=1
          call      sumar
          movf      resultado1,W
          movwf     xm_seq2
          movf      resultado0,W
```

```
          movwf     xm_seq3
;******************
continuar10


          ;indicar buffer de entrada listo para otro paquete de datos
;         bsf       STATUS,RP0
;         bcf       FIN_LINEA,0
;         bcf       STATUS,RP0

;*********************************
;         bcf                 STATUS,IRP                    ;banco 0,1(direccionamiento indirecto)
          movlw     BUFFER_ENTRADA
          movwf     FSR


          return

;*****************************************************************
;*****************************************************************
chksum_buffer_salida:
          clrf      chksum0                       ;inicialisamos el checksum a cero
          clrf      chksum1
          clrf      prev_chksum0        ;previo checksum igual a checksum
          clrf      prev_chksum1
          ;Apuntamos al inicio de buffer de salida
          movlw     BUFFER_SALIDA
          movwf     FSR
          decf      FSR,F
sumar_datos
          incf      FSR,F
          movf      INDF,W
          movwf     numero1_1
          incf      FSR,F
          movf      INDF,W
          movwf     numero1_0
          movf      chksum0,W
          movwf     numero2_0
          movf      chksum1,W
          movwf     numero2_1           ;numero2 = chksum
          call      calcular_check
          ;prev_chksum = chksum;
          movf      chksum0,W
          movwf     prev_chksum0
          movf      chksum1,W
          movwf     prev_chksum1
          ;revisamos si el ultimo byte es fin de linea
          movf      INDF,W
          sublw     0x0a
          btfsc     STATUS,Z
          goto      salirCalC
          ;revisamos si el ultimo byte es fin de linea
          decf      FSR,F
          movf      INDF,W
          sublw     0x0a
          btfsc     STATUS,Z
          goto      salirCalC
          incf      FSR,F
          goto      sumar_datos                   ;si no era fin de linea regresar a sumar_datos

salirCalC
          return
;*****************************************************************
;*****************************************************************
cargar_mensaje:
          ;apuntamos al inicio del buffer
          movlw     BUFFER_SALIDA
          movwf     FSR
          movlw     NIC_DATA
          movwf     REGISTRO
agregar_datos
          movf      INDF,W
          movwf     VALOR
          bcf                 PCLATH,4
          bcf                 PCLATH,3            ;PAGINA 0 DE PROGRAMA
          call      outport
          bcf                 PCLATH,4
          BSF                 PCLATH,3            ;PAGINA 1 DE PROGRAMA
          incf      FSR,F                         ;apuntamos al siguiente dato a enviar
          ;revisamos si el dato enviado es el fin de cadena
          movf      VALOR,W
          sublw     0x0a
          btfss     STATUS,Z
          goto      agregar_datos
;******************
          bsf                 STATUS,RP0                    ;banco 1
          btfss     IMPAR,0
          goto      salir
          clrf      IMPAR
          bcf                 STATUS,RP0                    ;banco 0
          goto      agregar_datos

salir
```

```
          bcf               STATUS,RP0                  ;banco 0
;******************
          return
```

A 2. HOJAS TECNICAS

A 2.1 HOJAS TECNICAS DEL RTL8019AS

Estas hojas técnicas también pueden ser encontradas en el disco compacto de este trabajo de graduación.

A 3. GLOSARIO

Firmware: Software que esta embebido en un dispositivo de hardware. Este es muchas veces encontrado en memorias flash ROM o de otro tipo, y como en nuestro caso es el software ejecutado dentro del PIC16F877A.

GPIB: General Purpose Instrumentation Bus, un bus para instrumentación de propósito general

HTML: acrónimo inglés de HyperText Markup Language, iseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

JSP: Java Server Page, es una tecnología de Java que permite a los desarrolladores de software generar páginas HTML dinámicas, XML y otros tipos de documentos para responder a un cliente WEB. Esta tecnología permite al código Java ser introducido (embebido) para ejecutar ciertas acciones predefinidas dentro de contenido estático.

LAN: Local Area Network, red de trabajo de área local o un grupo de ordenadores personales conectados en red

OSI: modelo de referencia de interconexión de sistemas abiertos.

Protocolo TCP/IP: Transmission Control Protocol / Internet Protocol, son los núcleos del conjunto de protocolos de internet, TCP es usado para crear conexiones entre múltiples host en red sobre las cuales se pueden intercambiar datos (paquetes).

Software: Programas que habilitan a la computadora para realizar una tarea especifica complementada por componentes físicos en un sistema (hardware).