# UNIVERSIDAD NACIONAL DEL ALTIPLANO

## FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

## ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

# "DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ UTILIZANDO EL PROTOCOLO MODBUS PARA LA COMUNICACIÓN EN REDES INDUSTRIALES"

**TESIS**

PRESENTADO POR:

**WILDER CLAUDIO TICONA APAZA**

PARA OPTAR EL TÍTULO PROFESIONAL DE:

**INGENIERO ELECTRÓNICO**

**PUNO – PERÚ**

**2014**

**UNIVERSIDAD NACIONAL DEL ALTIPLANO**

**FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS**

**ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA**

"DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ UTILIZANDO EL PROTOCOLO MODBUS PARA LA COMUNICACIÓN EN REDES INDUSTRIALES"

**TESIS**

PRESENTADA POR:

**WILDER CLAUDIO TICONA APAZA**

PARA OPTAR EL TÍTULO PROFESIONAL DE:

**INGENIERO ELECTRÓNICO**

**APROBADA POR EL JURADO REVISOR CONFORMADO POR:**

PRESIDENTE : _____

Mg. Ing. IVÁN DELGADO HUAYTA

PRIMER MIEMBRO : _____

M. Sc. Ing. GAVINO JOSÉ FLORES CHIPANA

SEGUNDO MIEMBRO: _____

Ing. JOSÉ RONALD CONDORI PAREDES

DIRECTOR DE TESIS: _____

M. Sc. Ing. GUIDO HUMBERTO CAYO CABRERA

ASESOR DE TESIS : _____

M. Sc. Ing. DAVID SALINAS MENDOZA

PUNO – PERÚ

2014

**ÁREA: Automatización e instrumentación**
**TEMA: Control y automatización**

## DEDICATORIA

*A Dios, por permitirme vivir para alcanzar cada una de las metas trazadas, a mis padres y hermanos quienes han dado todo para hacer de mí una persona de Bien. A mi familia en general por creer en mis capacidades y apoyarme durante todo el trayecto recorrido, estuvieron siempre ahí para apoyarme.*

**AGRADECIMIENTOS**

*Agradezco a todas las personas que creyeron en mí, agradezco, al Ing. Ivan Delgado Huayta por brindarme el apoyo y consejos necesarios, que de alguna forma hicieron posible este trabajo.*

# ÍNDICE

## INDICE DE FIGURAS

## INDICE DE TABLAS

## RESUMEN

El presente trabajo se diseñó e implemento, tiene como finalidad de analizar los aspectos funcionales tanto de las Redes Industriales como del Protocolo Modbus, que permite la supervisión de variables física utilizando el Protocolo Modbus. La cual está elaborada por una interfaz compuesta por elementos software y hardware necesario haciendo uso de las redes industriales abiertas de mayor uso en nuestro medio, utilizando el servidor OPC para registrar datos críticos para gestión y monitoreo. Por otra parte, fue necesario construir dispositivos Electrónicos con el fin de lograr establecer conexiones multipunto, se utilizo como medio físico una interfaz RS232 a señales RS485 como maestro la PC, y dispositivos esclavos utilizando microcontroladores PIC16F877A, capaces de comunicarse con un dispositivo Maestro utilizando el Protocolo de Comunicación Modbus. Como lo proponen las especificaciones del protocolo utilizado. Como en toda Red Industrial Modbus, es indispensable la utilización de un dispositivo Maestro Modbus, por ello se desarrolló una Aplicación para una estación meteorológica que sea capaz de registrar las principales variables meteorológicas (Temperatura, humedad, presión atmosférica y velocidad del viento) mediante el uso de sensores digitales y almacenar los datos en un medio digital y de fácil acceso, utilizando el Control de Comunicación Modbus para manejar la parte de Comunicación y recuperación de los datos contenidos en los dispositivos Esclavos.

**Palabras claves:** Protocolo modbus, Norma RS485, Servidor OPC, especificaciones modbus.

**ABSTRACT**

The present work was designed and implemented, aims to analyze both the functional aspects of Industrial Networks as the Modbus Protocol, which allows monitoring of physical variables using the Modbus protocol. Which is produced by a complex of elements necessary software and hardware using open industrial networks increased use in our environment, using the OPC server to record critical data management and monitoring interface? Moreover, it was necessary to build Electronic devices in order to achieve set multipoint connections, was used as the physical layer interface RS232 signals to RS485 PC as master and slave devices using PIC16F877A microcontroller, able to communicate with a master device using the Modbus Communication Protocol. As proposed protocol specifications used. As in all Industrial Modbus Network, using a Modbus Master device, so an Application for a weather station that is able to record the main meteorological variables (temperature, humidity, atmospheric pressure and wind speed) is indispensable developed by the using digital sensors and store data in a digital environment and easily accessible using Modbus Communication Control to manage part of communication and retrieval of the data in the Slave devices.

**Keywords**: modbus protocol, standard RS485, OPC Server, Modbus specifications.

## INTRODUCCIÓN

Teniendo en cuenta la importancia del estudio e implementación de la automatización industrial en el campo empresarial e institucional, las redes industriales nacieron con el fin de unir todos los dispositivos que coexisten dentro de una empresa, dedicados al control de máquinas o partes de un proceso cerrado. Debido a la variedad de equipos y necesidades que se presentan dentro de la industria (que difieren muy a menudo entre una empresa y otra), se ha desarrollado una gran variedad de protocolos que permitan la comunicación de PCs (Computadores Personales) o PLCs (Controladores Lógicos Programables ó Autómatas Programables) con los diferentes instrumentos de campo. Uno de estos protocolos es Modbus, desarrollado por Modicon hacia el año de 1979.

El presente proyecto tiene como finalidad realizar el **DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ UTILIZANDO EL PROTOCOLO MODBUS PARA LA COMUNUNICACIÓN EN REDES INDUSTRIALES,** Se escogió el protocolo Modbus, para el desarrollo de este proyecto que, ha tenido una gran acogida desde su creación; logrando una gran difusión dentro de las redes industriales. Además es un protocolo de mediana complejidad, por lo que constituye una opción factible para el desarrollo de este proyecto.

**CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA, ANTECEDENTES, JUSTIFICACIÓN Y OBJETIVOS DE LA INVESTIGACIÓN,** se establece el planteamiento del problema de investigación, los antecedentes, la justificación y los objetivos que persigue la misma.

**CAPITULO II: MARCO TEÓRICO, MARCO CONCEPTUAL, E HIPÓTESIS DE INVESTIGACIÓN,** se considera el marco teórico, que comprende la revisión bibliográfica para el desarrollo de la investigación, y tener un panorama general del protocolo modbus.

**CAPITULO III: MÉTODO DE INVESTIGACIÓN,** se refiere al planteamiento del diseño de investigación, los métodos y técnicas de observación utilizando juntamente con

los procedimientos de análisis empleados, Comenzamos, detalladamente la conversión de RS232 a RS485 procedimientos y cálculos para la transmisión de datos luego los diseños de los esclavos, acondicionamiento de variables físicas.

Seguidamente, desarrollamos el software para esclavos y maestro que empleara supervisión y monitoreo de datos, para así tener todas las señales que ingresaran al microcontrolador.

**CAPÍTULO IV: EXPOSICIÓN Y ANÁLISIS DE LOS RESULTADOS,** se hace referencia a los resultados, en la cual se usa herramientas y sus aplicaciones las mismas que servirán para el desarrollo de la investigación, para la comprobación del correcto funcionamiento del protocolo modbus, de igual manera, se analizarán los resultados obtenidos en dichas pruebas.

.

# CAPÍTULO I

## PLANTEAMIENTO DEL PROBLEMA, ANTECEDENTES, JUSTIFICACIÓN Y OBJETIVOS DE LA INVESTIGACIÓN

## 1.1. Planteamiento del Problema

En los últimos años la evolución tecnológica en el ámbito industrial ha obligado la modernización de muchas empresas con la finalidad de mantener y mejorar su productividad. Este índice está en función de varios parámetros entre los cuales se encuentran los sistemas de automatización basados en arquitectura complejas, cada vez son mayores los sistemas que han implementado redes de comunicación industrial debido a las ventajas que ofrecen.

El desarrollo de estas tecnologías está afectando de distintas formas a otros ámbitos, tales como en los entornos industriales haciéndose notar en el desarrollo de los denominados buses de campo, profibus, modbus, fieldbus, etc., y en la amplia variedad de dispositivos periféricos son necesarios en los altos niveles de producción para ser competitivos, aumentar la calidad, incrementar la utilidad y disminuir al mínimo el riesgo de parada de planta.

Para las soluciones de automatización existen poderosas herramientas de comunicación a nivel de campo, la cual podría ser el uso de las redes industriales o más propiamente dicha los buses de campo. Dependiendo de las dimensiones de la planta industrial van apareciendo las necesidades de tener estaciones dedicadas a determinadas labores.

### 1.1.1. Definición del problema.

Ante esta problemática se planteó las siguientes interrogantes:

a) ¿Cómo se logrará realizar el diseño e implementación de una interfaz para una red industrial utilizando el protocolo modbus?
b) ¿En qué medida es posible realizar el diseño para la comunicación utilizando la norma física RS-485?

## 1.2. Antecedentes de la Investigación

Entre los antecedentes del problema se citó:

- En la comunicación digital, es importante determinar tanto el medio físico a través del cual se establece la comunicación, como las características de la información misma. En este punto es importante indicar que los equipos de instrumentación digitales difieren en el tipo de protocolo que utilizan para comunicarse, siendo necesario revisar las características de los mismos. Es por esto que se tiene como principio el estudio de tecnologías abiertas de comunicación de campo, algunas de ellas modernas y otras no tanto pero de gran utilidad en le pasado y con gran importancia en el mercado actual.

- **Gould Modicon:** Modbus es un protocolo de transmisión desarrollado por Gould Modicon (ahora una división de Schneider Electric), para sistemas de control y supervisión de procesos con control centralizado. Este protocolo es muy utilizado en la industria en una gran variedad de sistemas SCADA.

- **"Diseño e Implementación de una red MODBUS para el monitoreo de variables eléctricas desde un reconectador de circuitos NOJA propiedad de Levaduras Collico S.A.":** Realizado en el año 2008 en la universidad austral de Chile realizado por Cristian Omar Oyarzo López, en este trabajo se desarrolló e implemento un sistema capaz de monitorear en línea el consumo eléctrico total de meda tensión de la empresa levaduras collico S.A. Mediante una interfaz gráfica de usuario el personal de planta pudo visualizar los parámetros eléctricos desde cualquiera de las estaciones de operador presentes en la sala de control. De esta forma se eliminó la necesidad de que el personal ingresa zonas de alto voltaje para observar estos parámetros directamente en los dispositivos medidores de consumo disponibles en planta.

- **"Diseño de un extensor de entradas y salidas analógicaspor MODBUS RTU sobre RS – 485":** Realizado en el año 2012 en la universidad oberta de Catalunya realizado por José Soriano Miras, en este trabajo se diseñó e implemento un programa en lenguaje C se realiza la recepción, interpretación y ejecución de las órdenes recibidas a través de Modbus RTU, así como también de preparar los datos

solicitados y transmitirlos para responder a dispositivo principal.

### 1.3. Justificación de la Investigación

En la actualidad el Protocolo Modbus, es un bus de campo muy usado en la región y en el mundo debido a su simplicidad y especificación abierta, permitiendo llevar a cabo proyectos de gran complejidad y obtener un funcionamiento fiable, facilitando así un mayor rango de aplicaciones que son implementados por el sector privado, público y académico, por estas razones muchos dispositivos utilizan la comunicación Modbus tales como PLC, Human Machine Interface (HMI), sensores y actuadores remotos.

Además el Modbus tiene la ventaja que puede alcanzar distancias muy largas común buen aislamiento a las perturbaciones y conectar múltiples dispositivos a la red, dando significativas ventajas en el sector industrial. Este atributo con que cuenta el Modbus se debe a su capa física RS-485 que define sus niveles de voltaje y la cantidad de dispositivos que se pueden conectar en una misma red. Con lo anterior mencionado la comunicación Modbus en el ámbito didáctico permite implementar múltiples aplicaciones como comunicación con interfaces hombre máquina, comunicación con sensores y actuadores entre otras, dándole al estudiante una perspectiva más amplia respecto a las comunicaciones industriales y la adquisición remota de datos.

#### 1.3.1. Justificación Académica

La automatización industrial es una de las áreas que plantea grandes perspectivas de expansión. El estudio de esta área científica puede proporcionar una visión más generalizada y una toma de contacto con tecnologías extendidas y en auge, en el mundo no solo de la informática, sino también de los entornos industriales.

El diseño de una red industrial representa la base para la construcción de un laboratorio de automatización industrial que permitirá a los estudiantes estar en contacto con elementos tales como detectores, sensores, actuadores y unidades de control; permitiendo la capacitación de los alumnos en materia de planeación y manejo de proyectos de automatización con procesos de diversa índole.

### 1.3.2. Justificación Técnica

De acuerdo a los avances tecnológicos en las distintas áreas se desarrolló el diseño e implementación de una red industrial para una estación meteorológica, el cual contribuirá a la supervisión del estado climático una instalación destinada a medir y registrar regularmente diversas variables meteorológicas. Estos datos se utilizan tanto para la elaboración de predicciones meteorológicas a partir de modelos numéricos.

## 1.4. Objetivos de la Investigación

### 1.4.1. Objetivo General

- Diseñar e Implementar una Interfaz Utilizando el Protocolo Modbus Para la Comunicación en Redes Industriales.

### 1.4.2. Objetivos Específicos

- Diseñar una red industrial usando el protocolo Modbus.
- Implementar un programa en la plataforma abierta para un sistema operativo compatible, que controle la transmisión, con protocolo industrial modbus.

# CAPÍTULO II

## MARCO TEÓRICO, MARCO CONCEPTUAL, E HIPÓTESIS DE INVESTIGACIÓN

## 2.1. Marco Teórico

### 2.1.1. Definición de comunicación

Cuando hablamos de comunicación de datos, conviene distinguir entre datos e información, los datos los definimos como el conjunto de diferentes estados que pueden adoptar una variable, mientras que la información la definimos como el resultado de procesar e interpretar esos datos, que en muchos casos serán redundantes para garantizar una comunicación fiable como se muestra en la figura Figura 01.

Definimos comunicación como el proceso de intercambio de datos, de cuyo análisis posterior se obtiene la información. En la comunicación de datos intervienen varios elementos.

- Equipo emisor/receptor: Equipos que inervienen en la comunicación (ordenadores, PLCs, perifericos, etc.).
- Canal: Recurso o medio fisico capaz de propagar señales (cable eléctrico, aire, fibra optica, etc).
- Mensajes: Datos que se transfieren entre ambos equipos.

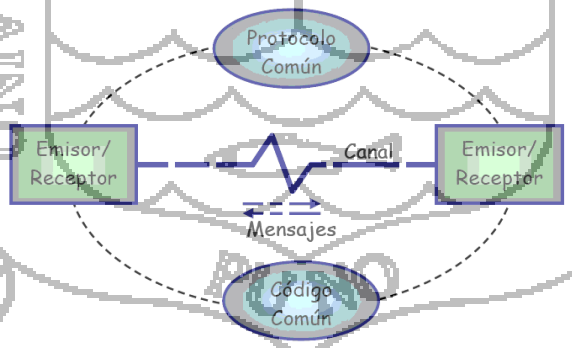



*Figura 01 - Elementos que intervienen en una comunicación.*

La Figura 02 muestra comunicación está sometida a perturbaciones y ruido de la misma naturaleza de las señales que circulan por el canal y que afectan negativamente a la transmisión. Además, los interlocutores deben utilizar el mismo código (es decir, el mismo

idioma, para poder entenderse); en caso contrario el receptor no podría transformar los datos recibido en información. Como consecuencia de la existencia de un flujo de datos bidireccional surge la necesidad de emplear un protocolo que no es sino un conjunto de reglas que regulan el flujo de la información también establecen mecanismos de control de detección de errores y recuperación de datos en caso de error.

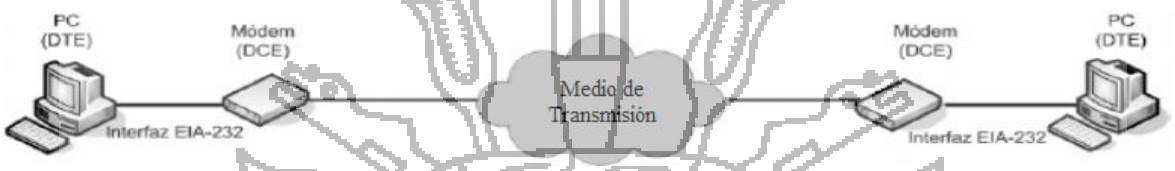



*Figura 02 -: Elementos de un sistema de comunicación de datos.*

### 2.1.2. Redes de Comunicación

La Figura 03 muestra las redes son más que un conjunto de medios para proporcionar servicios de telecomunicación entre cierto número de ubicaciones, una ubicación (fija o móvil) es conocida como punto de terminación de red (nodos de la red). Así pues, podríamos ver una red como algo abstracto que ofrece un determinado servicio en puntos de terminación de red. Una red es un conjunto de computadoras y/o dispositivos electrónicos (dos como mínimo), que se unen a través de medios físicos (hardware) y lógicos (software), con fin de llevar a cabo una actividad o labor de forma eficiente y eficaz.

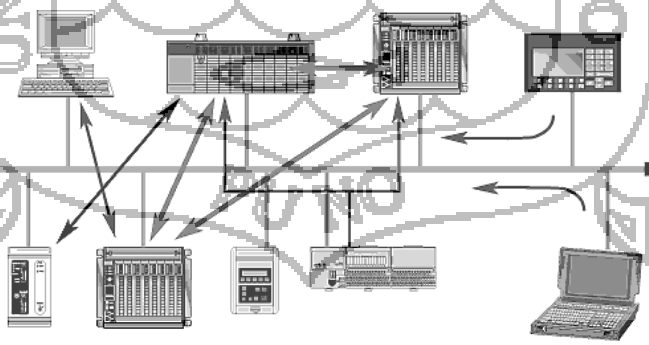


*Figura 03 - Redes de Comunicación.*

### 2.1.2.1. Componentes de una Red

#### *2.1.2.1.1. Nodos de Red*

Un nodo de red es aquel que tiene una o más tarjetas de red u otro medio que le permite comunicarse con la red.

- Servidores: En una red pueden haber varias computadoras que comparten sus recursos con la red, estas reciben el nombre de servidores.
- Estaciones: Aquellas computadoras que utilizan los recursos de los servidores se llaman estaciones.
- Dispositivos de Comunicación: permiten la comunicación entre los nodos de red.

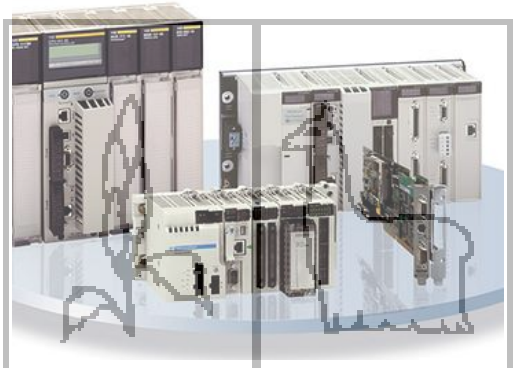


*Figura 04 - Nodos de red*

### 2.1.2.2. Medios de Transmisión

Para transmitir una señal eléctrica es necesario un medio de transmisión. La transmisión se consigue modificando algunos parámetros de la señal que transporta los datos (tensión, frecuencia, fase, y en algunos casos hasta intensidad).

#### *2.1.2.2.1. Cable par Trenzado*

Es de los más antiguos en el mercado y en algunos tipos de aplicaciones es el más común, consiste en dos alambres de cobre o a veces de aluminio, aislados con un grosor de 1 mm aproximado. Los alambres se trenzan con el propósito de reducir la interferencia

eléctrica de pares similares cercanos. Los pares trenzados se agrupan bajo una cubierta común de PVC (Policloruro de Vinilo) en la Figura 2.5 muestra los cables, se utilizan los siguientes tipos de cable pares trenzados:

- Cable de par Trenzado no Apantallado (UTP, Unshielded Twisted Pair): este tipo de cable no tienen ningún tipo de apantallado en el conductor, y es muy sensible a interferencias. Permite velocidades de transmisión de hasta 375Kbit/s sobre líneas de hasta 300m de largo.
- Cable de par Trenzado Apantallados (STP, Shielded Twisted Pair): es la denominación de los cables de par trenzado apantallados individualmente, cada par se envuelve en una malla conductora y otra general que recubre a todos los pares en este caso las distancias pueden llegar a los 1200m. Poseen gran inmunidad al ruido, pero una rigidez máxima.
- Cable de par Trenzado con Pantalla Global (FTP, Foiled Twisted Pair): En los cables FTP los pares se recubren de una malla conductora global en forma trenzada. De esta forma mejora la protección frente a interferencias, teniendo una rigidez intermedia, permiten una velocidad de transmisión de 1Mbit/s sobre distancias de hasta 2000m.

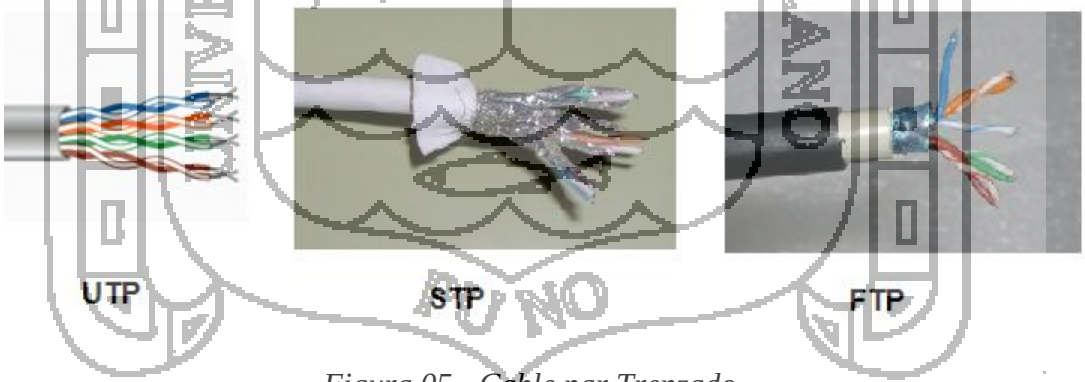



*Figura 05 - Cable par Trenzado.*

### *2.1.2.2.2. Cable Coaxial*

Permiten una alta velocidad de transmisión con la ventaja adicional que puede llevar muchos mensajes simultáneamente. El ancho de banda llega hasta 10Mhz. Los cables son

más caros que los trenzados y son raramente encontrados en el campo como se muestra en la Figura 06.

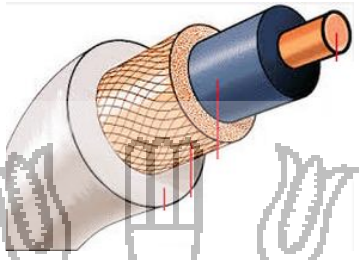


*Figura 06 - Cable Coaxial.*

***2.1.2.2.3. Cable Fibra Óptica***

Su capacidad de transmisión es 5 veces mayor a la del cable coaxial. El cable de fibra óptica contiene una fibra simple de vidrio la que por razones de estabilidad está rodeada de varios cubiertas protectoras de modo tal que es casi tan gruesa como un cable coaxial. Durante la transmisión, las señales eléctricas son convertidas en señales luminosas. Esto significa que los factores usuales de interferencia tales como campos electromagnéticos no tienen influencia, permiten velocidades de transmisión de Gigabits/s. debido al método más complicado de conexión como se muestra en la Figura 07.


*Figura 07 - Cable Fibra Optica.*

**2.1.3. Protocolos**

Se llama protocolo de comunicación al conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red. En este contexto, las entidades de las cuales se habla son programas de

computadora o automatismo de otro tipo, tales y como dispositivos electrónicos capaces de interactuar en una red, los protocolos gobiernan dos niveles de comunicación como se muestra en la Figura 08:

- Los protocolos de alto nivel: Estos definen la forma en que se comunican las aplicaciones. (Ej: Protocolo FTP, Protocolo SNMP).
- Los protocolos de bajo nivel: Estos definen la forma en que se transmiten las señales por cable. (Ej: Protocolo Modbus, Fielbus Foundation, Hard, Profibus ,TCP/IP).

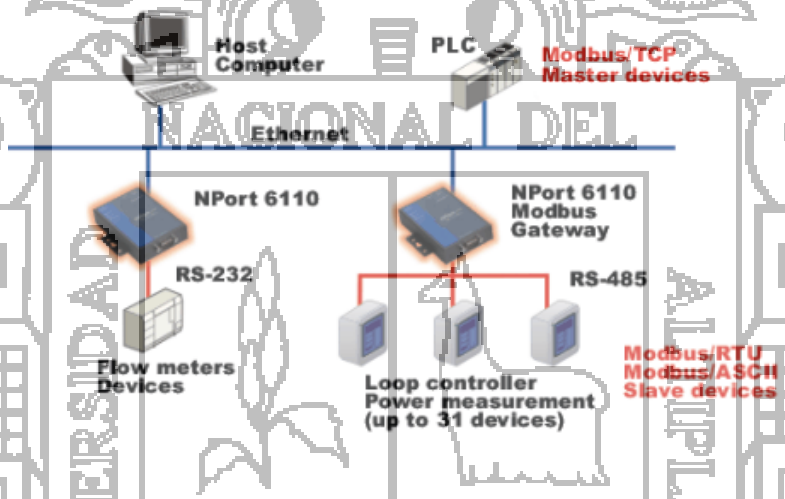



*Figura 08 - Protocolos de Comunicación.*

### 2.1.4. Tipos de Redes de comunicación

Podemos clasificar las redes en cuanto a las dimensiones de la tecnología de transmisión y del tamaño.

### 2.1.4.1. Tecnología de transmisión

#### *2.1.4.1.1. Modo Unicast*

La figura Figura 09 muestra la dirección unicast es el tipo más común en una red comunicación. El canal dirigido a un host específico[1].

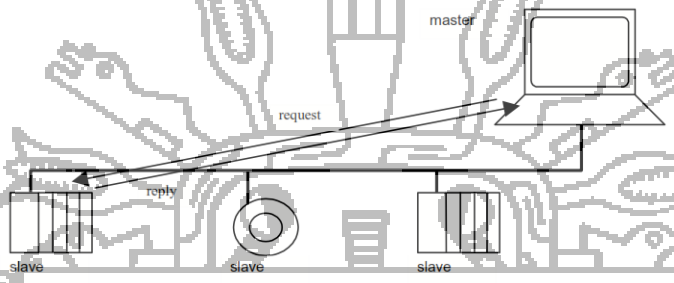



*Figura 09 - Modo Unicast.*

#### *2.1.4.1.2. Modo Broadcast*

Un solo canal de comunicación compartida por todas las maquinas, un paquete mandado por alguna maquina es recibido por todas las otras como se muestra Figura 10.

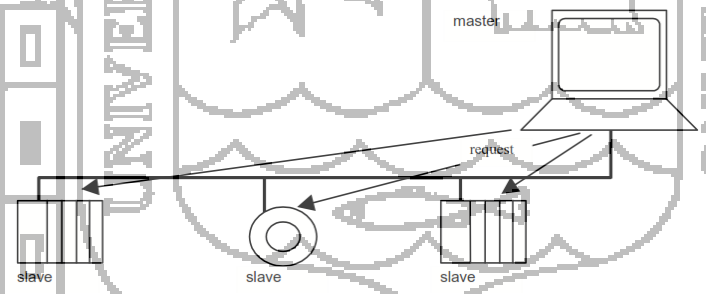



*Figura 10 - Modo Broadcast.*

### 2.1.4.2. Según su Escala o Tamaño

- LAN(Redes de Area Local): Se trata de redes en las que los equipos están distribuidos en una oficina o edificio (entre 10m  1Km), con velocidades típicas de comunicación entre 10Mbps hasta 1Gbps

[1] Las especificación para el modo Unicast y Broadcast aquí mencionadas fueron tomadas de MODBUS over Serial Line Specification y Imlemetation guide V1.0 Pag. 7.

- WAN (Redes de Área Extensa): Se trata de redes que interconectan equipos o redes de distintos lugares, y pueden cubrir áreas geográficas muy extensas. Normalmente se emplean redes públicas o privadas.

### 2.1.4.3. Según el Procesamiento

#### *2.1.4.3.1. Centralizado*

El host realiza las tareas de las terminales es cuando el control se realiza por un solo host.

Algunas características que remarcar son las siguientes:

- Es efectivo mientras el sistema no sea excesivamente grande ni compleja.
- Es fácil de mantener, ya que solo hay un único controlador.
- Al existir un único controlador, no existen problemas de compatibilidad.
- Son muy delicados a los fallos; si el controlador falla, todo se detiene.

#### *2.1.4.3.1. Distribuida*

Cada servidor o estación realiza tareas es cuando el control se realiza a través de diferentes sistemas conectados en red, las principales características son las siguientes.

- Para sistemas grandes o complejos.
- La responsabilidad es repartida entre diferentes controladores.
- Todos los controladores deben de comunicarse a través de una red.
- Su capacidad tiende a ser superior a un sistema centralizado.
- Se pueden hacer ampliaciones con otros controladores. Cuando estos están programados y con un funcionamiento correcto, entonces se integra en la red de comunicación de los demás controladores.
- Permite la integración de dispositivos de diferentes fabricantes comunicables entre sí.

### 2.1.5. Velocidades de modulación, transmisión y transferencia

Cuando nos referimos a la velocidad con que se transfiere o trasmiten datos, conviene distinguir entre estos tres términos, según nos referimos a cambios de estado de la señal, cantidad de datos, o cantidad de información útil.

- Velocidad de Modulación ($V_m$): Es el número de cambios de estado que se producen en la línea por segundo. De medida en baudios y un baudio puede equivaler a uno o más bits de datos.

- Velocidad de Transmisión ($V_t$): Es el número de bits que circulan por el canal por unidad de tiempo, y se mide en bits por segundo (bps).

- Velocidad de transferencia ($V_{tr}$): Es la cantidad en información útil que circula por el canal, suprimiendo los datos redundantes, en bits por segundo, como hemos comprobado con anterioridad, en la transmisión de información se añaden datos para la regulación de flujo de la transmisión, detección de errores, etc. Toda esta, es información redundante que no forma parte de los datos que el usuario o la aplicación desean trasmitir para lograr la comunicación. Así se denomina porcentaje de redundancia a la relación entre los bits de control y los bits totales.

$$R = \left[\frac{bits\ de\ control}{bits\ de\ control + bits\ utiles}\right] * 100$$

### 2.1.6. Modos de Comunicación

Otra característica de la comunicación digital entre dos dispositivos es el modo de comunicación es decir, la forma en la cual van a hablar entre ellos. Existen tres modos posibles:

#### 2.1.6.1. Comunicación Simplex

Por la cual la información fluye en una sola dirección. La confirmación de recepción del mensaje no es posible en este modo. Ejemplo de este modo son la radio y la televisión.

### 2.1.6.2. Comunicación half-duplex

Por la cual la información fluye en ambas direcciones; primero uno de los dispositivos transmite. Al término, el otro responde. Responde. Un ejemplo de esto es el telefax, en donde la comunicación debe ser primero establecida antes de transmitir el mensaje. Este es el modo de comunicación preferido para el campo.

### 2.1.6.3. Comunicación full-duplex

En la Figura 11 se muestra donde se puede transmitir y recibir simultáneamente. Un ejemplo es la comunicación telefónica entre dos personas. Para la comunicación entre dos máquinas sin embargo, se requiere líneas separadas de transmisión y recepción, de lo contrario la información no podría ser decodificada.

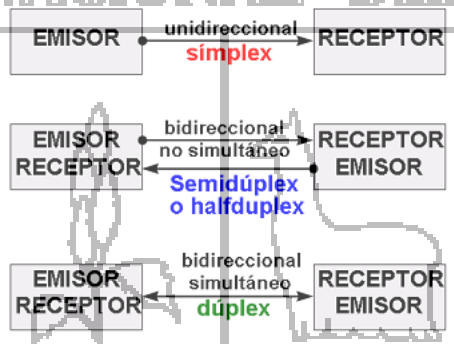



*Figura 11 - Modos de comunicación.*

## 2.2. Marco Conceptual

### 2.2.1. Redes Industriales

Existen diferentes elementos y tecnologías dependiendo de las necesidades y el tipo de la intercomunicación que se desee llevar a cabo. A menudo, existen tecnologías y sistemas con características similares orientados a un mismo rango de aplicaciones, con pocas diferencias excepto en la empresa que fabrica los elementos de comunicación. Esto provoca que no existan estándares completamente adoptados por todas las industrias.

En las empresas se encuentra una serie de equipos y dispositivos dedicados al control de una maquina o una parte cerrada de un proceso. Entre los dispositivos se encuentran los PLC´s, computadores de diseño y gestión, sensores, actuadores, entre otros.

Con las redes industriales se ha podido unir todos estos dispositivos, aumentando el rendimiento proporcionando nuevas posibilidades como se muestra en la Figura 12. Las redes industriales aportan numerosas ventajas a las empresas tales como:

- Visualización y supervisión de todo el proceso productivo.
- Toma de datos del proceso en tiempo real.
- Mejora del rendimiento general de todo el proceso
- Comunicación entre la Red corporativa e Industrial.

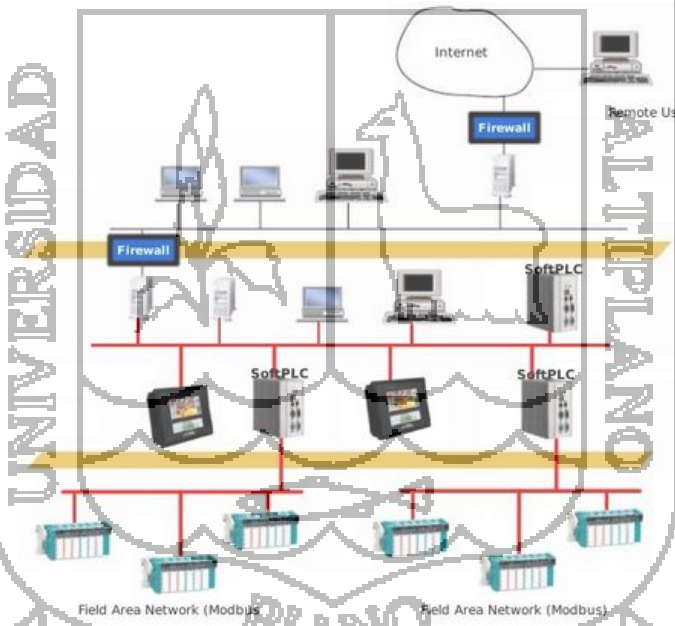



*Figura 12 - Red Industrial Convencional.*

Las necesidades primordiales del usuario común de una red industrial son:

- Reducción de la programación: Evitar el manejo de datos por el PLC o PC, evitar la programación en nodos existentes al añadir nuevos nodos.

- Aumentar las prestaciones del sistema: Determinismo, Efectividad del ancho de banda.

- Reducción del cableado: Control, programación y diagnosis sobre la misma red.

- Soluciones escalables: Elección del controlador adecuado para el control, no para el manejo de datos. Añadir o eliminar dispositivos sin influir en otros dispositivos del sistema.

- Reducción de los tiempos de paro: Diagnóstico de los dispositivos, información predictiva.

**2.2.1.1. Estructura Jerárquica de las Comunicaciones Industriales**

En una red industrial coexistirán equipos y dispositivos de todo tipo, los cuales suelen agruparse jerárquicamente para establecer conexiones lo más adecuadas a cada área. La integración de los diferentes equipos y dispositivos existentes se divide en las tareas entre grupos de procesadores con una organización jerárquica. Así, dependiendo de la función y el tipo de conexiones, se suelen distinguir en cuatro niveles en una red industrial como se muestra en la Figura 13.

- Nivel Gestión: Es el nivel más elevado y se encarga de integrar los niveles siguientes en una estructura de fábrica, e incluso de múltiples factorías. Las máquinas aquí conectadas suelen ser estaciones de trabajo que hacen de puente entre el proceso productivo y el área de gestión, formado básicamente por ordenadores tanto a nivel oficina como de ingeniería.
- Nivel Control: Se encarga de enlazar y dirigir las distintas zonas de trabajo. A este nivel se sitúan los autómatas de gama alta y los ordenadores dedicados a diseño, control de calidad, programación son ordenadores con aplicaciones específicas para el control del proceso.
- Nivel de campo y proceso: Se encarga de la integración de pequeños automatismos (autómatas compactos, multiplexores de E/S, controladores PID, etc.) dentro de subredes o "islas". En el nivel más alto de estas redes se suelen encontrar uno o

varios autómatas modulares, actuando como maestros de la red o maestros flotantes. En este nivel se emplean los buses de campo, son todos los componentes inteligentes que intervienen directamente en el proceso.

- Nivel de Entrada/Salida: Es el nivel más próximo al proceso. Aquí están los instrumentos de campo (sensores, actuadores, etc.), encargados de manejar el proceso productivo y tomar las medidas necesarias para la correcta automatización y supervisión, son todos los dispositivos que provocan los movimientos en el proceso productivo.

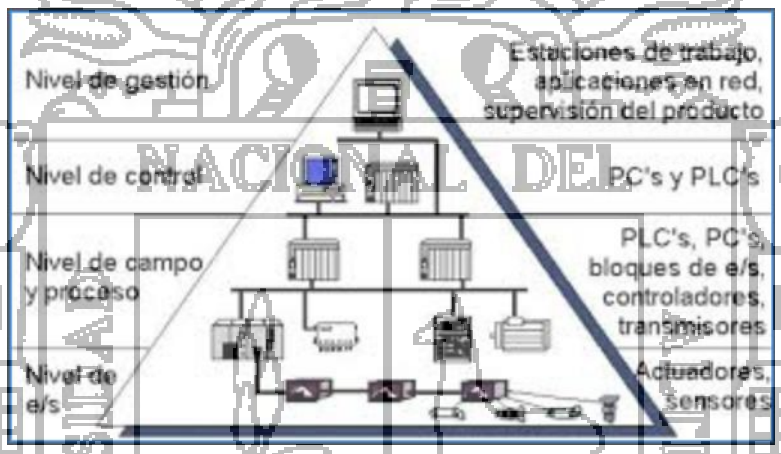



*Figura 13 - Niveles en una red Industrial.*

### 2.2.2. Buses de Campo

#### 2.2.2.1. Buses de Campo y Niveles OSI

El bus de campo constituye el nivel más simple y próximo al proceso dentro de la estructura de comunicaciones industriales. Está basada en procesadores simples y utiliza un protocolo mínimo para gestionar el enlace entre ellos. Los buses de campo más recientes permiten la comunicación con buses jerárquicamente superiores y más potentes, idealmente ver Figura 14, las especificaciones de un bus de campo deberían cubrir los siete niveles OSI, aunque cubren los niveles físico, enlace de datos y Aplicación establecidos en el modelo OSI (Open Systems Interconnection) ver Figura 15.

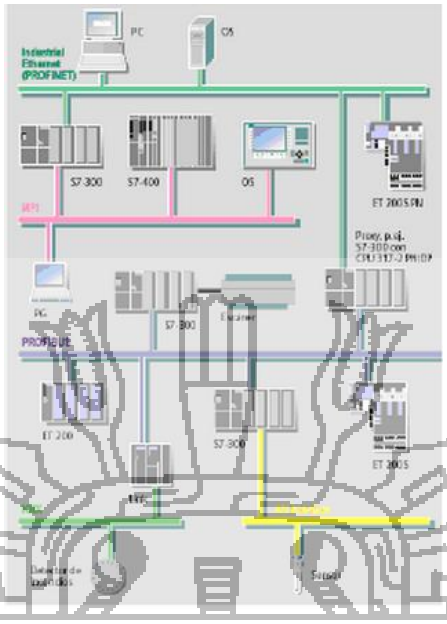


*Figura 14 - Buses de Campo.*

- Nivel Físico: le corresponden características eléctricas y mecánicas que deben garantizar la transmisión del flujo de BITS serie entre equipos. Algunas de estas características son: Voltajes, duración del BIT, forma de codificación, sincronización a nivel de BIT. De igual forma se especifica el tipo de medio físico: cable coaxial, fibra óptica, enlaces electromagnéticos, tipos de conectores, número de hilos, etc…

  La capa física no garantiza una comunicación fiable, es decir, no se realiza ningún tipo de comprobación de errores para asegurar que los datos se reciben correctamente en el destinatario, ya que el nivel físico se encarga únicamente de la transmisión de los datos en señales transmisibles por el medio o viceversa. Ejemplos de esta capa son las normas RS232, RS422 y RS485 empleadas en la industria.

- Nivel Enlace: Garantiza la fiabilidad de la transmisión entre los dos equipos de un enlace de datos (comunicados por el mismo medio físico en la red). Esta capa realiza la detección y corrección de errores (recuperación o reenvío de los datos erróneos) así como de regulación y control de flujo de datos (sincronización entre emisor y receptor).

El código de redundancia cíclica (CRC), en este nivel se garantiza una comunicación sin errores entre equipos del enlace de datos, pero si la secuencia de BITS no se interpreta de igual modo en los equipos implicados en la comunicación, estos no se entenderían.

- Nivel de Aplicación: es el dirigido al usuario, apoyándose en las funciones estándar antes mencionadas para crear programas de gestión y presentación. La aplicación suele ser propia de cada fabricante, permitiendo a lo sumo la programación en un lenguaje estándar, es el dirigido al usuario, apoyándose en las funciones estándar definidas en el nivel de enlace. En este nivel se define el significad de los datos (no hay un nivel de aplicación estándar para buses de campo).

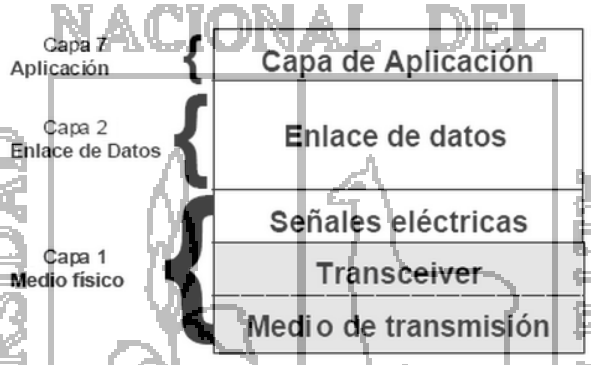



*Figura 15 - Niveles OSI.*

**2.2.2.2. Buses Propietarios y Buses Abiertos**

La existencia de un elevado número de buses de campo diferentes se debe a que cada compañía venía utilizando un sistema propio para sus productos, aunque en los últimos años se observa una cierta tendencia a utilizar buses comunes.

En los bucles de campo podemos distinguir:

- Buses Propietarios: Son propietarios de una compañía o grupo de compañías, y para utilizarlos es necesario obtener una licencia, que es concedida a la empresa que la disfruta con una serie de condiciones asociadas las cuales son: RIO, DH+, PPI.
- Buses Abiertos: Son todo lo contrario las especificaciones son públicas y disponibles a un precio razonable las cuales son: Modbus, Hart, Profibus

Foundation, Control net, Devicenet, Ethernet, AS-I.

#### 2.2.2.3. Infraestructura de una Red

Para formar una red, los distintos componentes se está deben tener algún tipo de enlace. De acuerdo al tipo de enlace que se escoja dependerá la eficiencia de la trasmisión de datos.

Para la elección de un tipo de enlace se tiene que considerar lo siguiente:

- Perdidas de señal de acuerdo al medio físico empleado.
- Interferencias en la señal debido al entorno de trabajo
- Capacidad de transporte de información del medio.
- Longitud máxima de cada segmento del medio.
- Costo.
- Flexibilidad

### 2.2.3. Normas de Interfaz

#### 2.2.3.1. Norma Física RS-232C

La Figura 16 muestra norma RS-232 fue propuesta por la asociación de industrias electrónicas o EIA, a la cual se le han venido haciendo modificaciones hasta llegar al estándar RS-232c, que constituye la tercera versión de la misma. Posteriormente se realizó una versión internacional llamada V.24, la cual no difiere de la RS-232C en casi nada, tal es asi que se habla en general de la interfaz serial RS-232 independientemente de se quiera hablar de V.24 o de RS-232C

El estándar RS-232 establece los niveles de voltaje para la representación de uno y ceros lógicos de comunicación digital, pero cabe notar que para compensar los efectos de

caída de voltaje en la línea, y los efectos del ruido se han establecido diferentes niveles de voltaje tanto para el lado de transmisor (TX) como para el receptor (RX).

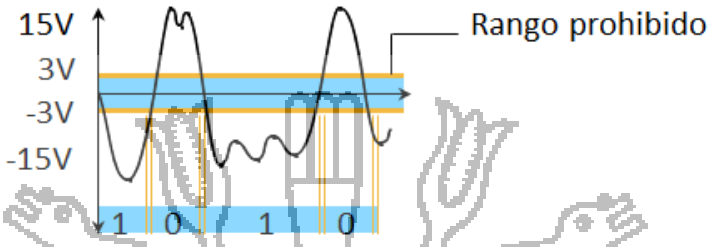



*Figura 16 - Niveles de voltaje en el RS-232 en el TX y en el RX.*

***2.2.3.1.1 Características eléctricas***

Los niveles de voltaje descritos en el estándar son los siguientes:

- Tensión < -3V nivel bajo (1)
- Tensión >+3V nivel alto (0)
- Alimentación +/- 15V
- Velocidad de transmisión < 20 kbit/s
- Longitud del cable hasta 15 Metros

Puede verse que los voltajes del emisor y el receptor son diferentes. Esta definición de los niveles de voltaje compensa las pérdidas de voltaje a través del cable. Las señales son atenuadas y distorsionadas a lo largo del cable. Este efecto es debido en gran parte a la capacidad del cable. En el estándar la capacidad máxima es de 2500 pf. La capacidad de un metro de cable es normalmente de 130 pf. Por lo tanto, la longitud máxima del cable est limitada a unos 17 metros. Sin embargo, esta es una longitud nominal definida en el estándar y es posible llegar hasta los 30 metros con cables de baja capacidad o utilizando velocidades de transmisión bajas y mecanismos de corrección.

#### *2.2.3.1.2 Características mecánicas*

En el estándar no se hace referencia al tipo de conector que debe usarse. Sin embargo los conectores más comunes son el DB-25 (25 pines) y el DB-9 (9 pines). El conector hembra debe estar asociado con el DCE y el macho con el DTE. Este estándar solo especifica las características mecánicas y eléctricas, mientras que la parte de funcionamiento se deja a cargo del usuario. El conector más usado es el DB9, en la siguiente tabla se muestra el tipo de señal y la función que desempeña en el estándar según la asignación de pines en el conector DB – 9 como se muestra en la Figura 17.

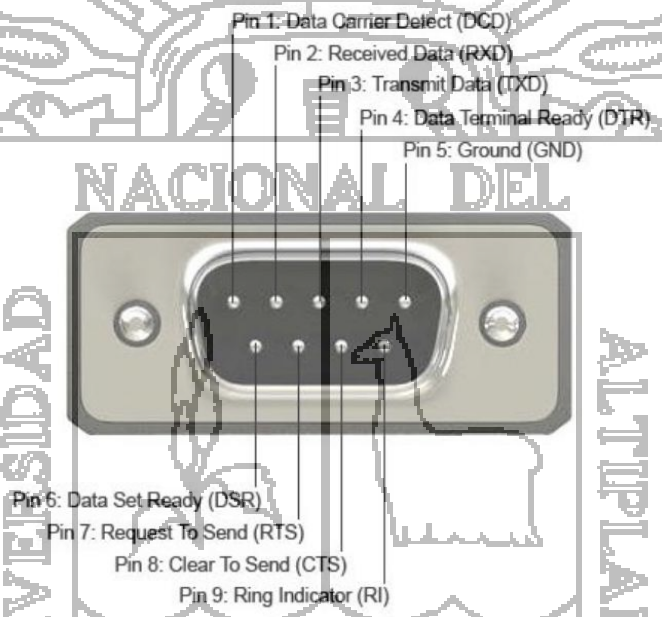



*Figura 17 - Conector DB-9 usando el estandar RS-232.*

| PIN | Abreviatura | Nombre de la señal | Tipo de señal |
|---|---|---|---|
| 1. | CD | Carrier detect | Control |
| 2. | RD | Received dta | Dato |
| 3. | TD | Transmitted data | Dato |
| 4. | DTR | Data terminal ready | Control |
| 5. | GND | Signal ground | Referencia |
| 6. | DSR | Data set ready | Control |
| 7. | RTS | Request to send | Control |
| 8. | CTS | Clear to send | Control |
| 9. | RI | Rind indicator | Control |

*Tabla 1 - Señales eléctricas en el conector DB-9.*

### 2.2.3.2. Norma Física RS-422

Cubre solamente los requerimientos eléctricos y físicos para la transmisión. Usa señales diferenciales y simétricas que permite altas velocidades de trasmisión de hasta 10Mbis/s. en el extremo final de recepción, la diferencia entre los niveles de voltaje es usada para decodificar las señales; la mayor diferencia positiva correspondiente "0" y la menor al "1". La ventaja está en que si un campo extremo actúa sobre la línea, ambas señales son influenciadas al mismo tiempo. La diferencia en la señal se mantiene con excepción del ruido individual de cada línea, sustancialmente igual. De esa manera es posible, tender líneas más largas que para la interfaz RS-232C. Además, desde que los efectos de la interferencia son restringidos son posibles velocidades mayores de transmisión.

Debido a las líneas diferenciales y la disponibilidad de drivers apropiados, esta interfaz es aplicable no solamente para caminos de transmisión extensos, sino también para estructuras de buses seriales, a pesar de ser concebido principalmente como sistema de punto a punto, se pueden operar hasta 16 dispositivos con un solo transmisor ver Figura 18.

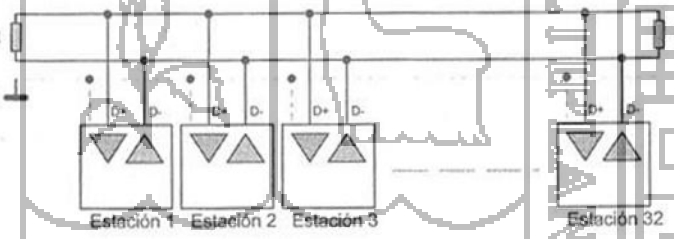



*Figura 18 - Configuración esquemática de una red en la norma RS-422.*

También sucede en la mayoría de redes, se deberán colocar resistencias terminadoras (Rt) en los extremos del canal de comunicación para mantener en todo momento la impedancia en la línea y que es del orden de los 120 ohmios.

### 2.2.3.3. Norma Física RS-485

La norma física es una transmisión diferencial, el voltaje producido por el driver aparece como diferencia entre las tensiones existentes en las líneas A y B Típicamente se produce tensiones 2 y 6V entre las terminales A y B de salida dispone de un habilitador (enable), las salidas se encuentran en alta impedancia (desconectado de la línea).

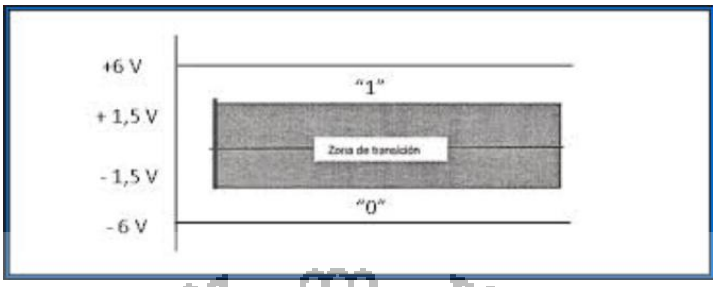



*Figura 19: Señales eléctricas por la interface RS-485.*

Las señales que utiliza esta norma son las siguientes

| Señal | Definición |
|---|---|
| A o D+ | Señal de emisión /recepción no invertida transmitida al canal de comunicación |
| B o D- | Señal de emisión/recepción invertida al canal de comunicación |
| FG | Masa de protección |

*Tabla 2 - Señales eléctricas por la interface RS-485.*

***2.2.3.3.1. Estandar RS-485C***

El estándar RS-485C permite la trasmisión diferencial en redes multipunto, y su uso está ampliamente extendido en ámbitos industriales. Sus principales características se definen a continuación.

- Hasta 32 emisores receptores en la misma línea (ampliable dependiendo del tipo de drivers empleados).
- Estado lógico 1 = diferencia de potencial <200mV
- Estado lógico 0 = diferencia de potencial >200mV
- El voltaje en modo común puede variar en el rango de +12V a -7v
- Longitud del cable de hasta 1200m.

- Velocidades de transmisión de hasta 10Mbps, dependiendo de la longitud del cable. Para velocidades elevadas se requiere el empleo de terminadores en la línea.

### *2.2.3.3.2. Consideraciones de velocidad en RS-485C*

En distancias cortas, la influencia de la línea se puede despreciar, influyendo solo el tiempo de subida y bajada del pulso. El estándar toma 10 Mbps, pero hay chips hoy en día que alcanzan los 25 Mbps, a partir de 10 m se puede calcular como:

$$V*L<10^8$$

Donde V es la velocidad en Mbps y L la longitud del cable en metros. Por ejemplo para 100m, tenemos un máximo de 1Mbps, para cables de mucha longitud influyen las perdidas debidas a la resistencia del cable (100 ohm/Km para cable de 0.6mm) y su capacidad:

| Velocidad | 1200bd | 2400bd | 4800bd | 9600bd | 19200bd | 38400bd | 57600bd | 115200bd |
|---|---|---|---|---|---|---|---|---|
| Capacidad del cable | 250nF | 120nF | 60nF | 30nF | 15nF | 750pF | 500pF | 250pF |

*Tabla 3 - Velocidades en RS-485C.*

La velocidad de transmisión en dispositivos en modo diferencial balanceado está limitada por la longitud de cable existente entre los mismos ver Figura 20.

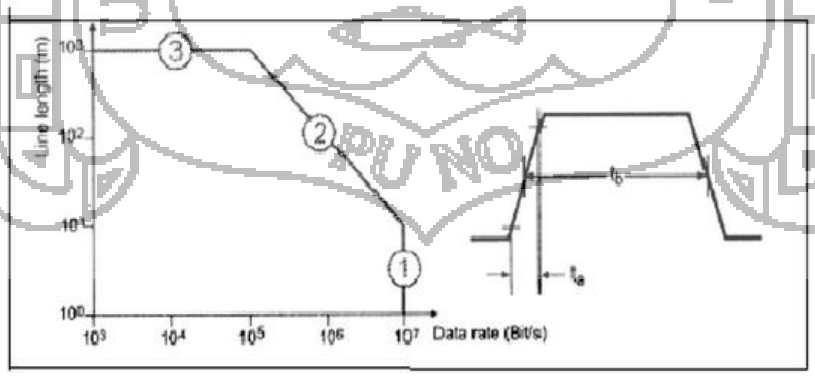



*Figura 20 - Relación de velocidad de transmisión-longitud en modo diferencial balanceado.*

- En la zona 1: la velocidad está limitada por el tiempo de subida de la señal de salida del generador la velocidad de transmisión puede alcanzar los 10 Mbps con longitudes de entre 1 y 10 m.
- En la zona 2: la velocidad está limitada por la distorsión causada por pérdidas en la línea de transmisión (como por ejemplo el efecto pelicular). Las velocidades van de 10 Mbps a 100 Kbps con longitudes de 10 m a 1200 m.
- En la zona 3: la atenuación más importante es la producida por la resistencia óhmica del cable (un cable de par trenzado de 1000 m y 0.6 mm de diámetro tiene una resistencia de aproximadamente 100 Ohmios).

***2.2.3.3.3. Conexionado de redes en RS-485C***

El estándar RS-485 se puede emplear para conexiones de tipo punto a punto, o bien para conexiones en bus de tipo multipunto con hasta 32 emisores receptores. En las figuras siguientes se muestran los dos tipos de conexión multipunto empleados con RS-485C.

- Conexión a 2 hilos: aprovechando la capacidad de los drivers de tipo RS-485C de permanecer en modo triestado, es posible emplear un único par de hilos que será común a todos los driver de transmisión y de recepción es recomendable conectar las líneas de masa para evitar tensiones en modo común elevadas y conseguir una mayor inmunidad al ruido. Por supuesto, tan solo uno de los dispositivos podrá estar transmitiendo (línea enable habilitada), mientras el resto permanecen a la escucha. El modo de acceso más común en estos casos es de tipo maestro- esclavo; uno de los dispositivos hace las veces de maestro y va consultando cíclicamente el resto (esclavos), que tan solo podrán transmitir a petición del maestro. En este caso la comunicación es de tipo semiduplex, por tratarse de un medio compartido ver Figura 21.

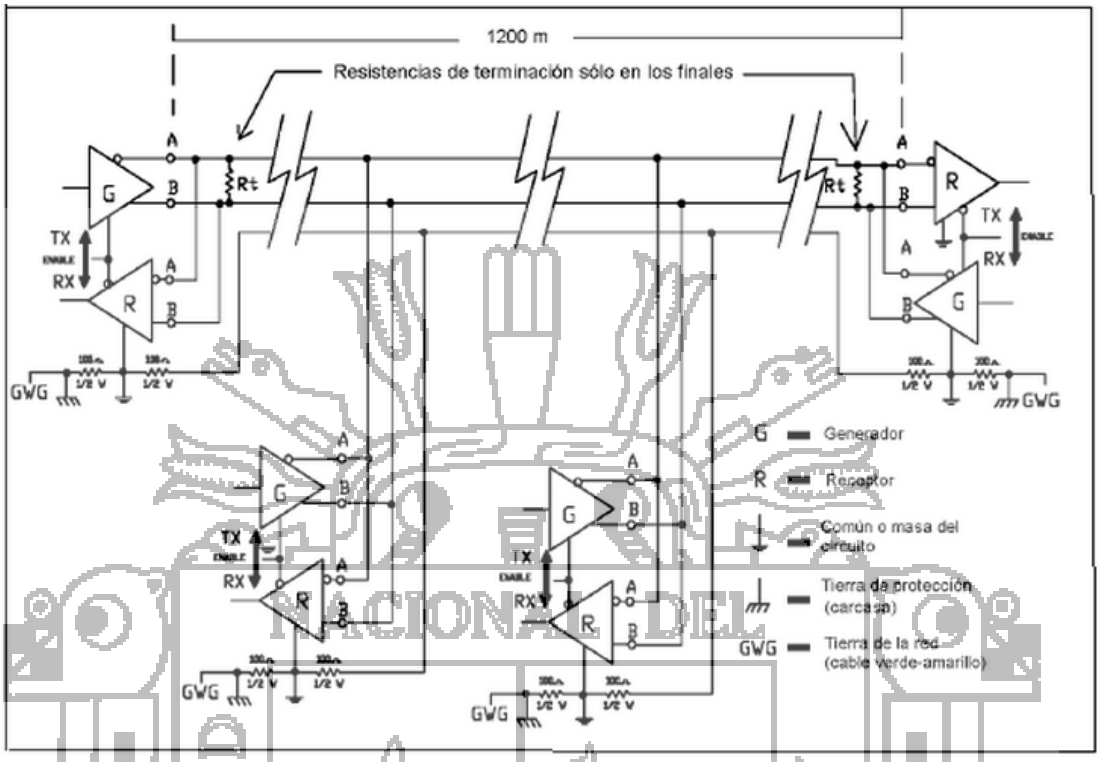



*Figura 21 - Conexión de dos hilos RS-485C.*

- Conexión a 4 hilos: Es una configuración típica para sistemas maestros esclavo, en la que se emplean dos hilos para el driver de transmisión del maestro, a los que se conectaran todos los de recepción del resto de dispositivos (esclavos). Un segundo par de hilos se emplean para conectar el driver de recepción del maestro, a los que se conectan todos los drivers de emisión de esclavos. De esta manera, u empleando la mecánica de comunicación antes descrita, el maestro puede mantener una comunicación en modo dúplex completo con cada uno de los esclavos secuencialmente (en resto de drivers de transmisión deberán estar en alta impedancia) ver Figura 22.

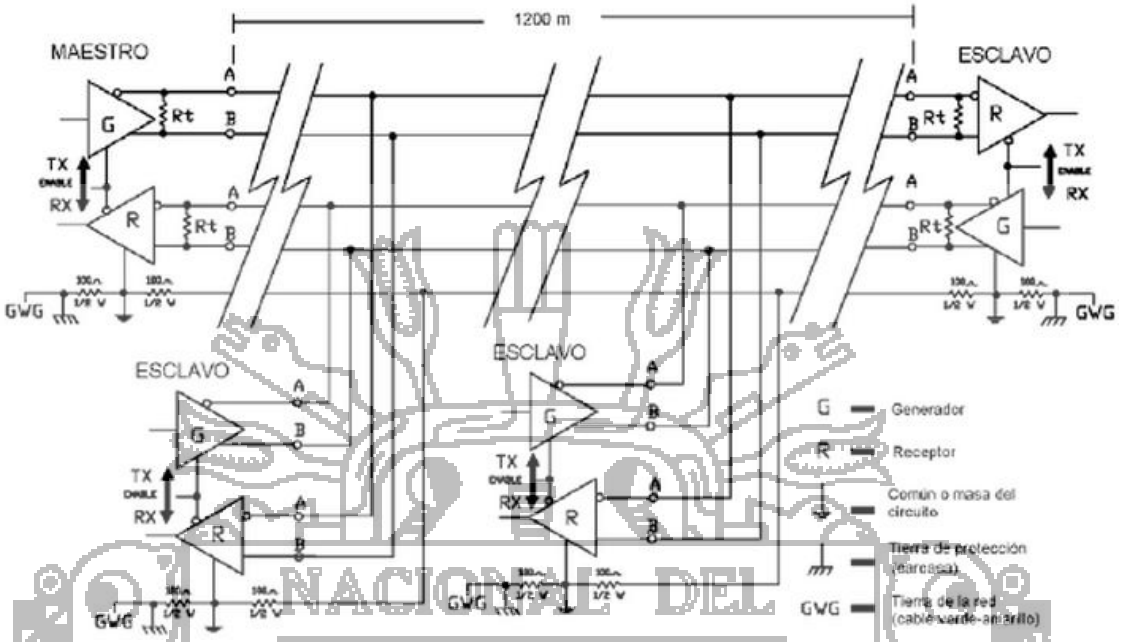



*Figura 22: Conexión de cuatro hilos RS-485.*

### *2.2.3.3.4. Control de triestado en drivers RS-485C*

El control de la patilla triestado en los drivers conectados a la red RS-485C es fundamental para controlar el tiempo que dichos dispositivos se encuentran transmitiendo (el resto del tiempo deben permanecer en triestado). Es común aplicar la misma señal de habilitación al driver de emisión y al de recepción de cada nodo, de modo que dicho nodo no reciba los datos cuando está transmitiendo. En ocasiones el receptor se deja siempre habilitado para recibir eco de todas las transmisiones, existen dos formas de controlar la señal de habilitación en los drivers de transmisión ver Figura 23:

- Empleando la patilla de control RTS de la UART que se encarga de las transmisiones en el nodo conectado a la red. Para ello la señal RTS se debe activar un instante antes de iniciar la transmisión para habilitar el driver, y desactivarse justo después de terminar la misma para liberar el bus.

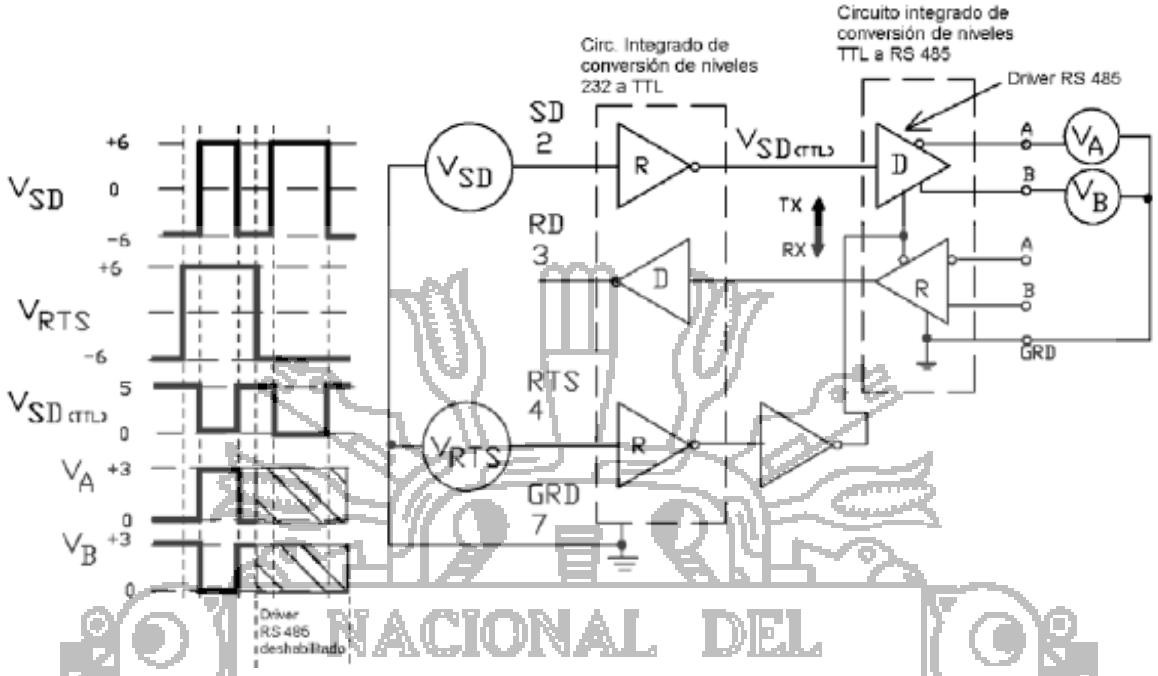



*Figura 23 - Control RTS.*

- Realizar el control de habilitación en modo automático durante el tiempo que dure la transmisión de datos. Para ello ha de emplearse un circuito que sea capaz de detectar el flanco descendente del bit de inicio para habilitar el driver de transmisión dicho circuito estar temporizado para generar un pulso que mantenga la línea activa durante toda la transmisión. Típicamente se producirá un retardo igual a la duración de un carácter tras la transmisión del último bit como se muestra en la Figura 24.

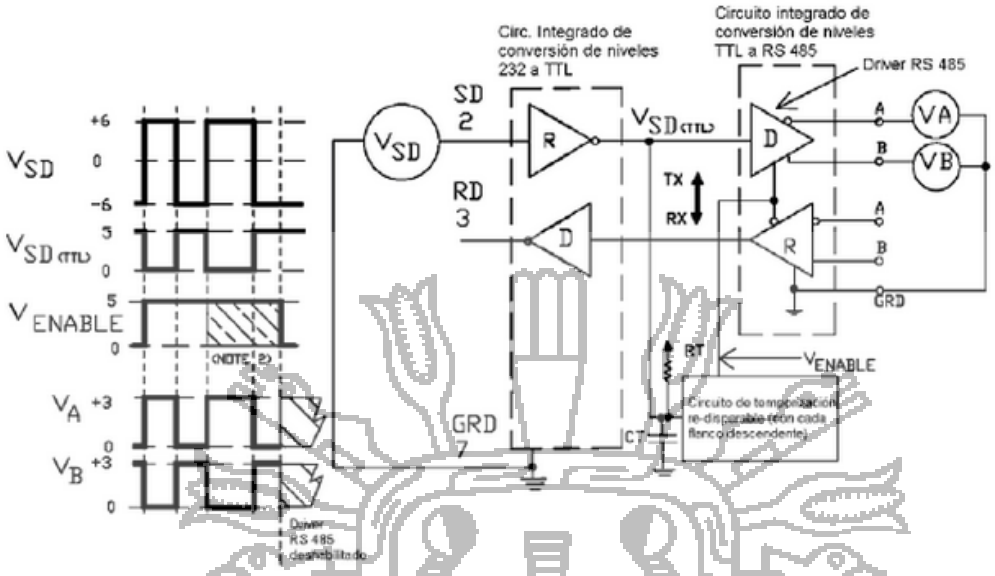



*Figura 24 - Control automático por temporización.*

***2.2.3.3.5. Terminadores***

Los terminadores se emplean para ajustar la impedancia de un nodo a la impedancia de la línea de transmisión que se está utilizando. Si dichas impedancias difieren, la señal transmitida no es totalmente absorbida por la carga y una parte es reflejada en la línea. Si las impedancias de la fuente, la línea de transmisión y el receptor son iguales estas reflexiones son eliminadas. El uso de terminadores tiene el inconveniente de aumentar la complejidad de la instalación y modificar los valores requeridos para resistencias de PULL – UP y PULL-DOWN. La decisión del empleo de terminadores se suele basar en la velocidad y longitud de cable utilizados en el sistema.

Hay muchos métodos para terminar líneas de transmisión de datos. El más frecuente se denomina terminación en paralelo, y consiste en añadir una resistencia en paralelo con las líneas de transmisión "A" y "B"con un valor igual al de la impedancia característica del cable, que es independiente de su longitud (normalmente 120 OHMIOS en cables de par trenzado). Se colocan dos terminadores, uno en cada final de la línea. El problema de este tipo de terminador es que añade una carga en continua elevada, que puede sobrecargar convertidores RS232C – RS485 que se alimenten del propio puerto del ordenador. Para ello se puede colocar un pequeño condensador en serie con el terminador, que elimina el efecto

De carga en continua y en alterna, pero presentan el inconveniente de que el valor del condensador depende mucho de las propiedades de la instalación. En la Figura 25 se muestra los terminadores acoplados en continua y en alterna como.

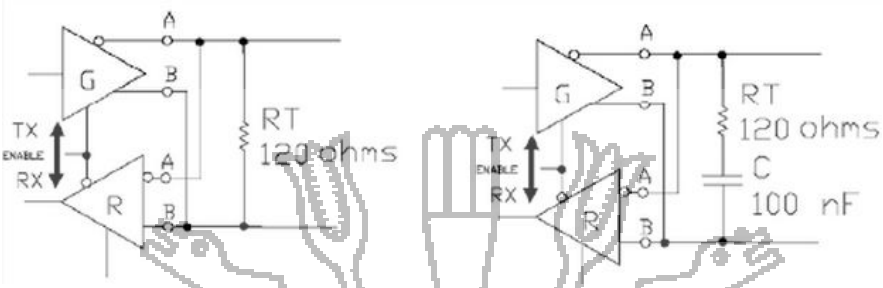



*Figura 25 - Terminadores en DC y AC.*

***2.2.3.3.6. Resistencias de bías***

Cuando en una red RS 485C todos los nodos están a la escucha (todos los transmisores en modo de alta impedancia) el estado de las líneas de transmisiones está indeterminado. Para evitar esto es conveniente añadir resistencias de Bias ( pull-up y pull-down) que fuercen el estado de las líneas a un nivel lógico estable cuando se produzca esta condición. Las resistencias de Bias consisten en una resistencia de pull – up en la línea B (a + 5 V) y una resistencia de pull – down (a masa) en la línea A. El valor de las resistencias de Bias depende el número de nodos y del valor de las resistencias de terminación. El objetivo es conseguir una corriente de Bias en continua suficiente para generar una tensión de un mínimo de 200 mV entre las líneas de datos B y A. Estas resistencias se pueden colocar en cualquier punto de la red o incluso se pueden repartir en varios nodos ver Figura 26.

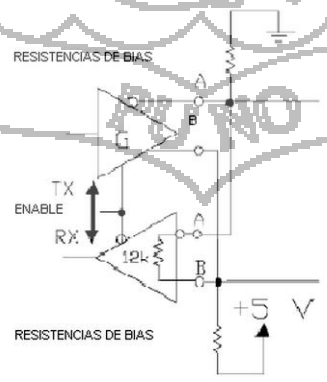



*Figura 26 - Conexión de las resistencias de vías.*

### 2.2.4. Protocolo Modbus

El protocolo Modbus es una estructura de mensajería desarrollado por Modicon (actualmente Schneider Electric) en 1979, es un estándar de hecho verdaderamente abierto y el protocolo de red más utilizado en el ambiente de la industria. El protocolo modbus proporciona un método industrial estándar que los dispositivos modbus utilizan para analizar los mensajes.

Es un protocolo de mensaje en la capa aplicación, posicionado en el nivel 7 del modelo OSI que sirve para establecer la comunicación entre dispositivos inteligentes que tiene un acceso al medio maestro - esclavo/cliente - servidor entre dispositivos conectados en diferentes tipos de buses o redes. Utiliza la transmisión serial que intercambia peticiones MODBUS entre un equipo maestro y uno o varios esclavos utilizando modo de comunicación half-duplex o semiduplex. Por la norma física EIA/TIA-232, EIA/TIA-422, EIA/TIA-485C el más frecuente ver Figura 27.

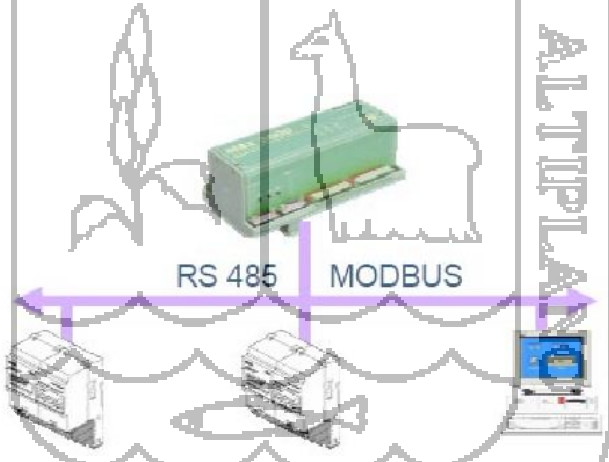



*Figura 27: Norma física* EIA/TIA-485C.

Un sistema tipo Maestro-Esclavo, el nodo "maestro" inicia la pregunta el nodos "esclavo" procesa la respuesta ya sea de lectura/escritura en esclavos, los nodos son de 64 incluyendo el maestro. Los nodos esclavos no transmiten data sin una petición del nodo maestro y no se comunica con otros esclavos. Los equipos se encuentran en una trocal de cable constituido de 3 conductores, 2 de ellos conductores forman un par de cables balanceado, en el cual los datos se transmiten bidireccionalmente y el tercer cable la línea

común cada equipo puede conectarse:

La transferencia de datos se organizó en términos de registros de 16 bits (formato de entero) o como información de estado en términos de bytes de datos. Con los años el protocolo fue extendido y ha ido adoptado por otros fabricantes también. Se han añadido nuevos tipos de datos, en especial para lograr una resolución más alta para los valores que se transmiten. El protocolo modbus es de hecho un protocolo de maestro único, el maestro controla la transmisión completa y monitoreo si se produce posibles tiempos de espera sobrepasados (no hay respuesta desde el dispositivo direccionado) ver Figura 28.

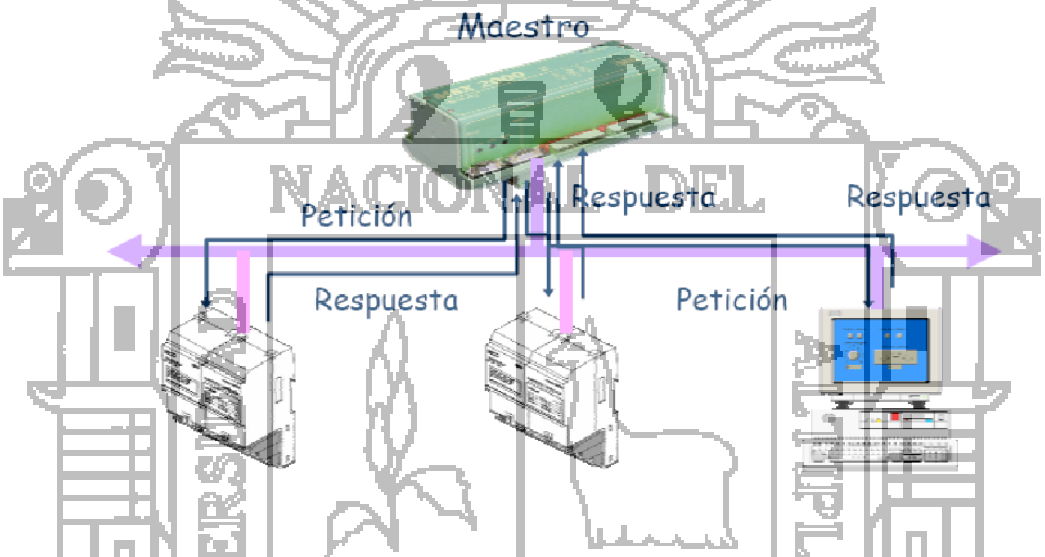



*Figura 28 - Acceso al medio maestro - esclavo/cliente – servidor.*

- Desarrollado por Modicon para comunicacion entre PLC's.
- Control de acceso al medio tipo Maestro/Esclavo.
- A cada esclavo se le asigna una direccion fija y unica en el rango de 1 a 247.
- Debido a su simplicidad y especificacion abierta, actualmente es ampliamente utilizado por diferentes fabricantes.
- Entre los dispositivos que lo utilizan podemos mencionar: PLC, HMI, RTU, Drives, sensores y actuadores remotos.
- El protocolo establece como los mensajes se intercambian en forma ordenada y la detección de errores.
- Ofrece servicios codificados mediante códigos de función.

### 2.2.4.1. Formato del Mensaje

La Figura 29 muestra el formato general para el mensaje modbus esta mostrado.

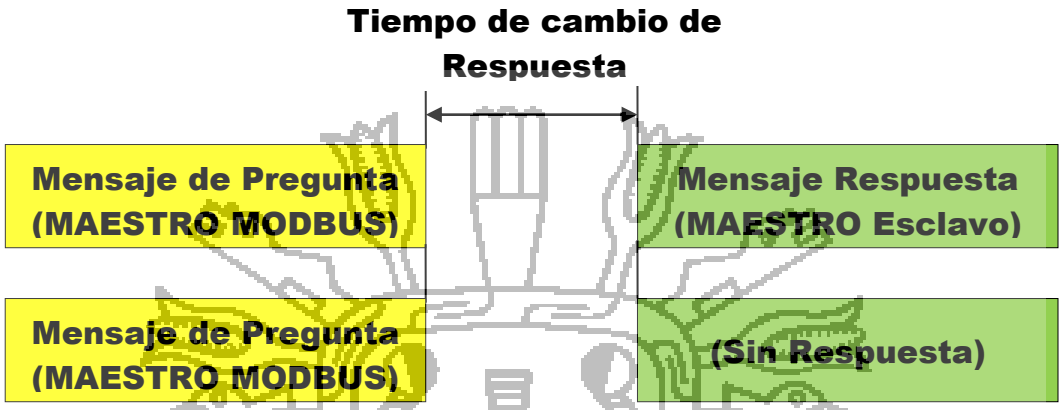



*Figura 29: Formato de trama Modbus.*

Una distinción está hecha entre dos dispositivos de comunicación. El dispositivo el cual inicia la transferencia, es llamado el maestro y el otro dispositivo el esclavo, el dispositivo maestro empieza la transferncia enviando una pregunta o emitiendo un mensaje de petición. Un esclavo completa esta transferencia enviando un mensaje de respuesta, si el esclavo envía ninguna respuesta es llamado respuesta de difusión sin respuesta.

### 2.2.4.2. Modos de Transmisión en serie

El modo de transmisión define el contenido de bits de los bytes del mensaje de transmisión a lo largo la red, y como la información del mensaje va a ser empaquetada en la secuencia de mensajes y decodificada el protocolo modbus posee dos modos esenciales de funcionamiento.

### 2.2.4.3. Modo de transmisión ASCII

Este modo particular se da cuando los controladores están configurados para comunicarse con la red MODbus utilizando la codificación estándar ASCII (American

Standard Code for Information Interchange). Para este modo cada byte de 8 bits en un mensaje es enviado como dos caracteres ASCII.

La principal ventaja de este modo radica en que los intervalos de tiempo por encima de un segundo, que se dan entre dos caracteres no ocasionan ningún error.

El formato para cada byte en el modo ASCII es el siguiente:

- Sistema de codificación:

  Hexadecimal, caracteres ASCII 0-9, A-F un carácter hexadecimal contenido en cada carácter ASCII del mensaje.
- Bits por byte:

  1 bit de arranque.

  7 bits de datos, el menos significativo se envía primero.

  1 bit para paridad Par o Impar; ningún bit para No paridad.

  1 bit de paro si se usa paridad; 2 bits si no se usa paridad.
- Campo de Chequeo de error:

  Comprobación Longitudinal Redundante (LRC).

***2.2.4.3.1. Estructura ASCII***

En el modo ASCII los mensaje comienzan con dos puntos (:) el carácter ACSII 3 A hexadecimal, y finaliza con un retorno de línea (Carriage return – line feed) (CRLF) par, que corresponde a un carácter ASCII 0D y 0A hexadecimal. Los siguientes caracteres transmitidos por todos los otros campos son hexadecimales 0-9, A-F. El monitoreo de los dispositivos en la red MODbus se realiza continuamente por el carácter dos puntos.

Cuando uno es recibido, cada dispositivo decodifica el siguiente campo (el campo de dirección) y lo encuentra afuera si es de dispositivo direccionado. Los intervalos por encima de un segundo pueden transcurrir entre caracteres dentro del mensaje. Sí ocurre un gran intervalo, el dispositivo que recepciona asume que ha ocurrido un error. Una

estructura típica del mensaje[2] en el modo ASCII puede verse a continuación en la Figura 30.

| INICIO | DIRECCION | FUNCION | DATO | LRC | FINAL |
|---|---|---|---|---|---|
| 1 Caracter | 2 Caracteres | 2 Caracteres | 0 hasta 2x252 caracteres | 2 caracteres | 2 Caracteres CR,LF |

*Figura 30 - Estructura de un mensaje Modbus en modo ASCII.*

**2.2.4.4. Modo de transmisión RTU**

Cuando los controladores se configuran en el modo[3] RTU (Remote Terminal Unit) cada byte de 8 bits en un mensaje contiene dos caracteres hexadecimales de 4 bits. La mayor ventaja de este modo radica en el hecho de que la mayor densidad de caracteres permite un mejor rendimiento en el modo ASCII para casos donde se tiene la misma velocidad de baudios. Cada mensaje debe ser transmitido en un flujo continuo como se muestra la Figura 31.

El formato para cada byte en el modo RTU es el siguiente:

- Sistema de codificación:

  8 Bits en binario, hexadecimal 0-9, A-F

  2 Caracteres hexadecimales contenidos en cada campo de 8 bits del mensaje

- Bits por byte:

  1 bit de start

  8 bits de datos, el menos significativo se envia primero

  1 bit para paridad par o impar, no se envia BIT si no hay paridad

  1 bit de stop, si se usa paridad; 2 bits si no hay paridad

[2] Las especificación para la estructura de mensaje tomadas de MODBUS over Serial Line Specification y Imlemetation guide V1.0 Pag. 17.

[3] Para el modo de transmisión RTU su especificación fueron tomadas de MODBUS over Serial Line Specification y Implementation guide V1.0 Pag. 13

- Campo de chequeo de error:

  Chequeo de redundancia cíclica (CRC)

| DIRECCION | FUNCION | DATO | CRC |
|---|---|---|---|
| 1 byte | 1 byte | 0 hasta 252 bytes | 2 bytes CRC Low, CRC high |

*Figura 31 - Modo de transmisión RTU.*

***2.2.4.4.1. Estructura RTU***

Los mensajes del modo RTU comienzan con un intervalo de silencio por lo menos 3,5 de tiempos de caracteres implementado como un múltiplo de tempos de caracteres a la velocidad de transmisión para todos los campos son valores hexadecimales 0-9, A-F. Un dispositivo de red monitoreo continuamente la red, incluyendo los intervalos de silencio, y cuando el primer campo es recibido (la dirección) después de un intervalo de silencio de al menos 3,5 veces el tiempo de carácter, el dispositivo lo decodifica para determinar si es el dispositivo direccionado, tras el último carácter de transmisión, un intervalo de 3,5 veces el tiempo de carácter marca el final del mensaje y un nuevo mensaje puede comenzar después de este intervalo.

Todo el mensaje debe ser transmitido como un flujo continuo. Si un intervalo de silencio de más de 1,5 veces un carácter se produce antes de la finalización del formato (no un flujo continuo), el silencio de recepción vacía el mensaje incompleto y asume que el siguiente byte será el campo e dirección de un nuevo mensaje. En forma similar, si un nuevo mensaje comienza antes que 3,5 veces el carácter siguiente a un mensaje anterior, esto generará un error, ya que el valor en el campo final CRC no será válido para los mensajes combinados ver Figura 32.

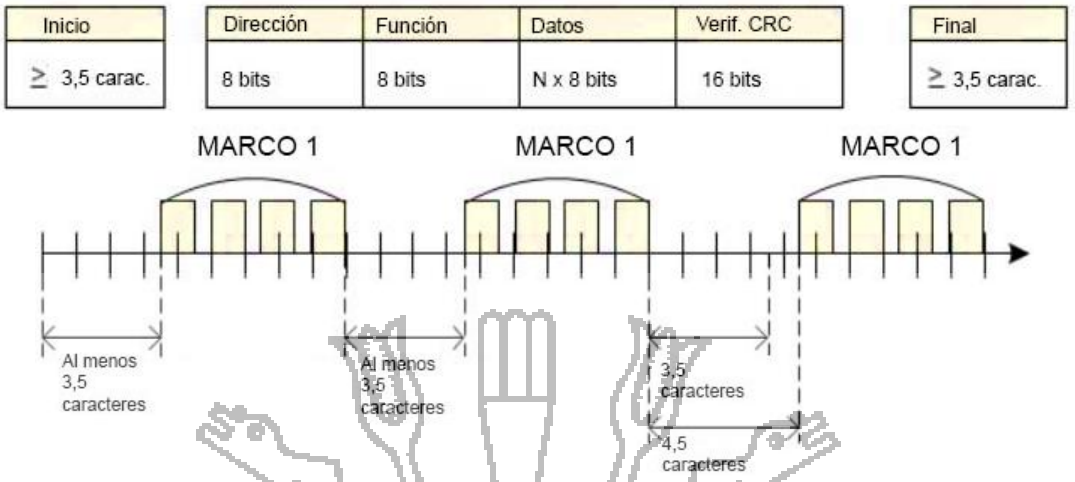



*Figura 32 - Estructura típica del mensaje en RTU*

### 2.2.4.5. Campo del Mensaje

En campo de mensajes[4], El protocolo Modbus aplica todavía el principio de maestro esclavo aunque el método de comunicación de la red sea par a par si un controlador origina un mensaje, lo hace como un maestro y espera una respuesta de un del esclavo. De la misma forma, cuando el esclavo recibe un mensaje, construye una respuesta y regresa el mensaje al maestro ver Figura 33.

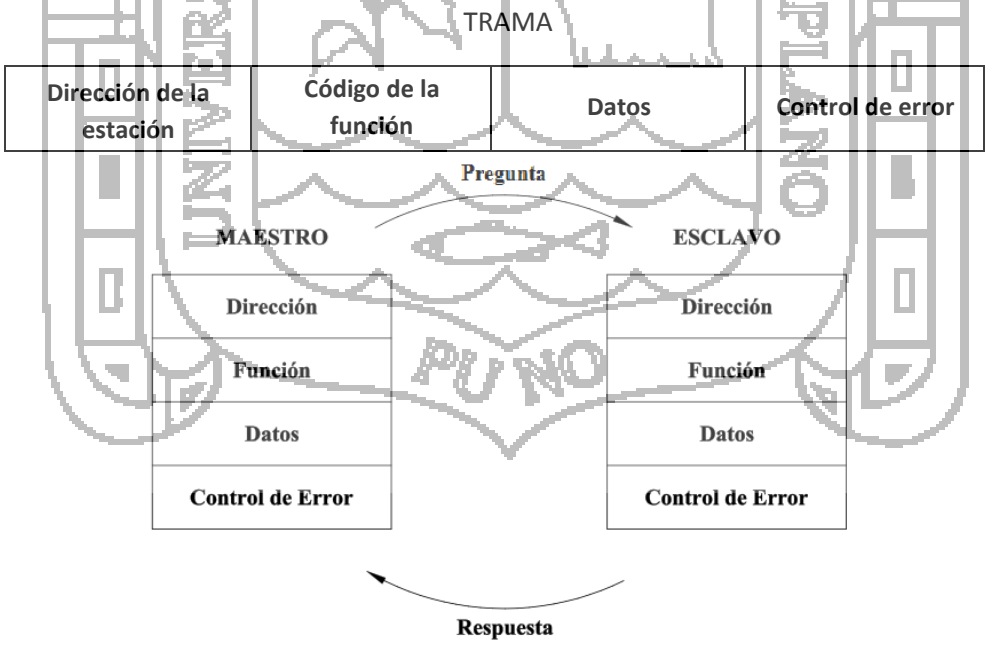



*Figura 33* - Ciclo de pregunta y respuesta maestro – esclavo.

[4] Las especificación para la estructura de mensaje en modo RTU tomadas de MODBUS over Serial Line Specification y Imlemetation guide V1.0 Pag. 13.

- **La pregunta:** El maestro indica al esclavo direccionado el tipo de acción a realizar. Los bytes de datos contienen cualquier información adicional que el esclavo necesitará para llevar a cabo el mensaje de pregunta.
- **La Respuesta:** contienen los datos recolectados por el esclavo, tales como valores de registros o estados.

#### *2.2.2.5.1. Campo de Dirección*

El campo dirección de un mensaje contiene dos caracteres (ASCII) u ocho bits (RTU). Las direcciones de esclavo válidas están en el rango de 0 – 247 decimal. Los dispositivos esclavos individuales tienen direcciones asignadas en el rango 1 – 247. Un maestro direcciona un esclavo situando la dirección del esclavo en el campo dirección del mensaje. Cuando el esclavo envía su respuesta, sitúa su propia dirección en este campo dirección de la respuesta para dar a conocer al maestro qué esclavo está respondiendo[5].

Los códigos de función están codificados en un byte la dirección 0 es utilizada como dirección de difusión, la cual todos los dispositivos esclavos reconocen. Cuando el protocolo Modbus es usado en redes de nivel más alto.

#### *2.2.2.5.2. Campo de Función*

El campo de función de una trama de mensaje contiene dos caracteres (ASCII) u ocho bits (RTU). El campo de código de función de una trama de mensaje contiene ocho bits. Los códigos válidos están en el rango de 1- 255 decimal. De los cuales, solo algunos códigos pueden ser utilizados. Cuando un mensaje es enviado desde un maestro a un dispositivo esclavo, el campo de código de función[5] indica al esclavo qué tipo de acción ha de ejecutar.

Cuando el esclavo responde al maestro, utiliza el campo de código de función para indicar, ya sea, una respuesta normal (libre de error) o que algún tipo de error ha tenido lugar (denominado respuesta de excepción). Para una respuesta normal, el esclavo simplemente replica el código de función original. Para una respuesta de excepción, el esclavo devuelve un código que es equivalente al código de función original con su bit más

[5] Las especificaciones aquí mencionadas fueron tomadas de Modicon Modbus Protocol Referente Guide. PI-MBUS-300 Rev. J. Pag. 19

significativo (MSB) puesto en 1, los códigos de función están codificados en un byte como se muestra la Tabla 5.

| Rango | Uso |
|---|---|
| 0 | No Valido |
| 1-127 | Funciones Validas |
| 128-255 | Respuesta de Excepción |

*Tabla 4* - Codificación de función *Modbus*.

| Código | Acción | Significado |
|---|---|---|
| 01 | Leer Bobinas (0:xxxx) | Obtiene el estado actual ON/OFF de un grupo de bobinas lógicas. |
| 02 | Leer Entradas (1:xxxx) | Obtiene el estado actual ON/OFF de un grupo de entradas lógicas. |
| 03 | Leer Registros (4:xxxx) | Obtiene el valor binario de uno o más registros de almacenamiento. |
| 04 | Leer Registros (3:xxxx) | Obtiene el valor binario de uno o más registros de entrada. |
| 05 | Escribir Bobina (0:xxxx) | Fuerza el estado de una bobina. |
| 06 | Escribir Registro (4:xxxx) | Escribe el valor binario de un registro de almacenamiento. |
| 15 | Escribir Bobinas (0:xxxx) | Fuerza el estado de un grupo de bobinas. |
| 16 | Escribir Registros (4:xxxx) | Escribe el valor binario de un grupo de registros de almacenamiento. |

*Tabla 5- Funciones básicas y códigos de operación Modbus.*

***2.2.2.5.3. Campo Datos***

El campo datos se construye utilizando conjuntos de 2 dígitos hexadecimales, en el rango de 00 á FF hexadecimal. Pueden formarse a partir de un par de caracteres ASCII o desde un carácter RTU, de acuerdo al modo de transmisión serie.

El campo de datos de los mensajes enviados desde un maestro a un esclavo, contiene información adicional que el esclavo debe usar para tomar la acción definida por el código de función. Esto puede incluir partes como direcciones discretas y de registros, la cantidad de partes que han de ser manipuladas y la cantidad de bytes de datos contenidos en el campo.

Por ejemplo, si el maestro solicita a un esclavo leer un grupo de registros sostenidos (código de función 03), el campo de datos especifica el registro de inicio y cuántos registros han de ser leídos. Si el maestro escribe sobre un grupo de registros en el esclavo (código de función 10 hexadecimal), el campo de datos especifica el registro de inicio, cuántos registros se van a escribir, la cantidad de bytes de datos que siguen en el campo de datos y los datos que se deben escribir en los registros.

Si no ocurre error, el campo de datos de una respuesta desde un esclavo al maestro, contiene los datos solicitados. Si ocurre un error, éste campo contiene un código de excepción que la aplicación del maestro puede utilizar para determinar la próxima acción a tomar. Un sistema que tiene señales de interfaz, sensores, actuadores necesita de un controlador (cerebro) para que pueda garantizar el correcto dato Tabla 6.

| Primeros Índices | Tipo de objeto | Tipo | comentarios |
|---|---|---|---|
| Entradas discretas | Un solo bit | Solo lectura | Este tipo de dato puede ser suministrado por I/O del sistema |
| Bobinas | Un solo Bit | Lectura – Escritura | Este tipo de dato puede ser suministrado por una aplicación del programa |
| Registro de entrada | Palabra de 16-bit | Solo lectura | Este tipo de sato puede ser suministrado por I/O del sistema |
| Registros holding | Palabra de 16-bit | Lectura - Escritura | Este tipo de dato puede ser alterado por una aplicación del programa |

*Tabla 6 - Modelo de dato Modbus.*

### *2.2.2.5.4. Comprobación de Error*

Dos tipos de métodos de comprobación de error son utilizados para las redes Modbus Estándar. El contenido del campo de Comprobación de Error depende del método que esté siendo utilizado, Si no ocurre errores relacionados con el código de función requerido por el maestro, el campo de datos de la respuesta del maestro contiene los datos solicitados ver Figura 34.

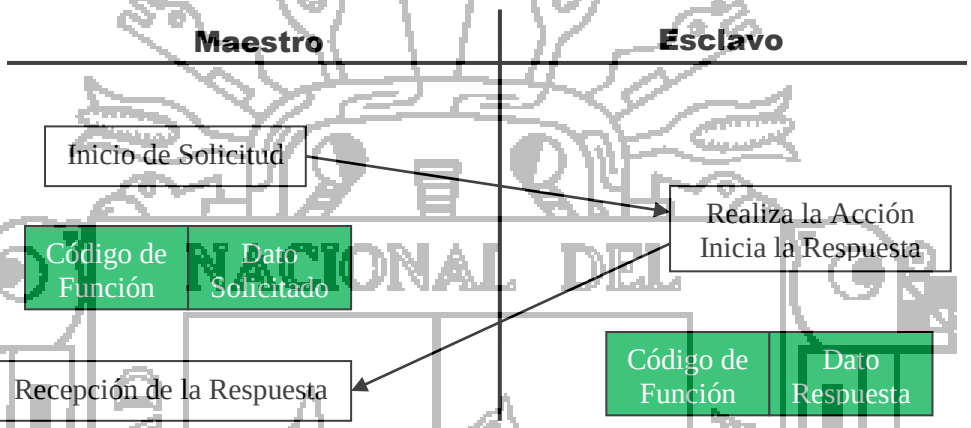



*Figura 34 - Respuesta sin error.*

Sin embargo durante el ciclo consulta respuesta pueden darse diferentes eventos que obligan al esclavo a enviar un mensaje de excepción ver Figura 35.

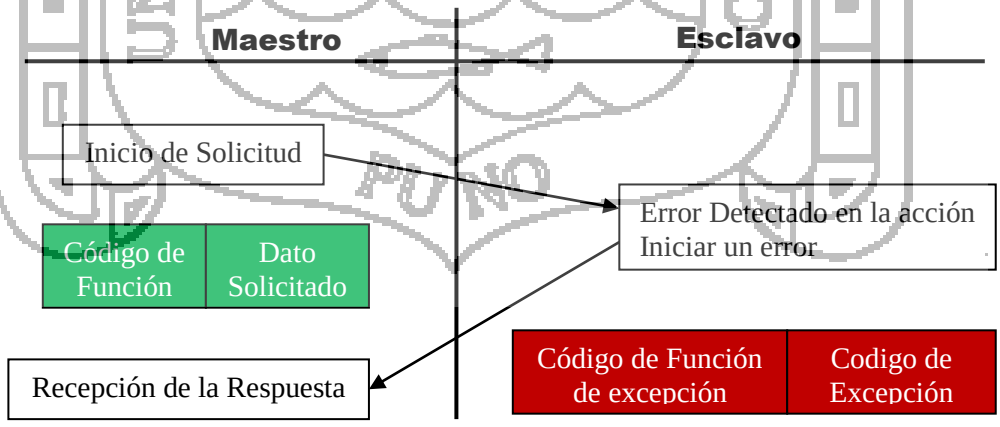



*Figura 35 - Respuesta con error.*

Si hay error el esclavo envía una “respuesta de excepción”, los mensajes de excepción no se envían cuando existen errores de comunicación como son errores de paridad en el LRC o CRC. Si existe uno de estos errores el esclavo no enviará algún mensaje. Un mensaje de excepción se presentará si se dan algunos de los siguientes eventos ver Tabla 7:

- En el mensaje de consulta se le ha solicitado al esclavo que ejecute una función que no soporta.
- Cuando se le solicita un bit o una palabra que no existe.
- Cuando el número de datos solicitado excede el máximo del esclavo.
- Cuando el esclavo tiene dificultades o detecta errores cuando intenta ejecutar una función.
- Cuando el esclavo está ocupado.
- Cuando el esclavo requiere de un tiempo mayor al *timeout* para ejecutar la función solicitada.
- Cuando se detecta error en la ejecución de una función enviada en un programa.
- Cuando se detectan errores en una memoria extendida.

| Código | Nombre | Significado |
|---|---|---|
| 01 | Función ilegal | El código de función recibido en la consulta no es soportada por el esclavo. |
| 02 | Dirección de dato ilegal | La dirección de datos en la consulta no es una dirección válida para el esclavo. |
| 03 | Valor de datos ilegal | Una dirección enviada dentro del campo de datos en la consulta no es una dirección válida para el esclavo. |
| 04 | Falló el dispositivo esclavo | Un error desconocido ocurrió mientras el esclavo intentaba ejecutar la función solicitada en la consulta. |
| 05 | Reconocimiento | El esclavo ha aceptado la consulta y está procesándola, pero requiere de mayor tiempo para ejecutarla. Esta respuesta es enviada para evitar un error por *timeout* en el maestro. |
| 06 | Dispositivo esclavo ocupado | El esclavo está ejecutando un comando de programa que requiere de mucho tiempo. |
| 07 | Reconocimiento negativo | El esclavo no puede ejecutar la función recibida el la consulta. Este código es enviado por una consulta de programa infructuosa que usa los códigos de función 13 o 14 decimal. El maestro debe hacer una consulta de diagnóstico o información de error del esclavo. |
| 08 | Error de paridad en memoria | El esclavo intenta leer una memoria extendida, pero detecta un error de paridad en esta. El maestro puede enviar la consulta nuevamente, pero algún servicio puede necesario en el esclavo. |

*Tabla 7 - Comprobación de Error Modbus.*

- ***ASCII*:** Cuando el modo ASCII es usado, el campo de Comprobación de Error contiene dos caracteres ASCII. Los caracteres de comprobación de error son el resultado de un cálculo de Comprobación de Redundancia Longitudinal (LRC), que es realizado sobre el contenido del mensaje; excluyendo los 'dos puntos' del comienzo y los caracteres 'CRLF' de finalización. Los caracteres LRC son añadidos al mensaje como el último campo que precede a los caracteres CRLF.

  No está ampliamente implantado en procesos industriales principalmente porque

- ***RTU*:** Cuando el modo RTU es usado, el campo de Comprobación de Error contiene un valor de 16 bits implementado como dos bytes de 8 bits. El valor de comprobación de error es el resultado de un cálculo de Comprobación de Redundancia Cíclica (CRC), realizado sobre el contenido del mensaje. El campo CRC es añadido al mensaje como último campo del mensaje. La forma de hacerlo es, añadir primero el byte de orden bajo del campo, seguido del byte de orden alto. El byte de orden alto del CRC es el último byte a enviar en el mensaje.

#### *2.2.2.5.5. Uso del timeout*

Los timeout´s son usados en enlaces seriales para la detección de errores, y para prevenir la pérdida del final del mensaje y las secuencias de mensajes.

Después de enviar un mensaje de pregunta, el maestro debiera esperar aproximadamente 500 milisegundos antes de asumir que el esclavo no respondiera su petición.

### 2.2.4.6. Generación de Error

Las redes series standard Modbus utilizan dos tipos de comprobación de error. La comprobación de paridad (par o impar) puede ser aplicada opcionalmente a cada carácter. La comprobación de la trama (LRC o CRC) es aplicada al mensaje completo. Ambas comprobaciones, de carácter y de trama de mensaje son generadas en el dispositivo maestro y aplicadas a los contenidos del mensaje antes de la transmisión. El dispositivo esclavo comprueba cada carácter y la trama del mensaje completo durante la recepción.

El maestro es configurado por el usuario para aguardar durante un tiempo de espera predeterminado antes de abortar la transacción. Este intervalo es establecido para ser lo suficientemente largo para que cualquier esclavo responda normalmente. Si el esclavo detecta un error de transmisión, el mensaje no será tenido en cuenta. El esclavo no construirá una respuesta para el maestro. Así el tiempo de espera expirará y permite al programa del maestro tratar el error. Observe que un mensaje direccionado a un dispositivo esclavo inexistente también causará un error de tiempo excedido - time out -.

**2.2.4.6.1. Generación de Paridad**

Los usuarios pueden configurar los controladores para Control de paridad Par o Impar, o Sin Control de paridad. Esto determinará cómo será iniciado el bit de paridad en cada carácter. Si se especifica cualquier control de paridad Par o Impar, se contabilizará la cantidad de bits que tienen valor 1 en la porción de datos de cada carácter (siete bits de datos para modo ASCII, u ocho para RTU). Al bit de paridad habrá de darse valor 0 o 1, para que se obtenga finalmente un número par o impar, respectivamente, de bits con valor 1.

Por ejemplo, estos 8 bits de dato forman parte de una trama de carácter RTU:

1100 0101

La cantidad de bits de valor 1 en el dato es cuatro. Si se utiliza Control de Paridad Par, el bit de paridad de la trama debe establecerse a valor 0, haciendo que la cantidad de bits de valor 1 siga siendo un número par (cuatro). Si se utiliza Control de Paridad Impar, el bit de paridad deberá tener valor 1, resultando una cantidad de bits de valor 1, impar (cinco).

Cuando el mensaje es transmitido, el bit de paridad es calculado y aplicado a la trama de cada carácter. El dispositivo receptor cuenta la cantidad de bits de valor 1 y establece un error si no coincide la paridad con la configurada para ese dispositivo (todos

los dispositivos en la red Modbus deben ser configurados para usar el mismo método de Control de paridad).

Obsérvese que la comprobación de paridad sólo detecta si un número *impar* de bits se han alterado en una trama de carácter durante la transmisión. Por ejemplo, si se utiliza control de paridad Impar y dos bits de valor 1 de un carácter que tiene en origen 3 bits con valor 1, han quedado falseados (pasan a valor 0) durante la transmisión, el resultado es todavía un cómputo impar de bits de valor 1 (y por lo tanto el error no es detectado por este método).

Si se especifica control No Paridad, no se transmite bit de paridad y no se hace comprobación de paridad. Se transmite un bit de paro adicional para rellenar la trama de carácter.

**2.2.4.6.2. Generación de *LRC***

En modo ASCII, los mensajes incluyen un campo de comprobación de error que está basado en un método de Comprobación Longitudinal Redundante (LRC). El campo LRC controla el contenido del mensaje, a excepción de los ':' del comienzo y el par CRLF. Es aplicado con independencia de cualquier método de control de paridad utilizado para los caracteres individuales del mensaje.

El campo LRC es un byte, conteniendo un valor binario de ocho bits. El valor LRC es calculado por el dispositivo emisor, que añade el LRC al mensaje. El dispositivo receptor calcula el LRC durante la recepción del mensaje y compara el valor calculado con el valor recibido en el campo LRC. Si los dos valores no son iguales, resulta un error.

El valor LRC se calcula sumando entre sí los sucesivos bytes del mensaje, descartando cualquier acarreo y luego complementando a dos el valor resultante. Se realiza sobre el contenido del campo de mensaje ASCII excluyendo el carácter ':' de comienzo del mensaje y excluyendo el par CRLF de final de mensaje.

### 2.2.4.6.3. Generación de *CRC*

En modo RTU, los mensajes incluyen un campo de comprobación de error que está basado en un método Comprobación de Redundancia Cíclica (CRC). El campo CRC controla el contenido del mensaje completo. Se aplica con independencia de cualquier método de control de paridad utilizado para los caracteres individuales del mensaje.

El campo CRC es de dos bytes, conteniendo un valor binario de 16 bits. El valor CRC es calculado por el dispositivo emisor, que añade el CRC al mensaje. El dispositivo receptor calcula el CRC durante la recepción del mensaje y compara el valor calculado con el valor recibido en el campo CRC. Si los dos valores no son iguales, resulta un error.

Para calcular el valor CRC Modbus se precarga un registro de 16 bits, con cada uno de los bits puestos en 1. Luego comienza un proceso que toma los sucesivos bytes del mensaje y los opera con el contenido del registro y actualiza éste con el resultado obtenido. Sólo los 8 bits de dato de cada carácter son utilizados para generar el CRC.
Los bits de arranque y paro y el bit de paridad, no se tienen en cuenta para el CRC.

Durante la generación del CRC, se efectúa una operación booleana OR exclusivo (XOR) a cada carácter de 8 bits con el contenido del registro. Entonces al resultado se le aplica un desplazamiento de bit en la dirección de bit menos significativo (LSB), rellenando la posición del bit más significativo (MSB) con un cero. El LSB es extraído y examinado. Si el LSB extraído fuese un 1, se realiza un XOR entre el registro y un valor fijo preestablecido[6]. Si el LSB fuese un 0, no se efectúa un el XOR.

Este proceso es repetido hasta haber cumplido 8 desplazamientos. Después del último desplazamiento (el octavo), el próximo byte es operado XOR con el valor actual del registro y el proceso se repite con ocho desplazamientos más, como se ha descrito más arriba y así con todos los bytes del mensaje. El contenido final del registro, después de que todos los bytes del mensaje han sido procesados, es el valor del CRC.

[6] El valor preestablecido es A001 hex, correspondiente al polinomio generador CRC16 'Inverso', que es el que se aplica al CRC Modbus. Modicon Modbus Reference Guide. PI-MBUS-300 Rev. J. Pag. 112

Cuando el CRC es añadido al mensaje, primero se añade el byte de orden bajo seguido del byte de orden alto. Un procedimiento para generar un CRC es:

- Cargar un registro de 16 bits que denominaremos registro CRC, con FFFF (todos 1).
- XOR del primer byte - 8 bits - del mensaje con el byte de orden bajo del registro CRC de 16 bits, colocando el resultado en el registro CRC.
- Desplazar el registro CRC un bit a la derecha (hacia el LSB "bit menos significativo") rellenando con un cero el MSB (bit más significativo). Extraer y examinar el LSB.
- (Si el LSB era 0): Repetir paso 3 (otro desplazamiento). (Si el LSB era 1): Hacer XOR entre el registro CRC y el valor polinómico A001hex (1010 0000 0000 0001).
- Repetir los pasos 3 y 4 hasta que se hayan efectuado 8 desplazamientos. Una vez hecho esto, se habrá procesado un byte completo – 8 bits.
- Repetir los pasos 2 al 5 para el próximo byte – 8 bits – del mensaje. Continuar haciendo esto hasta que todos los bytes hayan sido procesados.
- El contenido final del registro CRC es el valor CRC.
- Cuando el CRC es situado en el mensaje, sus bytes de orden alto y bajo han de ser permutados.

### 2.2.4.7. Descripción de las funciones

#### *2.2.4.7.1. Función 01 y 02*

Leen el estado de n entradas ó salidas discretas, respectivamente. La petición especifica el bit de inicio y la cantidad de bits a leer. En la respuesta, los datos se empaquetan[7] en bytes, donde el primer bit solicitado ocupa el bit de menos peso del primer byte, continuando con los siguientes, si no se completa un byte, se ponen a cero el resto e bits ver Figura 36 y Figura 37.

[7] Las especificaciones aquí mencionadas fueron tomadas de Modicon Modbus Protocol Referente Guide. PI-MBUS-300 Rev. J. Pag. Desde 24 hasta 27.

| # Esclavo (00-3Fh) | Código de Función (01 o 02) | Dirección del primer bit | | Numero de bits | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| **1 byte** | **1 byte** | **MSB** | **LSB** | **MSB** | **LSB** | **MSB** | **LSB** |

*Figura 36 - Petición Funcion (01 y 02) Maestro.*

| # Esclavo (00-3Fh) | Código de Función | Número de bytes leídos | Primer byte de datos | ..... | Último byte de datos | CRC(16) | |
|---|---|---|---|---|---|---|---|
| **1 byte** | **1 byte** | **1 byte** | **1 byte** | **.....** | **1 byte** | **MSB** | **LSB** |

*Figura 37 - Respuesta Función (01 y 02) Esclavo.*

### *2.2.4.7.2. Función 03 y 04*

Leen el contenido binario de N registros mantenidos ó de entrada. Estos registros guardan los parámetros y variables del controlador. La petición especifica el registro de inicio y la cantidad de registros a leer. En la respuesta, los datos de cada registro son empaquetados[8] con dos bytes ver Figura 38 y Figura 39.

| # Esclavo (00-3Fh) | Código de Función (03 o 04) | Dirección del primer registro | | Numero de registos a leer (máx. 51 ) | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| **1 byte** | **1 byte** | **MSB** | **LSB** | **MSB** | **LSB** | **MSB** | **LSB** |

*Figura 38 - Petición Función (03 y 04) Maestro.*

| # Esclavo (00-3Fh) | Código de Función | Número de bytes leídos | Valor del primer registro | | ... | Valor de último registro | | CRC(16) | |
|---|---|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | MSB | LSB | . | MSB | LSB | MSB | LSB |

*Figura 39 - Respuesta Función (04 y 05) Esclavo.*

### *2.2.4.7.3. Función 05*

Esta función permite forzar[9] los valores lógicos de los bits del dispositivo indicado. Para activar el bit de debe enviar 00h, y se activa mediante 01h ó FFh, escribiéndolo en el byte más significativo. En la petición se especifica la referencia al bit o bobina a modificar ver Figura 40 y Figura 41.

| # Esclavo (00-3Fh) | Código de Función (05) | Dirección del bit | | Valor del bit | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | MSB | LSB | MSB | LSB | MSB | LSB |

*Figura 40 - Petición Función (05) Maestro.*

| # Esclavo (00-3Fh) | Código de Función (05) | Dirección del bit | | Valor del bit | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | MSB | LSB | MSB | LSB | MSB | LSB |

*Figura 41 - Petición Función (05) Esclavo.*

[8] Las especificaciones aquí mencionadas fueron tomadas de Modicon Modbus Protocol Referente Guide. PI-MBUS-300 Rev. J. Pag. Desde 28 hasta 31.

[9] Las especificaciones aquí mencionadas fueron tomadas de Modicon Modbus Protocol Referente Guide. PI-MBUS-300 Rev. J. Pag. Desde 32 hasta 33.

### *2.2.4.7.4. Función 06*

Permite modificar el contenido[10] de un solo parámetro o registró en el dispositivo indicado. La petición especifica la referencia del registro a alterar, la respuesta del dispositivo es una réplica de la petición realizada ver Figura 42 y Figura 43.

| # Esclavo (00-3Fh) | Código de Función (06) | Dirección del registro | | Valor del registro | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | MSB | LSB | MSB | LSB | MSB | LSB |

*Figura 42 - Petición Función (06) Maestro.*

| # Esclavo (00-3Fh) | Código de Función (06) | Dirección del registro | | Valor del registro | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | MSB | LSB | MSB | LSB | MSB | LSB |

*Figura 43 - Respuesta Función (06) Esclavo.*

### *2.2.4.7.5. Función 15*

Fuerza el estado de una o más salidas, relacionadas como una secuencia de bits, a un estado lógico. La petición especifica[11] la representación de los bits y su secuencia en binario, la respuesta será la dirección del esclavo, el código de función, la dirección de inicio y numero de bits forzados ver Figura 44 y Figura 45.

| # Esclavo (00-3Fh) | Código de Función (15) | Dirección del primer byte | | Numero de bytes a leer | | CRC(16) | |
|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | MSB | LSB | MSB | LSB | MSB | LSB |

*Figura 44 - Petición Función (15) Maestro.*

[10] Las especificaciones aquí mencionadas fueron tomadas de Modicon Modbus Protocol Referente Guide. PI-MBUS-300 Rev. J. Pag. Desde 34.

[11] Las especificaciones aquí mencionadas fueron tomadas de Modicon Modbus Protocol Referente Guide. PI-MBUS-300 Rev. J. Pag. Desde 45.

| # Esclavo (00-3Fh) | Código de Función (15) | Número de bytes leídos | Valor del primer byte | | ... | Valor de último byte | | CRC(16) | |
|---|---|---|---|---|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | MSB | LSB | . | MSB | LSB | MSB | LSB |

*Figura 45 - Respuesta Función (15) Esclavo.*

### 2.2.5. Sistema de Adquisición, Control y Supervisión de Datos

SCADA proviene de las siglas de **S**upervisory **C**ontrol **A**nd **D**ata **A**dquisition (Adquicion de datos y supervisión de control). Es una aplicación software de control de producción, que se comunica con los dispositivos de campo y controla el proceso de gorma automática desde la pantalla del ordenadores, supervisores de control de calidad, supervisión, mantenimiento, etc.

Actualmente los sistemas de interfaz entre usuario y planta basados en paneles de control repletos de indicadores luminosos, instrumentos de medida y pulsadores, están siendo sustituidos por sistemas digitales que implementan el panel sobre la pantalla de un ordenador.

El control directo lo realizan los controladores autónomos digitales y/o autómatas programables y están conectados a un ordenador que realiza las funciones de dialogo con el operador, tratamiento de la información y control de la producción, utilizando SCADA.

#### 2.2.5.1 Funciones y prestaciones principales del SCADA

- Adquisición de datos para recoger, procesar y almacenar la información recibida.
- Supervisión para observar desde un monitor la evolución de las variables.
- Control para modificar la evolución del proceso, actuando bien sobre los reguladores autónomos básicos (consignas, alarmas, menús, etc) bien directamente sobre el proceso mediante las salidas conectadas.

### 2.2.6. Interfaz OPC

OPC es una norma de intercambio de datos para el nivel de planta basada en la tecnología OLE (Object Linking and Embedding) denominada OPC (OLE for Process Control), el cual es un método para el flujo transparente de datos entre aplicaciones corriendo bajo sistemas operativos basados en Microsoft Windows, proporcionando un acceso simple a los datos. La fundación OPC está formada por: Siemens, Fisher, Intuitivo, OPTO 22, Intellution, Rockwell, etc.

El OPC es un primer paso concreto que permite a una red compartir los datos de los dispositivos a nivel de proceso. La primera versión del OPC salió en agosto de 1996. Este define un estándar de intercambio de información y las reglas de negociación entre dispositivos de diferentes tipos. Así cualquier dispositivo que posea un software de control de tipo OPC podrá conectarse con cualquier software cliente OPC consiguiendo de esta manera una gran flexibilidad y conectividad, y la capacidad de añadir diferentes dispositivos a un software de control y adquisición de datos sin tiene que modificar el mismo las ventajas que obres esta tecnología.

El objetivo de OPC e que en la industria se puedan utilizar herramientas estándar (paquetes SCADA, bases de datos, hojas de cálculo) para construir un sistema que responda a sus necesidades de mejora de la productividad.

Para ello es necesario desarrollar una arquitectura de comunicación abierta y efectiva que se centra en el acceso a datos, no en los tipos de datos, es decir, que sea independiente del tipo de bus de campo o de datos empleando en cada una de las parte del proceso productivo y empresarial.

Hay muchas aplicaciones cliente que requieren datos de dispositivos y acceden a ellos desarrollando controladores o drivers de forma independiente ver Figura 46.

- Duplicación de esfuerzos todos los programas necesitan un driver para un determinado hardware (p.e. un bus de campo).

- Falta de consistencia entre drivers hay características del hardware no soportadas por todos los drivers.
- Cambios en el hardware hacen que los drivers queden obsoletos.
- Conflictos de acceso generalmente dos programas no pueden acceder simultáneamente al mismo dispositivo puesto que poseen drivers independientes


*Figura 46 - Diferencias entre la arquitectura convencional y OPC.*

OPC proporciona un mecanismo para extraer datos de una fuente y comunicarlos a cualquier aplicación cliente de manera estándar. Los fabricantes de hardware pueden desarrollar servidores optimizados para recoger datos de sus dispositivos. Dando al servidor una interface OPC permite a cualquier cliente acceder a dichos dispositivos.

- Permite crear objetos, que son piezas de código reutilizables para facilitar la implantación y mantenimiento de las aplicaciones.
- Permite crear objetos entre diferentes aplicaciones de modo que puedan interoperar y comunicarse atraves de una red.

Estos permiten que múltiples dispositivos que se comunican por distintos protocolos puedan compartir el mismo puerto de comunicación con el sistema de supervisión ver Figura 47.

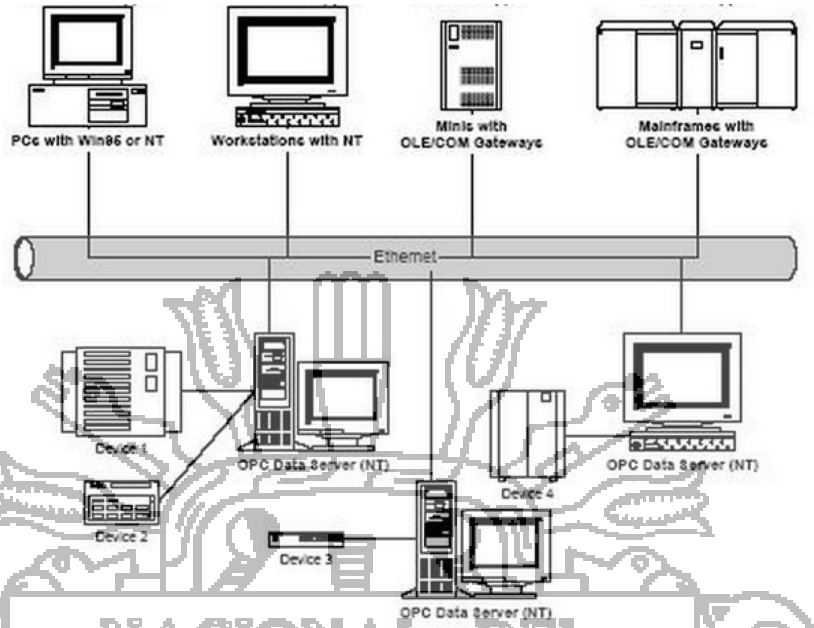



*Figura 47 - Esquema Genérico de conexionado OPC.*

Dentro de la tecnología OPC existen varias subtecnologías dependiendo del tipo y la forma de intercambiar información con los servidores OPC.

- OPC-DA (Data Access): Sirve para el intercambio de datos a tiempo real entre servidores y clientes.
- OPC-AE (Alrms y Events): Proporciona alarmas y notificaciones de eventos.
- OPC-B(Batch): Util en proceso discontinuo.
- OPC-DX (Data Exchange): Proporciona interoperabilidad entre varios servidores.
- OPC- HDA (Historical Data Access): Acceso histórico a datos OPC.

El trabajo del estándar OPC actualmente está acreditada por más de las 200 compañías y asociaciones más importantes del sector como por ejemplo Microsoft, CERN, Compap o National Instruments, lo que garantiza un soporte constante y continuas revisiones del estándar.

### 2.2.7. Estación Meteorológica

Una estación meteorológica es un lugar escogido adecuadamente para colocar los diferentes instrumentos que permiten medir las distintas variables que afectan al estado de

la atmósfera. Es decir, es un lugar que nos permite la observación de los fenómenos atmosféricos que miden las variables atmosféricas. Muchos de estos han de estar al aire libre, Estos datos se utilizan tanto para la elaboración de predicciones meteorológicas a partir de modelos numéricos como para estudios climáticos. Una estación meteorológica es una instalación destinada a medir y registrar regularmente diversas variables meteorológicas. Estos datos se utilizan tanto para la elaboración de predicciones meteorológicas a partir de modelos numéricos como para estudios climáticos instrumentos de Medición.

**2.2.7.1. Variables climáticas**

Para realizar estudios meteorológicos se debe tomar mediciones de las diferentes magnitudes físicas presentes en la naturaleza, con la finalidad de determinar el comportamiento del clima en una determinada zona, se describe los diferentes tipos de variables climáticas existentes dentro del estudio del clima.

- Temperatura.
- Humedad.
- Presión Atmosférica.
- Velocidad del viento.

**2.2.7.1.1. Temperatura**

La temperatura atmosférica es el indicador de la cantidad de energía calorífica acumulada en el aire. Aunque existen ciertas escalas, la temperatura del aire se suele medir en grados centígrados (ºC).

La temperatura depende de diversos factores como la inclinación de los rayos solares. También depende del tipo de sustratos (la roca absorbe energía, el hielo la refleja), la dirección y fuerza del viento, la latitud, la altura sobre el nivel del mar, la proximidad de masas de agua. Sin embargo, hay que distinguir entre temperatura y sensación térmica. Aunque el termómetro marque la misma temperatura, la sensación que se percibe depende de factores como la humedad del aire y la fuerza del viento.

**2.2.7.1.2. Humedad**

La humedad indica la cantidad de vapor de agua presente en el aire. Depende en parte de la temperatura, ya que el aire caliente contiene más humedad que el aire frío.

La humedad relativa se expresa en forma de tanto por ciento (%) de agua en el aire. La humedad absoluta se refiere a la cantidad de vapor de agua presente en una unidad de volumen de aire y se expresa en gramos por centímetro cúbico (g/cm3). La saturación es el punto a partir del cual una cantidad de vapor de agua no puede seguir creciendo y mantenerse en estado gaseoso, sino que se convierte en líquido y se precipita.

- Humedad absoluta: Es la cantidad de vapor de agua presente en el aire, se expresa en gramos de agua por kilogramos de aire seco (g/kg), gramos de agua por unidad de volumen (g/m³) o como presión de vapor (Pa o KPa o mmHg). A mayor temperatura, mayor cantidad de vapor de agua permite acumular el aire.

La humedad relativa: Es la humedad que contiene una masa de aire, en relación con la máxima humedad absoluta que podría admitir sin producirse condensación, conservando las mismas condiciones de temperatura y presión atmosférica. Esta es la forma más habitual de expresar la humedad ambiental. Se expresa en tanto por ciento %.

$$RH = \frac{p_{(H_2O)}}{p^*_{(H_2O)}} \times 100\%$$

- P (H2O) Es la presión parcial de agua en la mezcla de aire.
- P*(H2O) Es la presión se saturación de vapor de agua a la temperatura en la mezcla de aire.
- RH Es la humedad relativa de la mezcla de aire que se está considerando.

**2.2.7.1.3. Presión Atmosférica**

La presión atmosférica es el peso de la masa de aire por cada unidad de superficie. La presión suele ser mayor a nivel del mar que en las cumbres de las montañas, aunque no depende únicamente de la altitud.

Las grandes diferencias de presión se pueden percibir con cierta facilidad. Una presión alta produce sensación de cansancio. La presión atmosférica decrece al aumentar la altitud debido a que la cantidad de aire que se tiene por encima también disminuye. La presión atmosférica también varía con la temperatura. Así, si una zona de la atmósfera es calentada fuertemente, aumenta su temperatura y la densidad del aire disminuye, por lo tanto, también la presión atmosférica. De la misma manera, si en un lugar concreto el aire se enfría, la presión atmosférica de ese lugar aumentará.

**2.2.7.1.4. Velocidad del viento**

El viento es el movimiento natural del aire atmosférico. En meteorología, esta palabra se refiere, en general, a un movimiento del conjunto del aire cerca de la superficie terrestre o en altitud.

El movimiento del aire raramente es regular. Generalmente es turbulento con torbellinos de forma y dimensiones variadas, que se desarrollan en el aire y perturban su flujo.

**2.2.7.2. Sensores climáticos**

La normalización del instrumental es una cuestión muy importante dado que para la medida de una misma variable meteorológica pueden encontrarse en el mercado diferentes tipos de instrumentos que pueden diferenciarse en sus constantes de tiempo y en su precisión.

El instrumental de las estaciones debe estar aprobado y normalizado por los servicios Meteorológicos y, en general, son estas entidades las que suministran los instrumentos de observación, con lo que se garantiza la uniformidad

**2.2.7.2.1. Sensores de temperatura**

Probablemente sea la temperatura el parámetro físico más común que se mide en una aplicación electrónica; incluso en muchos casos el parámetro de interés no es la temperatura, pero ésta se ha de medir para incluir indirectamente su efecto en la medida deseada. Lo que permite realizar una compensación en el sistema de acuerdo a la variación de esta magnitud.

La diversidad de sus aplicaciones ha condicionado igualmente una gran proliferación de dispositivos sensores y transductores, desde la sencilla unión bimetálica de los termostatos, hasta los dispositivos semiconductores más complejos.

Existen infinidad de procesos industriales en los que resulta imprescindible monitorear y controlar la temperatura, e incluso en el ámbito doméstico es una de las magnitudes de mayor interés.

En general la mayor dificultad consiste en seleccionar que sensor se va a utilizar para determinada aplicación, por lo cual a continuación se describe los tipos de sensores de temperatura existentes y su principio de funcionamiento.

- **Termistores:** Un termistor es un resistor cuyo valor varía en función de la temperatura. Existen dos clases de termistores: NTC (Negative Temperature Coefficient, Coeficiente de Temperatura Negativo), que es una resistencia variable cuyo valor se decrementa a medida que aumenta la temperatura; y PTC (Positive Temperature Coefficient, Coeficiente de Temperatura Positivo), resistencia eléctrica aumenta cuando se incrementa la temperatura ver Figura 48.

*Figura 48 - Termistores.*

- **RTD (Termorresistencias).** Los sensores RTD (Resistance Temperature Detector), basados en un conductor de platino y otros metales, se utilizan para medir temperaturas por contacto o inmersión, y en especial para un rango de temperaturas elevadas, donde no se pueden utilizar semiconductores u otros materiales sensibles. Su funcionamiento está basado en el hecho que en un metal, cuando sube la temperatura, aumenta la resistencia eléctrica ver Figura 49.

*Figura 49: Termorresistencias.*

- **Termocuplas.** El sensor de una termocupla está formado por la unión de dos piezas de metales diferentes. La unión de los metales genera un voltaje muy pequeño, que varía con la temperatura. Su valor está en el orden de los milivoltios, y aumenta en proporción con la temperatura. Este tipo de sensores cubre un amplio rango de temperaturas: -180 a 1370 °C ver Figura 50.

*Figura 50 - Termocuplas.*

**2.2.7.2.2. Sensores de presión**

Los sensores de presión se encargan de medir la fuerza aplicada sobre una superficie cualquiera. Se aplica la ecuación siguiente una Figura 51.

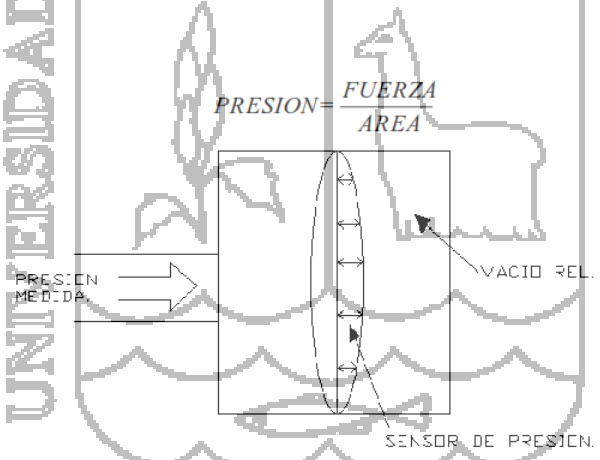



*Figura 51 - Sensor mide la presión de un fluido.*

- **Sensores de presión resistivos:** La presión ejercida sobre una membrana, hace variar el valor de las resistencias montadas en un puente de Wheatstone. Las Células de Carga y las Galgas Extensiométricas son elementos metálicos que cuando se someten a un esfuerzo sufren una deformación del material, y por lo tanto una variación de su resistencia interna ver Figura 52.

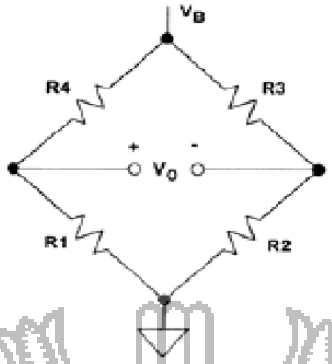



*Figura 52 - Sensor de presión resistivo.*

- **Sensores de presión piezo-cerámicos/multicapa:** La combinación de la tecnología piezo-cerámica y multicapa se utiliza para producir una señal eléctrica, cuando se aplica una fuerza mecánica en el sensor se esquematiza el funcionamiento de un sensor de presión Piezo-cerámico ver Figura 53.

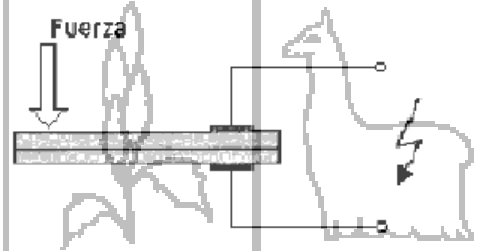



*Figura 53 - Sensor de presión piezo-cerámico.*

**2.2.7.2.3. Sensores de humedad relativa**

La detección de la humedad relativa es muy importante en una estación automática, ya que a partir de ésta se puede determinar el punto de rocío, que junto con la temperatura permiten determinar si pueden o no presentarse lluvias.

Por esta razón se debe tener en cuenta una variedad de sensores de humedad disponibles, entre ellos los capacitivos y resistivos, y algunos integrados con diferentes niveles de complejidad y prestaciones.

- **Sensores de Humedad Capacitivos.** El sensor lo forma un condensador de dos láminas de oro como placas y como dieléctrico una lámina no conductora que varía su constante dieléctrica, en función de la humedad relativa de la atmósfera ambiente. El valor de la capacidad se mide como humedad relativa. *Honeywell* fabrica este tipo de sensores.
- **Sensores de Humedad Resistivos.** Un electrodo polímero montado en tandem, censa la humedad en el material. Además un circuito acondicionador y linealizador dan una salida estándar.

### 2.2.7.2.4. Sensor de velocidad angular

El sensor construido está compuesto de un disco ranurado que gira solidariamente con respecto a un eje y cuya fricción es mínima de tal forma que es muy sensible al viento. El eje cuya posición se quiere medir va acoplado al disco, a medida que el eje gira se van generando pulsos en el receptor cada vez que la luz atraviese las marcas, llevando una cuenta de estos pulsos es posible conocer la posición del eje y se puede calcular la velocidad: el número de vueltas o pulsos por unidad de tiempo.

La resolución depende del número de marcas que se pueden poner físicamente en el disco ver Figura 54.

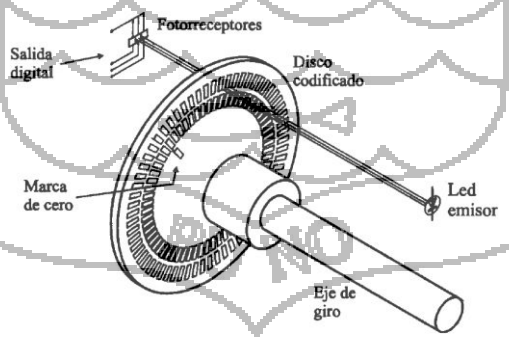



*Figura 54 - Codificador óptico formado por: disco ranurado, led y fotodetector.*

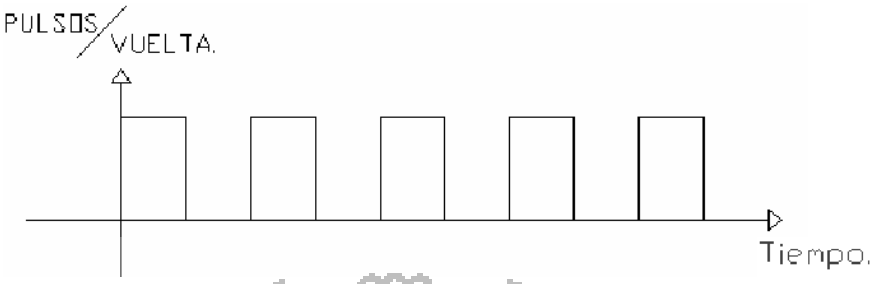



*Figura 55 - Señal de salida del sensor de velocidad*

## 2.3. Hipótesis de la Investigación

### 2.3.1. Hipótesis General

- Diseño e Implementación de una Interfaz Utilizando el Protocolo Modbus Para la Comunicación en Redes Industriales facilita la supervisión y monitoreo de datos.

### 2.3.2. Hipótesis Específicas

- Diseño y estudio de la información correspondiente a redes industriales y el protocolo de comunicación Modbus.
- Implementar la Comunicación, capaz de armar, procesar y destramar datos basándose en las especificaciones del protocolo Modbus.

# CAPÍTULO III

## MÉTODO DE INVESTIGACIÓN

### 3.1. Procedimientos de Diseño del Interfaz RS-232 A RS-485

El circuito implementado permite la comunicación serie entre 2 nodos (como máximo 32) a la distancia de hasta 1200 metros de longitud.

Es una etapa indispensable de la comunicación RS-485 considerando que la PC se comunica por RS-232. Inicialmente, modo de trabaja de comunicación RS-485 ver Figura 56:


*Figura 56 - operación de comunicación RS-485.*

- Tensión de señal <6V
- Estado lógico 1 = diferencia de potencial <200mV
- Estado lógico 0 = diferencia de potencial >200mV

Es necesario utilizar un driver que realice el trabajo de reconocer la señal diferencial y convertirla en señal de 0 a 5V. (MAX 485), para después utilizar otro driver que convierta la señal TTL a niveles RS-232 (MAX 232) para obtener, finalmente, la comunicación con el puerto serie de la PC.

Así como también, un circuito Monoestable para conmutar la señal de habilitación para transmisión o recepción de en baudios correspondientes ver Figura 57.

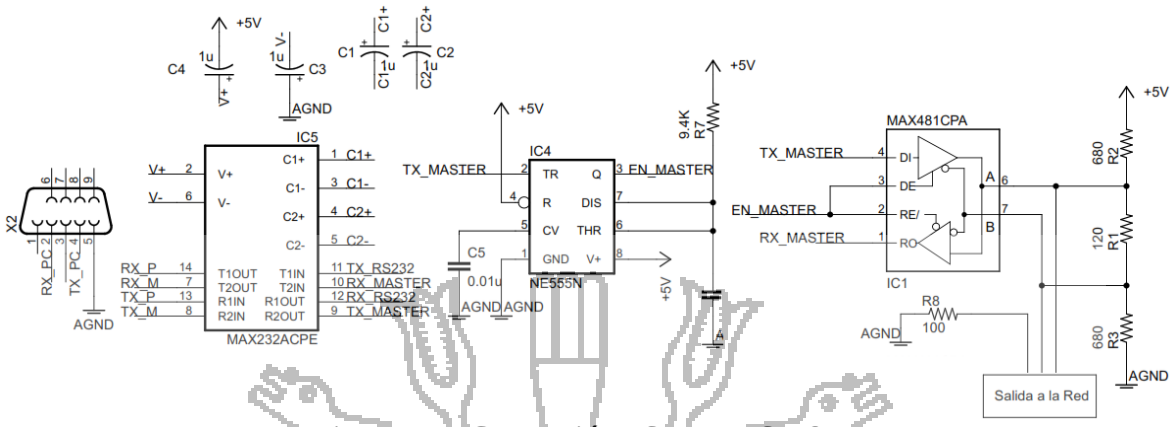
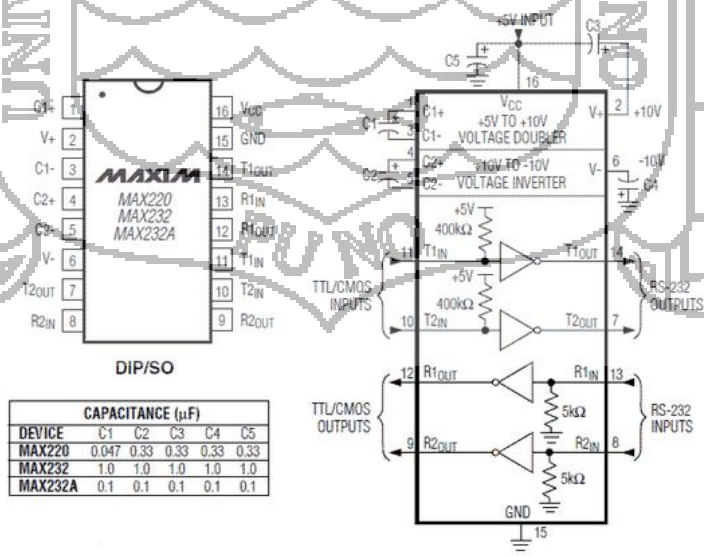



*Figura 57 - Conversión RS232 a RS458.*

### 3.1.1. MAX 232

Inicialmente se debe convertir las señales RS232 que llegan del puerto serie del PC a selñales con niveles de tensioon TTL/CMOS. El estandar RS232 define unos niveles de tension no equilibrados, tomando la señal tierra como nivel de referencia. El nivel logico "1" es transmitido con una tension negativa entre -3 y -25 voltios y nivel logico "0" con una tension positiva entre 3 y 25 voltios.

Por esta razon se ha empleado el ciruito integrado MAX232 para convertir estos niveles de tension en señales deigitales TTL/CMOS. Ahora los margenes de tension varian entre 0 y 0.4 voltios para el valor logico "0" y entre 3.5 y 5 voltios para el valor logico "1".




| CAPACITANCE (µF) | | | | | |
|---|---|---|---|---|---|
| DEVICE | C1 | C2 | C3 | C4 | C5 |
| MAX220 | 0.047 | 0.33 | 0.33 | 0.33 | 0.33 |
| MAX232 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| MAX232A | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

*Figura 58 - MAX232.*

En la ver Figura 58, debemos señalar que además de la conversión de niveles eléctricos se produce una inversión de polaridad. Ya que este componente se alimenta de una única tensión de 5 voltios. Gracias a un doblador de tensión que contiene internamente este integrado se obtiene una alimentación de +10 en el pin 2 y por medio de un inversor de tensión se logra en el pin 6 una alimentación de -10 voltios. Estas tensiones son necesarias para generar los niveles de salida del RS232. Los condensadores C1 y C2 son empleados por el integrado como conversores de tensión mientras que los condensadores C3 y C4 desacoplan y filtran las alimentaciones generadas como se muestra en la ver Figura 59.

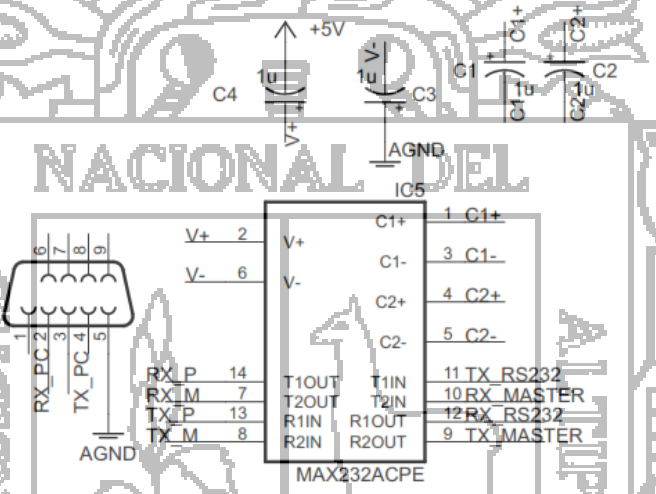



*Figura 59 - Niveles de tension en señales digitales TTL/CMOS.*

### 3.1.2. MAX 485

Este circuito integrado se encarga de la conversión de las señales digitales con niveles TTL a las señales diferenciales descritas en la norma RS-485. Este componente integra un receptor y un trasmisor diferencial y ha sido diseñado para conectarse a líneas de transmisión balanceada, con objeto de mejorar las características eléctricas de sistemas de comunicación Half-Duplex que emplean cables o líneas largas, o que están situados en ambientes ruidosos y entornos industriales ver Figura 60.

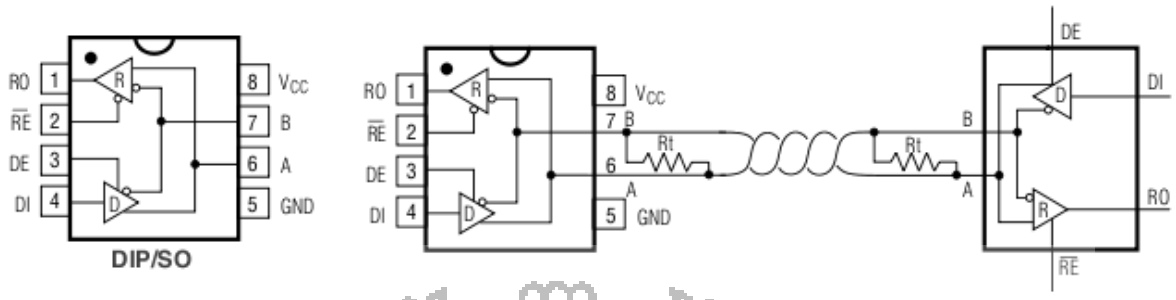



*Figura 60 - Conversión señales TTL a señales diferenciales.*

Para habilitar el transmisor y receptor diferencial se utilizan dos pines del MAX 485, el pin2 ( $\overline{RE}$) es el encargado de habilitar la recepción de datos y el Pin3 (DE) es el encargado de habilitar la transmisión de datos.

Como se observa en laos pines 2 y 3 estan unidos lo que significa que se puede transmitir o recibir datos en el modo de comunicación Half Duplex ver Figura 61.

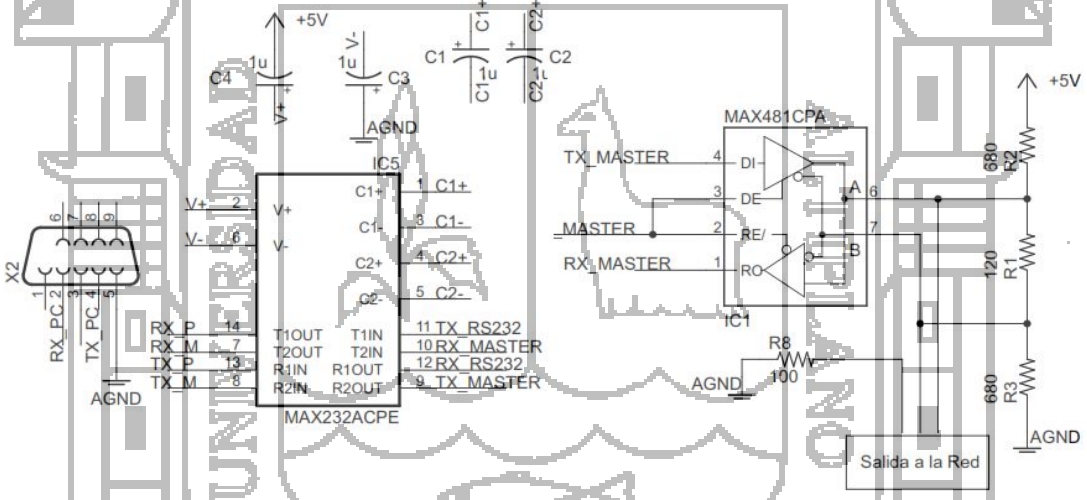



*Figura 61 - Conmutación de Tx – Rx para la comunicación RS-485.*

Se tiene una red de 3 nodos con dos terminadores de 120Ω, cada nodo de la red tiene una impedancia de entrada de 12K Ω, los tres nodos en paralelo tendrán un total de 4KΩ,

$$Req. = \frac{12000 * 12000}{24000} = 6000\Omega$$

$$Rtot. = \frac{6000 * 12000}{18000} = 4000\Omega$$

Las dos resistencias de terminación en paralelo añaden una carga de otros 60Ω, lo que resulta en una carga total (de todas en paralelo) de 59.11Ω, con o que se ve calramene que los terminadores son los responsables de la mayor parte de la carga.

$$Rtot. = \frac{4000 * 60}{4060} = 59.11\Omega$$

Para mantener una tensión minima de 200mV entre B y A, necesitamos una corriente de bias de 3.38mA a través de la carga.

$$Ibias = \frac{200mV}{29.11\Omega} = 3.38mA$$

Para crearla a partir de los 5V de alimentación, necesitamos una resistencia total de 1477.75Ω o menos.

$$R = \frac{5V}{3.38} = 1477.75\Omega$$

Si le restamos los 59.11Ω que ya están presentes en el circuito como parte de la carga, nos quedan 1418.64Ω.

$$R = 1477.75 - 59.11 = 1418.64\Omega$$

Poniendo la mitad de este valor como resistencia de pull – up, y la otra mitad como resistencia de pull – down, y la otra mitad como resistencia pull – down, nos quedan una resistencias de bias con un valor máximo de 709.32Ω cada una.

$$R = \frac{1418.64}{2} = 709.32\Omega$$

Por lo tanto, eligió la resistencia comercial de 680Ω ver Figura 62.

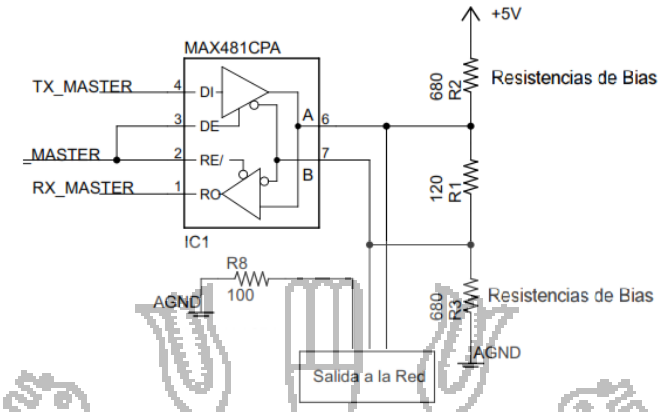



*Figura 62 - Calculo de las Resistencia de bias.*

### 3.1.3. LM555

La función de este circuito integrado es de generar una señal equivalente a la señal RTS entrega por el puerto serie de la PC, la velocidad de transmisión empleada en la red de comunicación 9600bps, por lo cual este circuito genera una ventana de tiempo cada vez que se trasmite un dato.

Este circuito es controlado por el controlador de la transmisión está habilitado durante toda la trasmisión, entonces se desactiva tan pronto como sea posible después de bit de parada final, el cual le envía 1 lógico cuando transmite información y a su vez el LM555 habilita los pines 2 o 3 del MAX485 a una velocidad de 9600bps.

El modo de operación en que trabaja es Monoestable ver Figura 63.

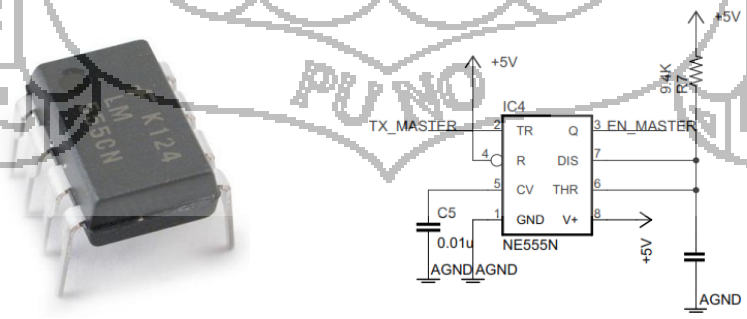



*Figura 63 - LM555.*

Calculo de Resistencia y condensador para generar la ventana de tiempo de 9600bps. Según análisis de diseño se escogió el siguiente valor para el condensador C=10nF Como la velocidad de transmisión es de 9600bsp.

$$t = \frac{1}{f}$$

Entonces: $t = \frac{1}{9600bps} = 104.1667us$

Remplazando valores: $t = 1.1 * Rx * c$

Despejando Rx: $Rx = \frac{104.1667us}{1.1*10nF} = 9469.69 = 9.4K$

### 3.2. Diseño e Implementación del Controlador Esclavos y Acondicionamiento de Señal

Las tarjetas esclavas Modbus, sigue las especificaciones creadas por Modicon para la construcción de un tarjetas Esclavas Modbus típico, en cuanto a la parte que al Protocolo se refiere.

Las tarjetas esclavos son las encargadas de la adquisición de señales de tipos analógicos del módulo en su totalidad. Este consta de comunicación, modbus sobre RS-232, RS-485, a continuación se detalla el diseño de las tarjetas esclavas. Estas tarjetas están basadas en un micro controlador PIC16F877A de la familia Microchip, el cual actúan como unos elementos principales de las tarjetas esclavas ya que son las encargadas de interactuar con todos los elementos existentes ver Figura 64.

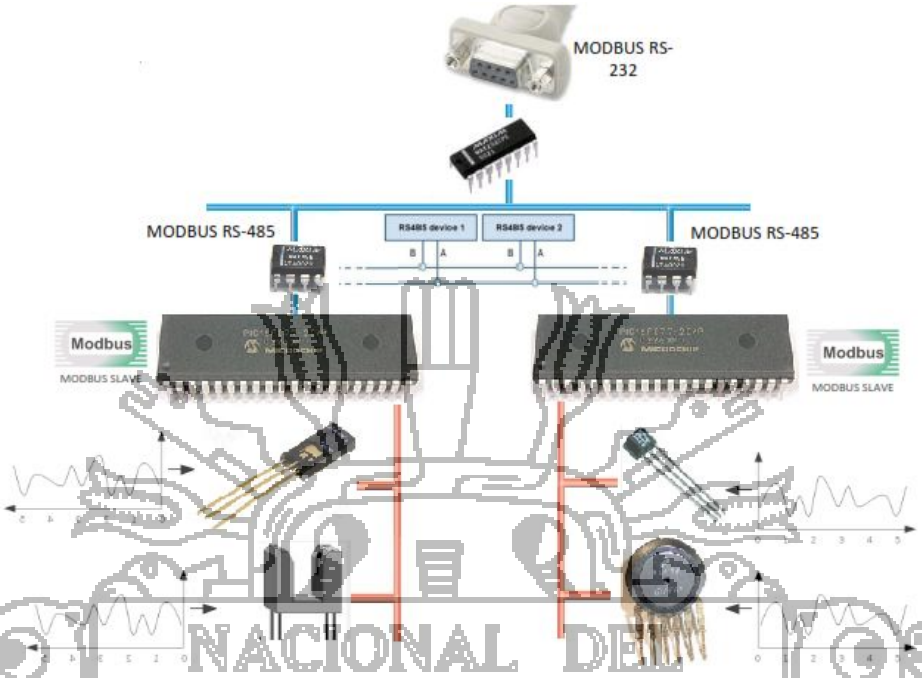



*Figura 64 - Esquema de los esclavos modbus.*

### 3.2.1. Comunicación RS-485.

Este circuito integrado se utiliza para poder cambiar las señales provenientes de la USART del microcontrolador a señales compatibles con la norma RS485 y de esta manera se pueda establecer una red industrial de hasta 32 nodos. El microcontrolador envía o recibe datos por medio de os pines de Tx y Rx (PORTB(0)) y (PORTB(1)) que a su vez van conectados a los pines R0(RX) y DI(TX) del MAX485 que se encargan de enviar o recibir los datos (punto A y B de la red) via RS485. Por otro lado el PORTE(0) se encarga de controlar los habilitadores $\overline{RE}$ (habilitador de Recepción) y $DE$ (Habilitador de Transmisión) del MAX485 ver Figura 65.

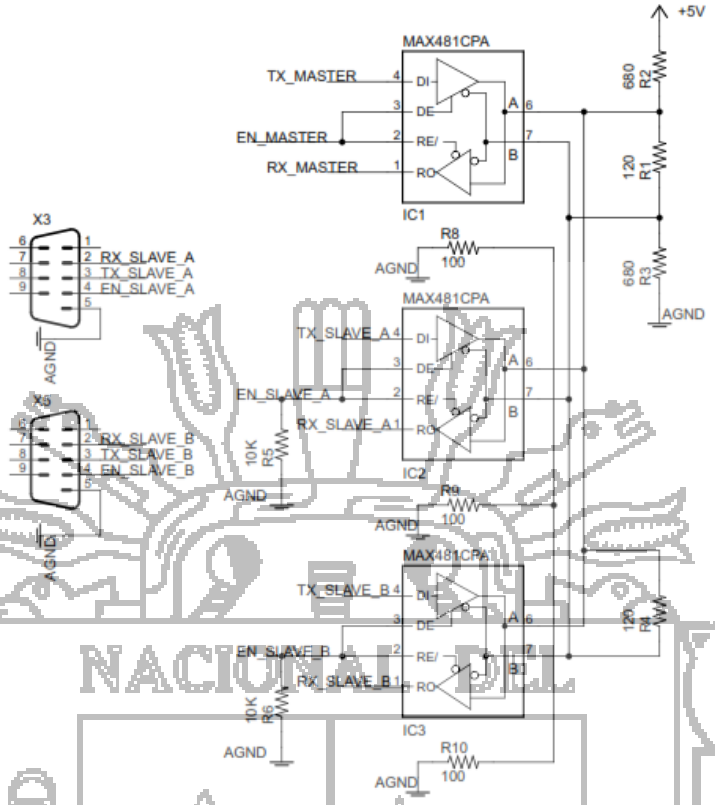



*Figura 65: Comunicación RS-485.*

Se ha usado la siguiente aplicación para la red de comunicación (half duplex) ya explicado en capítulos anteriores ver Figura 66.

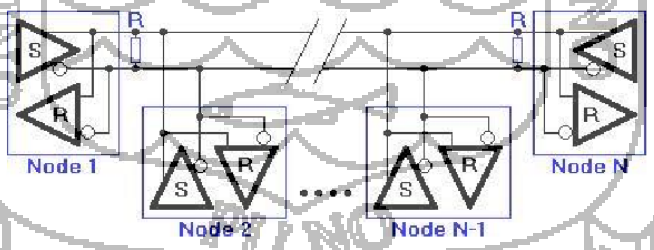



*Figura 66 - Estructura de interconexionado de nodos.*

### 3.2.2. Dispositivos Esclavos Modbus

El microprocesador es el cerebro del dispositivo Esclavos. En él, se encuentran programadas, todas las rutinas necesarias para el funcionamiento del dispositivo para la implementación del dispositivo Esclavo Modbus, se utilizó el microcontrolador PIC16F877

de Microchip, así como los puertos de Entrada/Salida. Estos circuitos tienen asignada las siguientes direcciones de esclavos 10 y 11 en sus memorias en la red industrial ver Figura 67.

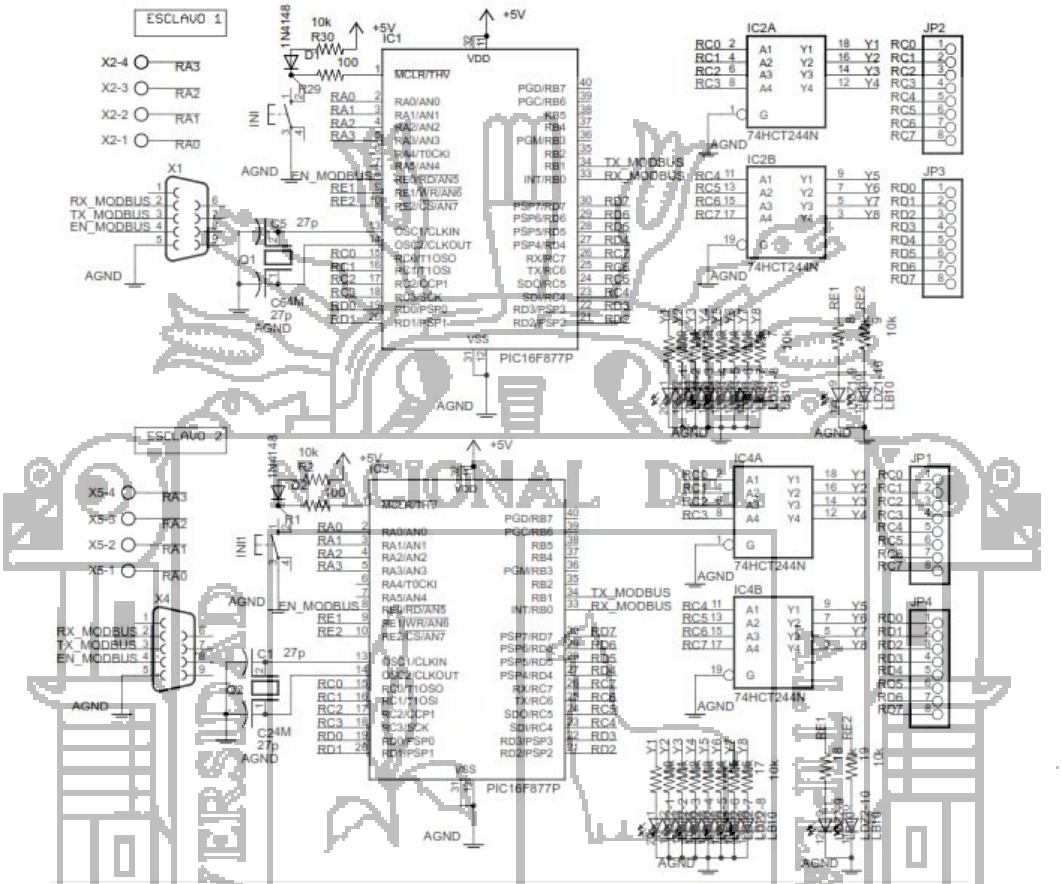



*Figura 67 - Dispositivos esclavos Modbus.*

- Ocho (4) puertos del microcontrolador, configurados como entradas análogas de 0 a 5VDC. El valor obtenido a través de cada uno de estos puertos es almacenado en dos registros de 8 bits de la memoria del microcontrolador, y a este grupo de registros se les denomina Registros de Entrada y su información puede ser obtenida mediante la función Modbus: Leer Registros de Entrada (Función 04).
- De igual manera, se configuraron otros ocho (8) puertos como entradas digitales de 5VDC y se almacenan en un grupo de registros denominados Entradas Digitales. La información contenida en estos registros puede ser leída mediante la función Modbus: Leer Estados de Entradas (Función 02).

- Además, se configuraron ocho (8) puertos como salidas digitales de 5VDC. Los valores correspondientes a los estados de cada uno de estos puertos, se almacenan en un grupo de registros denominados Bobinas, y estos valores pueden ser leídos mediante la Función Modbus: Leer Estados de Bobinas (Función 01) y pueden ser escritos mediante las Funciones Modbus: Forzar Única Bobina (Función 05) que permite activar o desactivar un único puerto y la función Modbus: Forzar Múltiples Bobinas (Función 15) que permite activar o desactivar simultáneamente varios de estos puertos de salida.
- Adicionalmente, se utilizan dos (2) puertos del microcontrolador para la indicación del estado del dispositivo.

**3.2.2.1. Elección del Microcontrolador**

Las principales Características del Microcontrolador, son las siguientes ver Figura 68:

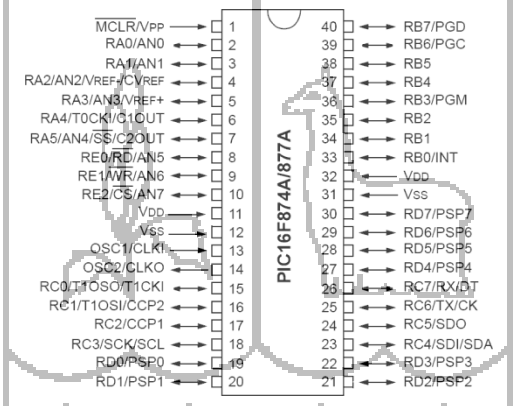



*Figura 68 - Patillaje del PIC16F877A.*

- Juego reducido de instrucciones (RISC/35 instrucciones).
- Velocidad máxima de operación 20Mhz (200ns ciclo de instruccion).
- Tamaño de Memoria Flash (8Kx14 palabras).
- Memoria Ran de 368x8 bytes.
- Maxima cantidad de interrupciones (14).
- Watchdog Timer (WDT) o perro guardián.
- Código de protección programable.
- SLEEP mode.

- Programación vía Puerto Serie y usb.
- Voltaje de operación: 0.20V a 5.5V
- Corriente de Sink/Sourse: 25mA
- 3 timers.
- Modulacion por ancho de pulso (PWM).
- Conversor análogo digital (ADC) de 10bits, 8 canales.
- Comunicación I2c.
- USART/SCI (Universal Syncheronus Asynchronous Receiver Transmitter)

En el mercado, se encontramos muchas alternativas para el Microcontrolador, de las cuales, las de mayores prestaciones tenemos el PIC16f628, PIC16f877A y el PIC 16f870. Para cumplir con lo requerido en este proyecto, la elección del Microcontrolador lo hacemos con ayuda de la siguiente Tabla 8:

| Modelo | Memoria de Programa (Word) | Memoria de Datos | | I/O | Canales de conversión A/D a 10 bits | CCP (PWM) | MSSP | USART | Timers 8/16 bits | Comparadores |
|---|---|---|---|---|---|---|---|---|---|---|
| | | SRAM (Byte) | EEPROM (Byte) | | | | | | | |
| PIC16F870 | 2048 | 128 | 64 | 22 | 5 | 1 | no | si | 2/1 | 0 |
| PIC16F877A | 8192 | 368 | 256 | 33 | 8 | 2 | si | si | 2/1 | 2 |
| PIC16F628A | 2048 | 224 | 128 | 16 | 0 | 1 | no | si | 2/1 | 2 |

*Tabla 8 -: Comparación de las principales características de los PICs.*

### 3.2.2.2. Microcontrolador PIC16F877A

El microcontrolador se encarga de recibir las señales análogas (empleando su ADC interno) provenientes de los sensores (PORTA); luego para comunicarse dentro de la red modbus por medio de la USART y por último se encarga de controlar los habilitadores de transmisión y recepción del MAX485.

### 3.2.2.3. Entradas análogas

Para la implementación de las entradas análogas, se utilizaron los puertos RA1-RA4 del microcontrolador, para un total de 4 entradas análogas, El rango de lectura de las entradas análogas va de 0 a 5V con una resolución de 8 bits (lo que permite en teoría leer variaciones de 20 mV en la entrada). Los datos leídos en cada entrada análoga se almacenan en dos registros sucesivos de 8 bits en la memoria RAM del microcontrolador (desde la dirección 51H hasta la 60H), estos registros se denominan: Registros de Entrada. De esta manera se asegura, que los datos de las entradas análogas estén disponibles en todo momento y puedan ser accedidos utilizando funciones propias del Protocolo de Comunicación Modbus, mientras el dispositivo esclavo esté en normal funcionamiento ver Figura 69.

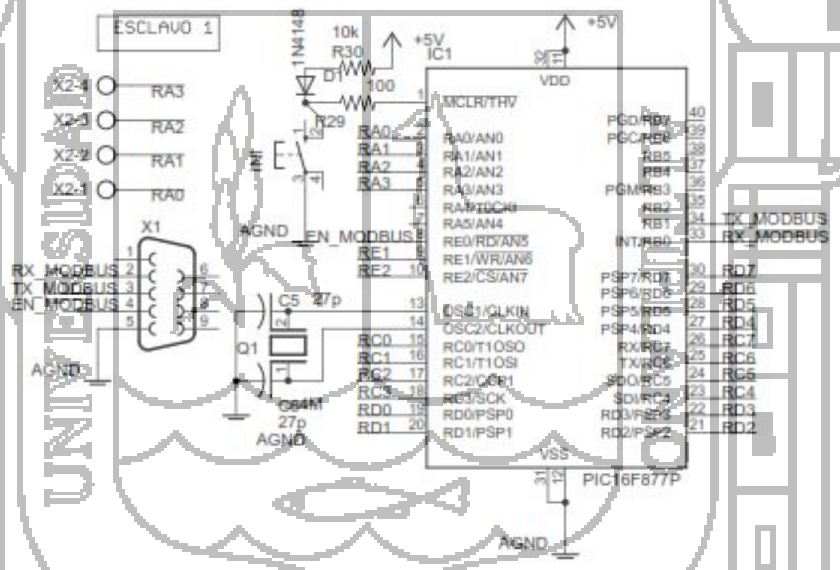



*Figura 69 - Esquema de conexión de las Entradas Análogas.*

### 3.2.2.4. Entradas Digitales

Las 8 entradas digitales disponibles en el dispositivo esclavo Modbus, se implementaron utilizando los puertos RD0-RD7 del microcontrolador. A estas entradas se pueden conectar sensores de dos estados activado y desactivado (encendido – apagado,

abierto – cerrado), manejando un voltaje de 0Vpara el estado desactivado y 5V para el estado activado ver Figura 70.

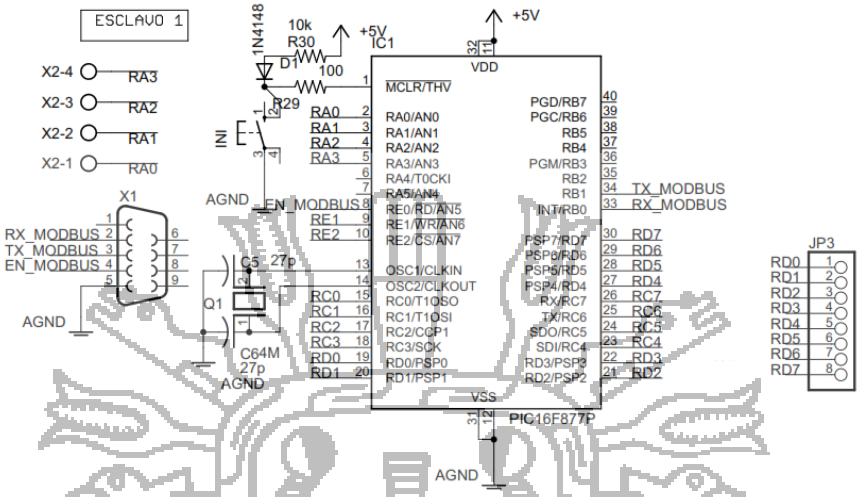



*Figura 70 - Esquema de conexión de las Entradas Digitales.*

### 3.2.2.5. Salidas Digitales

Las salidas digitales fueron implementadas usando los puertos RC0-RC7 del microcontrolador, para un total de 8 salidas digitales. Estas salidas proporcionan normalmente 0V para el estado apagado (OFF) y 5V para el estado Encendido (ON), con una corriente máxima por puerto de 25 mA.

Estas salidas pueden ser utilizadas para activar actuadores a través de etapas de potencia con entrada 0 a 5V DC (consumo de activación menor a 25mA) y salida de acuerdo al tipo de instrumento a manipular ver Figura 71.

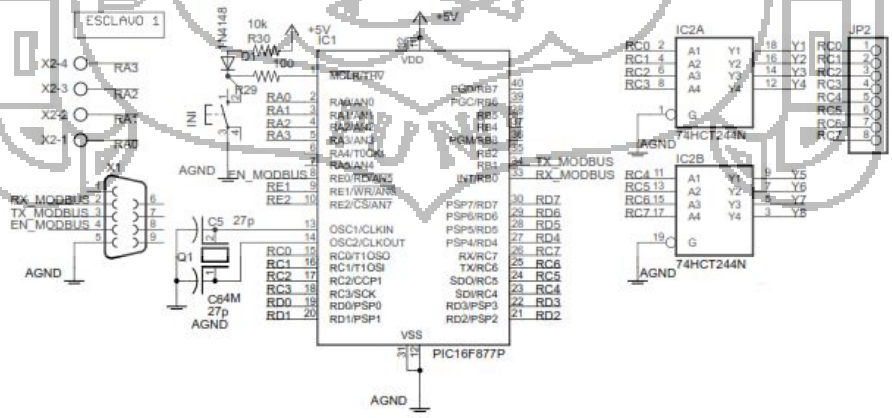



*Figura 71 - Esquema de conexión de las Salidas Digitales.*

### 3.2.2.6. Sistema de Visualización de Estados

Sistema de visualización que muestra el estado actual del mismo. Para crear este sistema se utilizaron los puertos RE1 – RE2, configurados como salidas. A cada uno de estos puertos está conectado un LED ver Figura 72.

Los dos estados indicados por estos LEDs, son:

- Dispositivo Ocupado: Indica que el dispositivo Esclavo está ocupado procesando una petición emitida por el dispositivo Maestro.
- Dispositivo Transmitiendo Datos: Indica que en ese momento el dispositivo está transmitiendo datos a través de la red Modbus hacia el dispositivo Maestro.

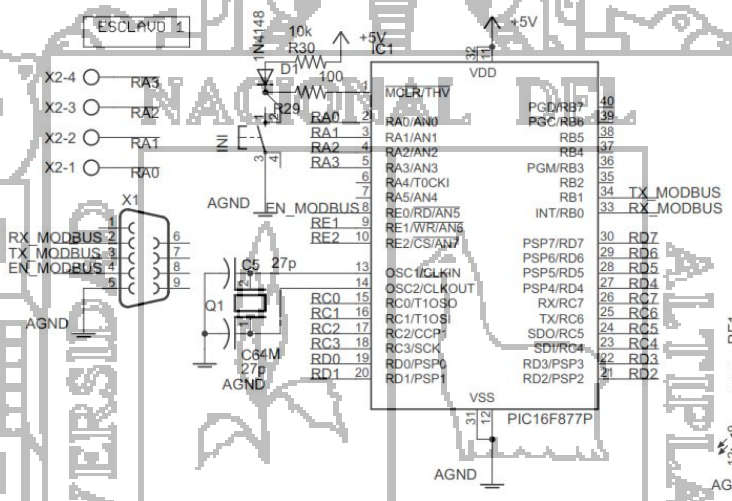



*Figura 72 - Sistema de Visualización de Estados del Dispositivo.*

### 3.2.3. Acondicionamiento de Temperatura

Este circuito está compuesto por el sensor TD5A de HONEYWELL INSTRUMENTS y el circuito de acondicionamiento diseñado para este sensor TD5A ver Figura 73.

*Figura 73 - Sensor TD5A.*

Este sensor es un RTD (aumenta su resistencia con el incremento de la temperatura).

Las principales características de este sensor son las siguientes

- Respuesta lineal (8Ω/°C).
- Sensibilidad lineal de temperatura.
- Valor de su Resistencia de 20°C es 2000Ω.
- Temperatura -40 a 150°C

Sus principales aplicaciones tenemos:

- HVAC (Calentamiento, Ventilación y Aire Acondicionado).
- Motores (Protección contra la sobrecarga).
- Procesos de control (Regula temperatura).
- Industria Automotriz (Temperatura del aire y del aceite)

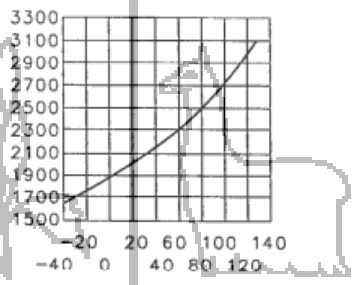



*Figura 74 - Curva Característica Sensor TD5A.*

| Temperature | | Resistance (Ω) | Temperature | | Resistance (Ω) |
|---|---|---|---|---|---|
| °C | °F | | °C | °F | |
| -40 | -40 | 1584 ± 12 (1.9°C) | +60 | 140 | 2314 ± 9 (1.1°C) |
| -30 | -22 | 1649 ± 11 (1.7°C) | +70 | 158 | 2397 ± 10 (1.2°C) |
| -20 | -4 | 1715 ± 10 (1.5°C) | +80 | 176 | 2482 ± 12 (1.4°C) |
| -10 | 14 | 1784 ± 9 (1.3°C) | +90 | 194 | 2569 ± 14 (1.6°C) |
| 0 | 32 | 1854 ± 8 (1.1°C) | +100 | 212 | 2658 ± 16 (1.8°C) |
| +10 | 50 | 1926 ± 6 (0.8°C) | +110 | 230 | 2748 ± 18 (2.0°C) |
| +20 | 68 | 2000 ± 5 (0.7°C) | +120 | 248 | 2840 ± 19 (2.0°C) |
| +30 | 86 | 2076 ± 6 (0.8°C) | +130 | 266 | 2934 ± 21 (2.2°C) |
| +40 | 104 | 2153 ± 6 (0.8°C) | +140 | 284 | 3030 ± 23 (2.4°C) |
| +50 | 122 | 2233 ± 7 (0.9°C) | +150 | 302 | 3128 ± 25 (2.5°C) |

*Tabla 9 - Valores en ohm cada 10°C.*

La ecuación de este sensor es la siguiente ver Figura 75.

$$R_t = R_0 + (3.81 * 10^{-3} * R_0 * T) + (4.94 * 10^{-6} * R_0 * T^2)$$

$R_t = Resistencia\ a\ temperatura\ R$

$R_0 = Resistencia\ a\ 0°C$

$T = Temperatura\ a\ °0C$

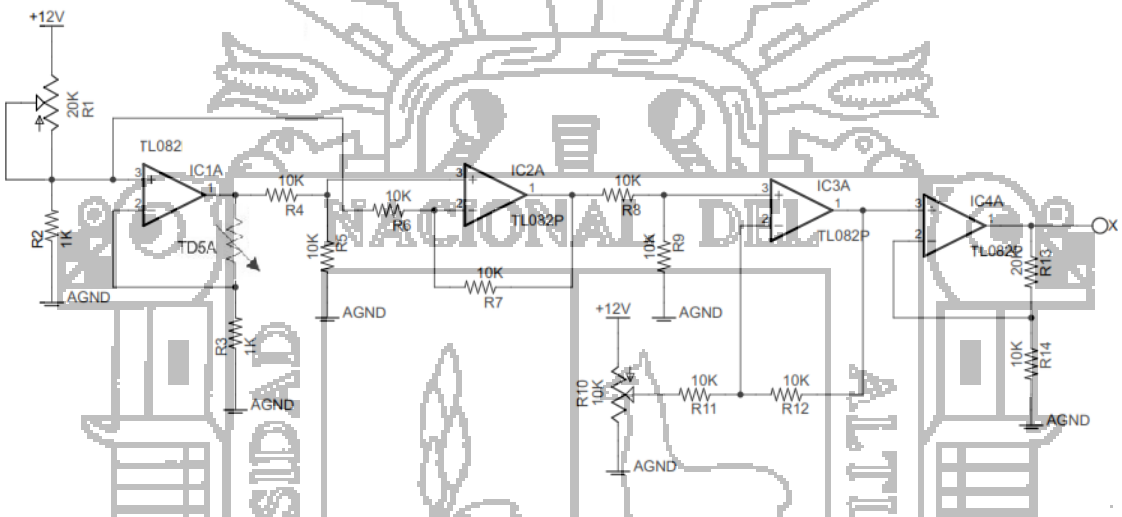



*Figura 75 - Circuito acondicionado TD5A.*

Acondicionamiento

Primera Etapa ver Figura 76.

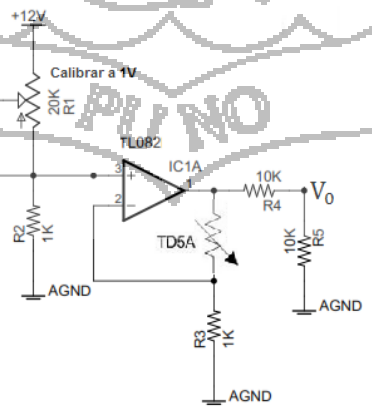



*Figura 76 - Primera etapa TD5A.*

Se observa en la figura esta configuración corresponde a un amplificador no inversor por lo tanto.

$$V_0 = V_i\left(\frac{R_2}{R_1} + 1\right) = 1\left(\frac{R_{td}}{1000} + 1\right) = 0.001R_{td} + 1$$

Se agrega la segunda etapa ver Figura 77.

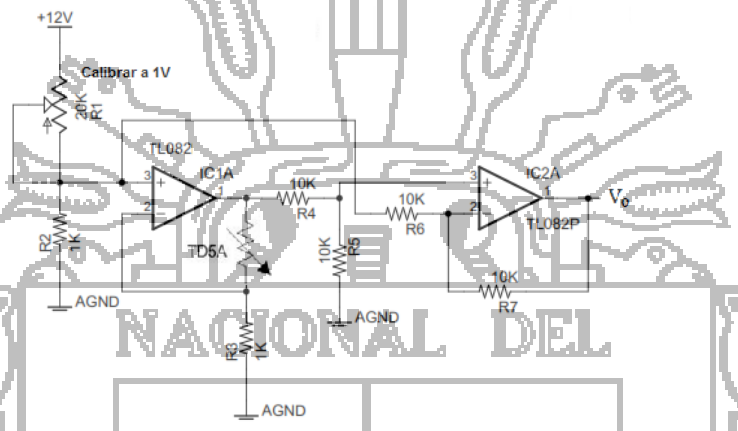



*Figura 77 - Segunda etapa TD5A.*

En la segunda etapa corresponde a un amplificador diferencial por lo tanto.

$$V_0 = V_1 - V_a = (0.001R_{td} + 1) - 1 = 0.001R_{td}$$

Agregando la tercera etapa ver Figura 78.

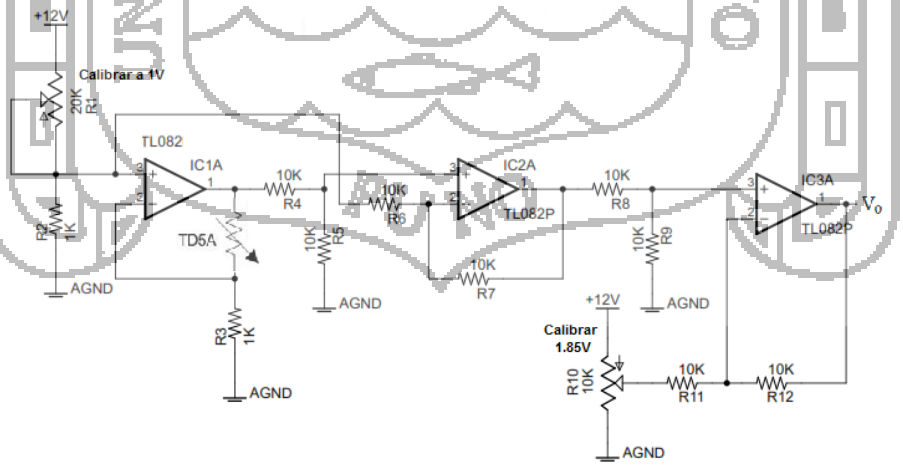



*Figura 78 - Tercera etapa TD5A.*

Esta etapa también corresponde a un amplificador diferencial lo que permite restarle el valor de la resistencia del RTD cuando está a 0°C (1854Ω).

$$V_0 = V_1 - V_a = (0.001R_{td} - 1.854)$$

Agregando la última etapa para el acondicionamiento ver Figura 79.

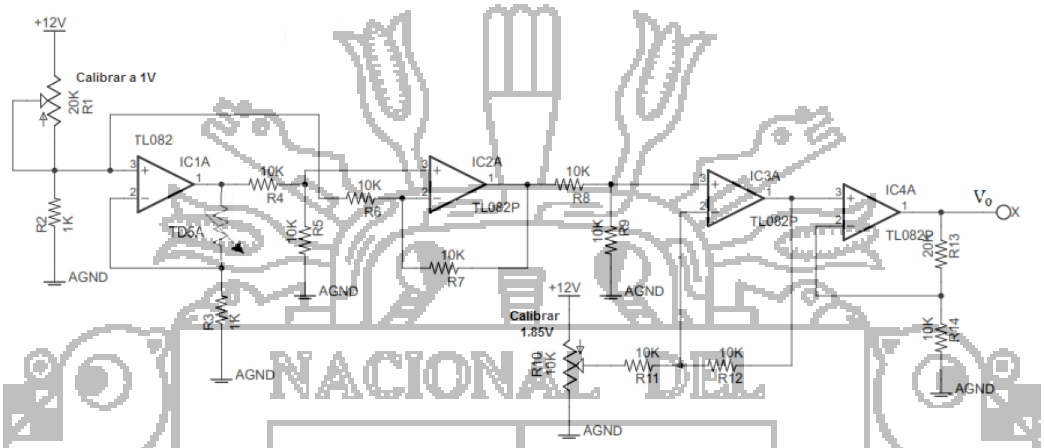



*Figura79- Cuarta etapa TD5A.*

Esta etapa se ha agregado porque como ya se restó el valor de la resistencia del RTD a 0°C (1854Ω), se va a tener una variación de 7 a 8Ω por cada °c que aumente y eso traducido a voltaje vendría a ser 7mV/°C, como el microcontrolador es que recibe estas señales pero lo hace con una resolución de 20mV para lo cual se necesita acomodar este circuito a lo que necesita el microcontrolador por lo tanto se multiplica este voltaje de entrada por 3 y obtendrá 21mV.

Se agrega un amplificador no inversor por lo tanto.

$$V_0 = V_i\left(\frac{R_{13}}{R_{14}} + 1\right) = (0.001R_{td} - 1.854) * \left(\frac{20000}{10000} + 1\right) = (0.001R_{td} - 1.854) * 3$$

Calculando para una temperatura de 20°C u como se sabe el valor de resistencia del RTD en esta temperatura es 2000Ω

$$V_0 = (0.001R_{td} - 1.854) * 3 = (0.001(2000) - 1.854) * 3$$

$$V_0 = 0.438V$$

Ahora por medio de la ecuación del sensor calculamos su valor de resistencia para 21°C

$$R_{td} = R_0 + (3.81 * 10^{-3} * R_0 * T) + (4.94 * 10^{-6} * R_0 * T^2)$$

$$R_{td} = 1854 + (3.81 * 10^{-3} * 1854 * 21) + (4.94 * 10^{-6} * 1854 * (21)^2) = 2007.54\Omega$$

Entonces

$$V_0 = (0.001R_{td} - 1.854) * 3$$

$$V_0 = (0.001 * 2007.54 - 1.854) * 3 = 0.460V$$

Si restamos ambos valores tenemos la resolución del circuito.

$$V_{res} = 0.460 - 0.438 = 22mV$$

Se observa el valor obtenido es 22mV lo cual se aproxima al valor de la resolución del microcontrolador por lo tanto se tendrá un error de 2mv.

**3.2.4. Acondicionamiento de Humedad Relativa.**

Este circuito está compuesto por el sensor de humedad HIH4000 y por su cirucito de acondicionamiento ver Figura 80.

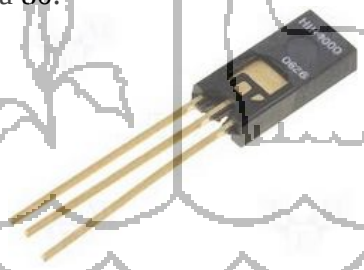


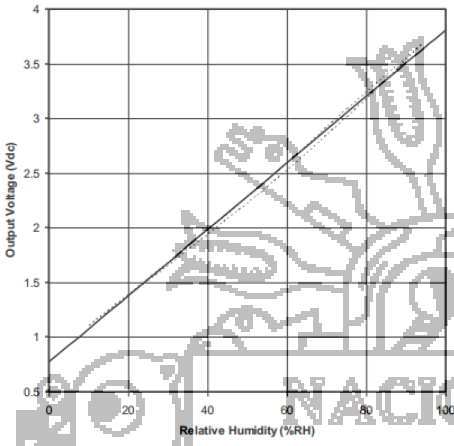
*Figura 80 - Sensor HIH4000.*

Las principales características de este sensor son:

- Voltaje de salida lineal %RH.
- Potencia de consumo baja
- Tiempo de respuesta rápido.
- Resistencia Química.
- Rango de trabajo de 0%-100%RH

Sus aplicaciones:

- Refrigeración.
- Secado.
- Meteorología.




| | |
|---|---|
| Model | HIH-4000-003 |
| Channel | 92 |
| Wafer | 030996M |
| MRP | 337313 |
| Calculated values at 5 V | |
| $V_{OUT}$ at 0% RH | 0.826 V |
| $V_{OUT}$ at 75.3% RH | 3.198 V |
| Linear output for 3.5% RH accuracy at 25 °C | |
| Zero offset | 0.826 V |
| Slope | 31.483 mV/%RH |
| RH | ($V_{OUT}$ - zero offset)/slope |
| | ($V_{OUT}$ - 0.826)/0.0315 |
| Ratiometric response for 0% RH to 100% RH | |
| $V_{OUT}$ | $V_{SUPPLY}$ (0.1652 to 0.7952) |

*Figura 81 - Curva y características eléctricas HIH4000.*

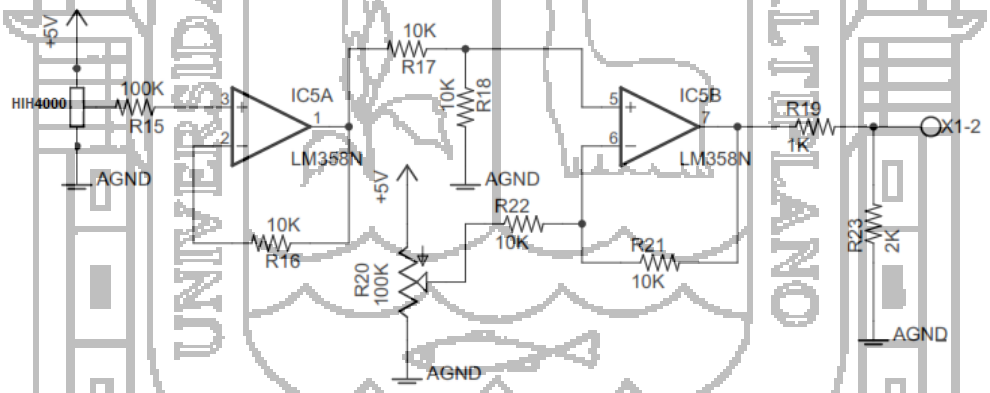



*Figura 82 - Acondicionamiento de HIH4000.*

Cálculos matemáticos ver Figura 83.

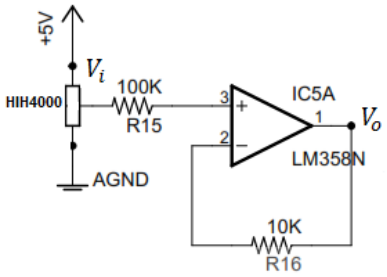



*Figura 83 - Primer etapa HIH4000.*

Este tipo de configuración es de un amplificador con ganancia 1 simplemente se a usado para que el sensor HIH4000 no nos entregue mucha corriente y esta sea brindada por medio del amplificador operacional.

Segunda Etapa ver Figura 84.

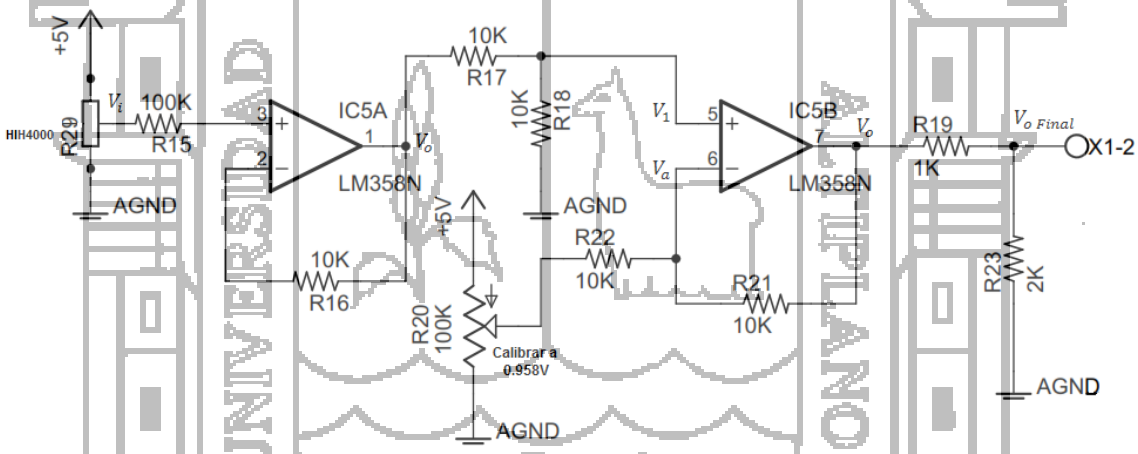



*Figura 84 - Segunda etapa HIH4000.*

Como se observa en la figura la configuración del amplificador operacional usada es la de un amplificador diferencial. Mediante esta configuración no permite restar el valor que entega el sensor a 0% RH (0.958) con el valor que sensor en el momento para así no tener un pedestal. Como se vio en la tabla anterior del HIH4000 este cuenta con una resolución de 30.68mV/%RH por lo que se ha utilizado un divisor de voltaje a la salida de la segunda etapa con el objetivo de obtener 20mV de resolución para el microcontrolador PIC16F877A.

Por lo tanto.

$$V_o = V_1 - V_a$$

$$V_o = V_{hih} - 0.958$$

Se aplica el divisor de voltaje tendremos ver Figura 85.

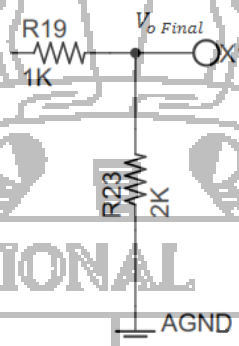



*Figura 85 - Divisor de Tensión.*

$$V_{o\ Final} = \frac{(Vhih - 0.958) * 2000}{3000} = \frac{(V_{hih} - 0.958) * 2}{3}$$

$$V_{O\ Final} = \frac{(V_{hih} - 0.958) * 2}{3}$$

Calculando el voltaje que entrega el

$$RH = \frac{(V_{out} - 0.958)}{0.0307}$$

$$V_{out} = (RH * 0.0307) + 0.958$$

$$V_{out} = (30 * 0.0307) + 0.958 = 1.879V$$

Calculando cuando el voltaje ingresa a la segunda etapa

$$V_o = V_1 - V_a$$

$$V_o = 1.879 - 0.958 = 0.924V$$

Haciendo el divisor de voltaje se obtiene el voltaje de salida final del circuito

$$V_{o\,Final} = \frac{V_o * R_{23}}{R_{19} + R_{23}} = \frac{(0.921) * 2000}{3000} = 0.614V$$

Calculando el voltaje que entrega el sensor a 31% RH con su ecuación.

$$RH = \frac{(V_{out} - 0.958)}{0.0307}$$

$$V_{out} = (RH * 0.0307) + 0.958$$

$$V_{out} = (RH * 0.0307) + 0.958 = (31 * 0.0307) + 0.958 = 1.909V$$

Calculando cuando el voltaje ingresa a la segunda etapa

$$V_o = V_1 - V_a = 1.9097 - 0.958 = 0.9517V$$

Haciendo el divisor se obtiene el voltaje de salida final del circuito.

$$V_{o\,Final} = \frac{V_o * R_{23}}{R_{19} + R_{23}} = \frac{(0.9517) * 2000}{3000} = 0.6345V$$

Para calcular la resolución del circuito restamos los $V_{O\,final}$ a 31%/RH de $V_{O\,final}$ a 30%RH

$$V_{res} = 0.6345 - 0.614 = 20.4mV$$

Como se puede observar la resolución obtenida es de 20.4mV lo cual nos da un error de 0.4mV.

### 3.2.5. Acondicionamiento de presión atmosférica

El sensor de presión atmosférica utilizado es el MPX4115A de la Casa Motorola, proporciona un voltaje de salida compensado por temperatura y está compuesto por un

transductor piezo-resistivo monolítico de silicio que genera una señal de salida precisa que es proporcional a la presión aplicada.

Este tipo de sensores requiere una alimentación de +5 Voltios, están compensados para variaciones debidas a la temperatura y entregan una salida lineal, con una exactitud de ± 1.5% para temperaturas comprendidas entre 0 y 85 ºC a escala completa se medirá la presión del medio ambiente que se aplica a un solo cuerpo con respecto a una

La salida del sensor tiene un margen de escala de 0.2 a 4.8 V. Además el sensor de presión entrega a la salida niveles compatibles con él A/D del uC.

La resolución proporcionada por el convertidor A/D de 12 bits con las referencias de voltaje bajo y alto de 0 y 5 volts respectivamente ver Figura 86.

*Figura 86 - Sensor MPX4115A.*

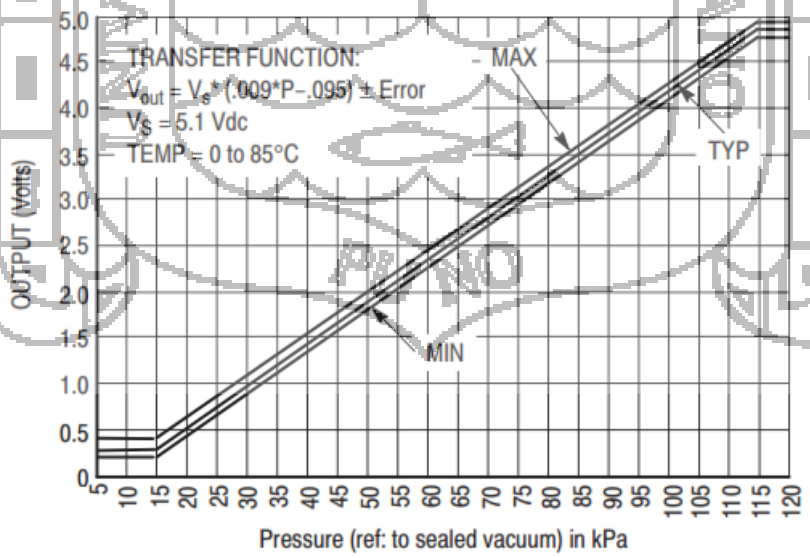



*Figura 87 - Curva y características eléctricas MPX4115A.*

Así se tiene que la función de transferencia para este sensor:

$$V_s = V_{cc}(0.009(\text{Presión}) - 0.095)$$

Dónde: $V_s$ Es la tensión de salida del sensor.

$V_{cc}$ Es la tensión de alimentación del sensor

Presión Es la presión barométrica aplicada al sensor [KPa]

Si se utiliza una fuente de alimentación de 5 V, con lo cual nuestra ecuación se reduce a:

$$V_s = (0.045(\text{Presión}) - 0.475)$$

Si tenemos una presión de 15KPa, la tensión de salida del sensor de presión cuando está alimentado con 5 V será de:

$$V_s = (0.045(\text{Presión}) - 0.475)$$
$$V_s = (0.045(15) - 0.475) = 200mV$$

En la ecuación anterior se observa que la tensión del sensor variará 200mV por cada KPa pues el término 0.475v es constante, es decir, por cada KPa a partir de 15KPa habrá un incremento

El sensor MPX4115A tiene una exactitud de +/- 0.1125 KPa, cuando esta calibrado ver Figura 88.

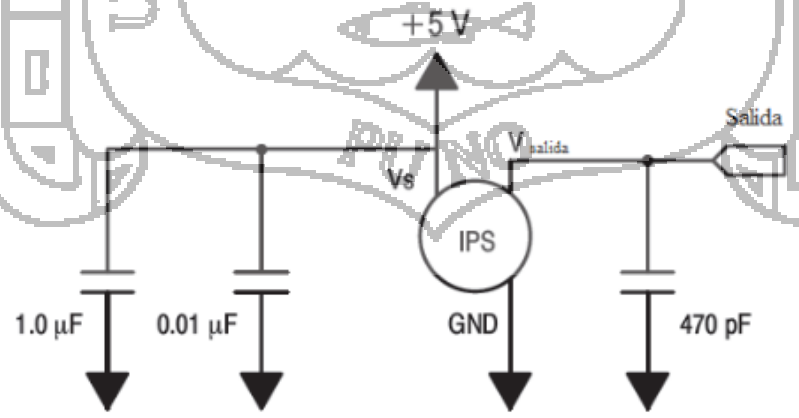



*Figura 88 - Acondicionamiento sensor de presión.*

En la figura anterior se muestra un circuito filtro de desacoplamiento típico para impedir las interacciones entre la conexión de salida del sensor de presión a la entrada del A/D del uC.

### 3.2.6. Acondicionamiento de Velocidad de viento.

#### 3.2.6.1. Encoder y Fotoreceptor

Necesitamos medir la velocidad del viento entonces, acoplarle al medidor un encoder, el encoder debe ser de la mayor cantidad de ranuras posible. Se acoplo un encoder de 20 ranuras ver Figura 89.

*Figura 89 - Encoder de 20 ranuras.*

Nuestro transductor debera poder leer esta frecuencia maxima, ademas de funcionar a baja potencia 5 Vdc.

El transductor seleccionado es el H21A1, cuya frecuencia de lectura, supera largamente la requerida ver Figura 90.

*Figura 90 - Fototransistor H21A1.*

El transductor a utilizar es un fototransistor o interruptor óptico este interruptor debe estar situado, de tal forma que la rueda con las divisiones pase entre medio de sus dos "paredes", con esta transductor, el led emisor envía un haz de luz que incide sobre el fototransistor, si entre estos dos componentes aparece un elemento opaco, provocará un cero a la salida entre terminales del optoacoplador. Esta salida es conectada a un circuito conversor a TTL, el cual nos proporcionara los impulsos para la entrada del contador del microcontrolador la función del circuito acondicionador es enviar los impulsos al microcontrolador se utiliza este circuito, debido a que sin su utilización los voltajes de salida del optoacoplador no se adaptan a los "1" y "0" Lógicos del microcontrolador. De esta forma nos aseguramos que siempre aya un +5V para un "1", lógico y 0V para el "0" ver Figura 91.

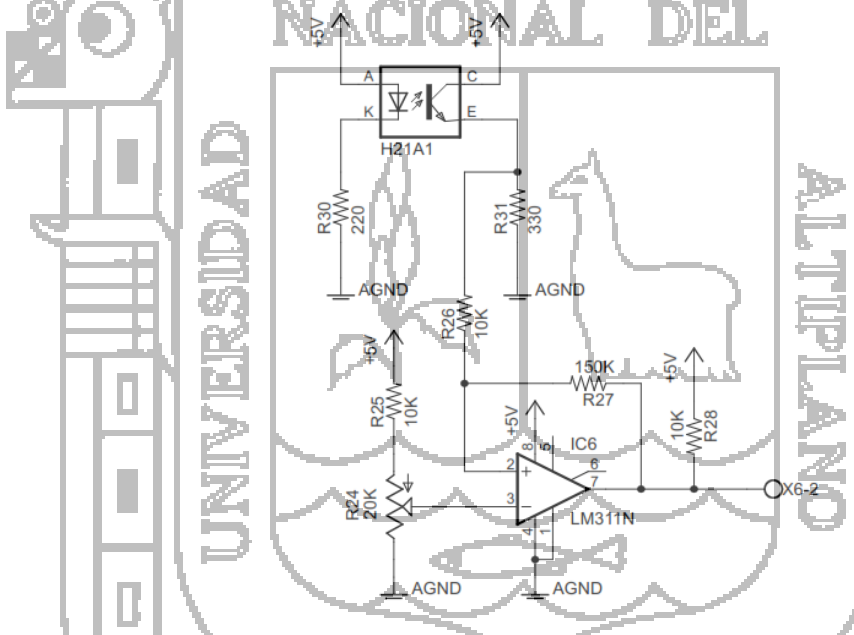



*Figura 91 - Acondicionamiento del encoder.*

Dentro de las características principales tenemos:

- Anchura de ranura 5mm
- Profundidad 8mm
- Tiempo de ascenso descenso: 13/17uS ( lectura en el orden de MHz)
- Voltaje nominal fototransistor 5V (Ic:2mA)

- Voltaje nominal led 5V (Id=20mA)

### 3.2.7. Diseño del Programa del Microcontrolador

El microcontrolador se encarga de monitorear constantemente cada uno de sus puertos y actualizar sus valores en los registros correspondientes. Además, se encarga de monitorear el puerto serial y llevar a cabo el procesamiento de los datos cuando sea necesario.

Cuando el microcontrolador recibe datos a través del puerto serial, este espera alrededor de 26ms antes de procesar los datos obtenidos. Este tiempo de espera, se utiliza para determinar el envío de una "trama petición completa" tal y como lo especifica el protocolo Modbus, la cual se evalúa posteriormente, con el fin de determinar si efectivamente la trama obtenida corresponde a la trama enviada por el dispositivo Maestro.

Luego de determinar la integridad de la trama, el microcontrolador determina si la trama enviada fue dirigida hacia ese dispositivo Esclavo, y de ser así realiza las operaciones solicitadas y se encarga de emitir la respuesta correspondiente. De lo contrario, simplemente desecha la trama y sigue realizando sus otras tareas.

#### 3.2.7.1. Estructura del Programa Principal

El diseño del software se compone de una estructura principal. Cada una de estas rutinas realiza una tarea diferente que se ejecuta de acuerdo a los requerimientos de Software mencionados del módbus Esclavo.

- Rutina de inicialización: en la cual se lleva a cabo la inicialización de variables, espacio de memoria y configuración de las librerías; esta rutina solo se ejecuta una vez en el programa principal.
- Rutina Configuración del Módulo Esclavo: que usa la información que se suministra a través de la interfaz de usuario con la configuración de los parámetros del dispositivo Esclavo. A partir de aquí se inicia un ciclo infinito en el programa principal.

- rutina de adquisición de datos: la cual sirve para leer los datos de entrada Análogos-Digitales del sistema, cuyos datos son adecuados en un formato real y didáctico para que cuando exista una solicitud estén listos para ser enviados.

Enseguida son enviadas al registro de recepción serial, si no hay ninguna interrupción serial el programa principal salta hasta "la rutina Configuración módulo Esclavo" y continua por las demás rutinas siguientes hasta volver a consultar el registro de recepción serial quedándose en un ciclo infinito hasta que ocurra la interrupción serial, en caso que la interrupción ocurra se ejecuta la

- Rutina Modbus: en la cual se recibe la solicitud modbus, se procesa y se envía una respuesta de acuerdo a la solicitud.

Una vez la rutina se haya ejecutado el programa principal regresa a "la rutina Configuración del módulo Esclavo" creando otro ciclo de programa infinito ver Figura 92.

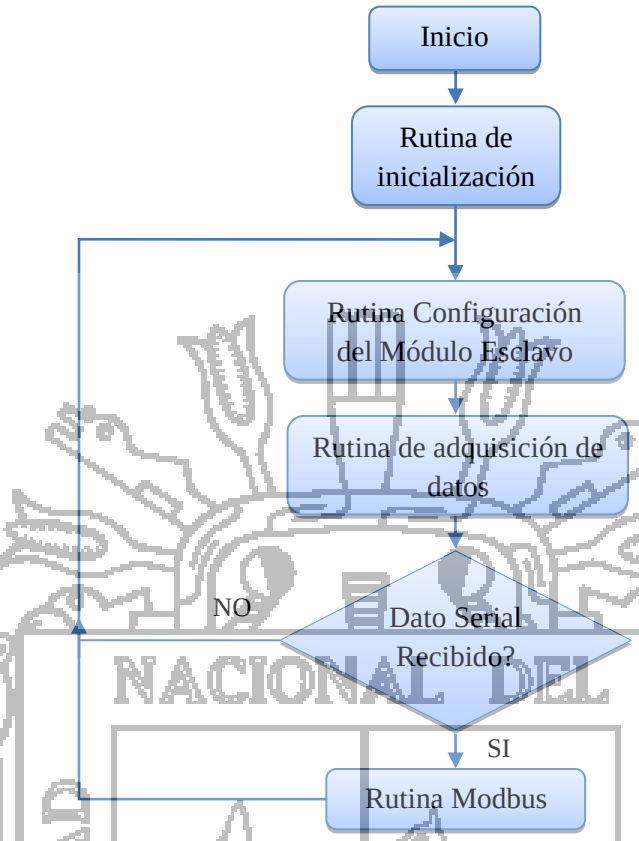



*Figura 92 - Diagrama de flujo de Programa principal.*

***3.2.7.1.1. Rutina de inicialización***

Esta rutina solo se ejecuta cada vez que se energiza los esclavos, en esta misma se lleva a cabo tareas tales como configurar puertos de Entradas-Salidas, configuración del puerto serial (velocidad, modo de comunicación), configuración de contadores, interrupciones y espacios de memoria. ver Figura 93.

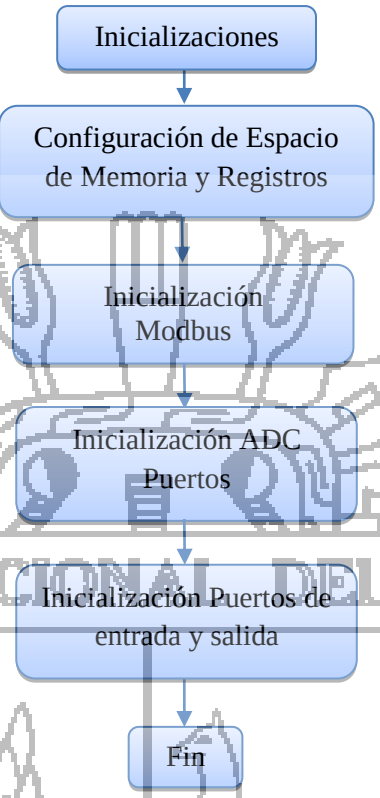



*Figura 93 - Diagrama de Flujo Rutina Inicializaciones.*

- Configuración de Espacio de Memoria y Registros

  En esta subrutina se lleva a cabo la configuración e inicialización de los registros y espacios de memorias los cuales son usados en las rutinas del programa principal se muestra en la siguiente tabla y las variables utilizadas ver Tabla 10.

| Variable | Función |
|---|---|
| input_regs[] | Variable tipo chart en la cual se almacena el valor de las 2 señales análogas una vez se haya hecho la adecuación, para después ser utilizada en la solicitud *Modbus.* |
| inputs | Variable tipo int8 la cual almacena el estado real de las 8 entadas digitales una vez se le haya hecho la adecuación, para después ser utilizada en la solicitud *Modbus.* |
| input_regs[] | Variable tipo chart en la cual se almacena el valor para ser escrito, para después ser utilizada en la |
| coils | Variable tipo int8 la cual almacena el estado real de las 8 salidas digitales, para después ser utilizada en la solicitud *Modbus.* |

*Tabla 10 - Variables de Programa.*

- Inicialización Modbus

  Rutina que permite la inicialización y configuración del microcontrolador utiliza la directiva utilizando la directiva #define, aquí se definen algunos parámetros para el funcionamiento del software de los esclavos ver Tabla 11.

| Directiva | Función |
|---|---|
| MODBUS_SERIAL_TYPE MODBUS_RTU | Directiva para el modo de transmisión en modo ASCII o RTU, para después ser utilizada. |
| MODBUS_TYPE MODBUS_TYPE_SLAVE | Directiva para el tipo esclavo, para después ser utilizada. |
| MODBUS_SERIAL_RX_BUFFER_SIZE | Directiva para la recepción de la trama modbus |
| MODBUS_SERIAL_ENABLE_PIN | Directiva para la habilitación del driver MAX485 utilizando el PIN_E0 |
| MODBUS_SERIAL_TX_PIN | Directiva para transmisión de la trama modbus asignando el pin PIN_B1. |
| MODBUS_SERIAL_RX_PIN | Directiva para recepción de la trama modbus asignando el pin PIN_B0. |

*Tabla 11 - Asignación de parámetros modbus.*

- Inicialización ADC Puertos

  En esta subrutina se configuran las entradas análogas, para poder gestionar estas funciones desde e maestro se tendrá que identificar las direcciones de memoria estándar del protocolo ModBus con las correspondientes direcciones de memoria específicas de este microcontrolador mediante siguiente mapa de memoria ver Tabla 12:

| Dirección de memoria | Descripción | Rango de Medida |
|---|---|---|
| 30001 | Entrada Analogica 1 PIN_A0 | 0…5V |
| 30001 | Entrada Analogica 1 PIN_A1 | 0…5V |

*Tabla 12 - Asignación puertos modbus.*

- Inicialización Puertos de entrada y salidas

  Rutina para la configuración de puertos, se asigna el puerto D como entrada para las 8 entradas digitales y se define el puerto C como salidas digitales.

### *3.2.7.1.2. Rutina Configuración del Módulo Esclavo*

En la Figura 94 se muestra esta rutina se modifican parámetros del programa según la información suministrada mediante la interfaz de usuario, se configura dos parámetros: La dirección del Esclavo y la velocidad de transmisión ver Tabla 13.

| Directiva | Función |
|---|---|
| MODBUS_ADDRESS | Parámetro que se le ingresa la dirección del esclavo |
| MODBUS_SERIAL_BAUD | Variable tipo chart que se modifica para configurar la velocidad de transmisión del Esclavo mediante la interfaz de usuario, los valores de velocidad son (*1200-2400-4800-9600-19200bps*) |

*Tabla 13 - Configuración del Módulo Esclavo.*

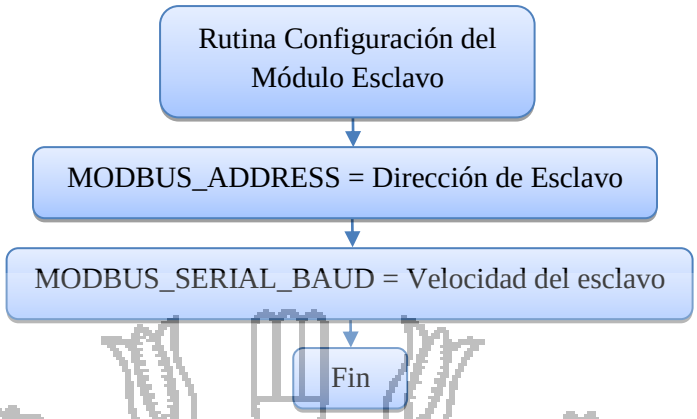



*Figura 94 - Diagrama de flujo configuración esclavo.*

***3.2.7.1.3. Rutina de adquisición de datos***

En este bloque se llevan a cabo las tareas de adquirir los datos de entrada provenientes de las señales del sistema que se requiere monitorear y se almacenan en variables para ser usados en una solicitud *Modbus* ver Figura 95.

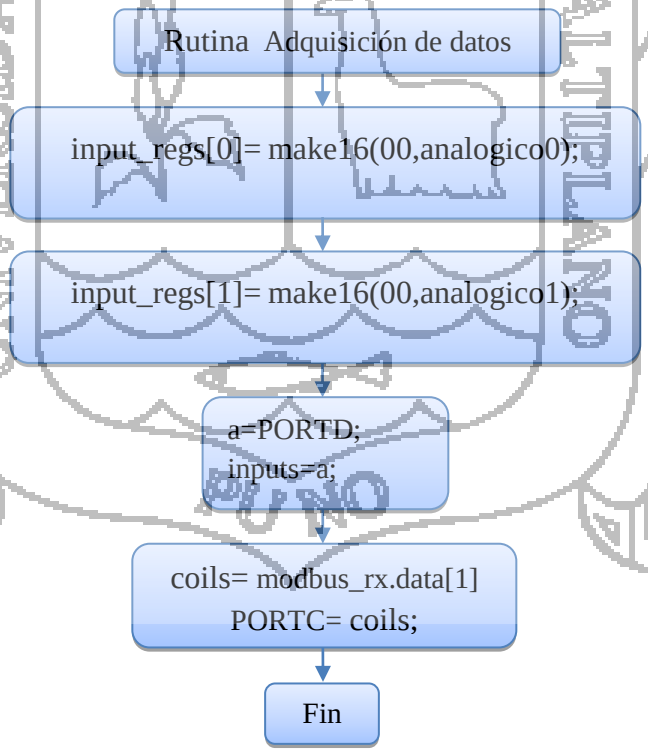



*Figura 95 - Diagrama de flujo Rutina de adquisición de datos.*

### *3.2.7.1.4. Rutina Modbus*

Cada vez que haya una interrupción serial se ejecuta esta rutina que es la encargada de armar la trama con los datos recibidos serialmente y analizar su validez, además responde a la solicitud requerida y prepara los datos para construir una trama de respuesta que es enviada al Maestro *Modbus* ver Figura 96.

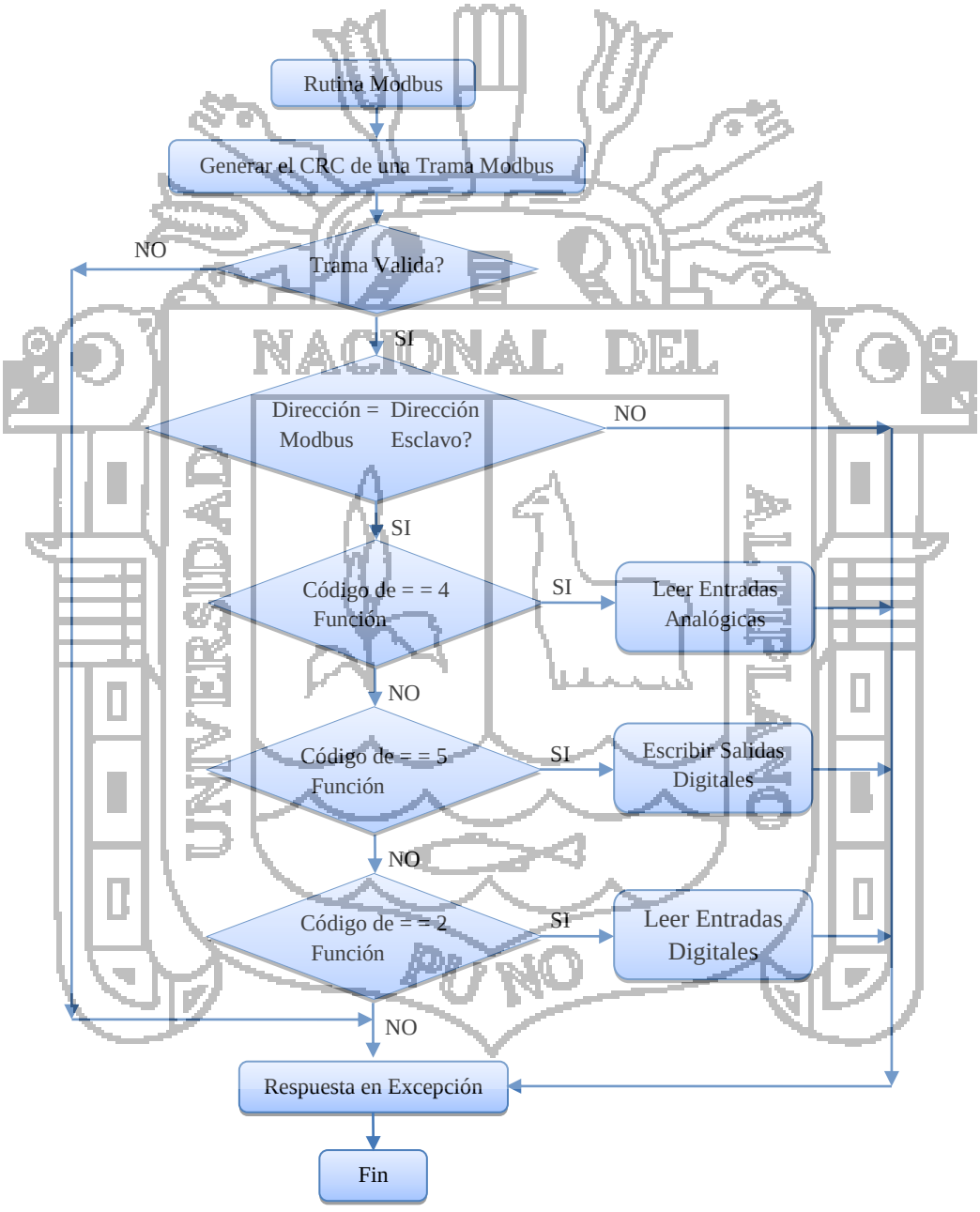



*Figura 96 - Diagrama de flujo Rutina Modbus.*

- **Generar el CRC de una Trama Modbus**

La generación del CRC de una trama de petición o respuesta Modbus, inicialmente una variable que se incrementara en una unidad *(i)* luego se carga el registro de dos bytes que va a contener el valor del CRC *(modbus_serial_crc. b[]=0xFF)*, es decir se ponen en 1 todos los bits del registro. Además, carga un registro auxiliar, denominado (*byte_count)*, con el valor correspondiente al número de bytes que contiene la trama a procesar y una variable que contiene el dato *(data)* siguientes, mientras que el valor del registro (*byte_count)* sea mayor que cero, se llevan a cabo las siguientes operaciones como se muestra en Figura 102.

- Se realiza la operación XOR entre el (*modbus_serial_crc.b[])* y el registro que contiene el dato *(byte_count),* y se guarda el resultado en (*uIndex)*.
- Se crea un ciclo que va desde 0 hasta 7 con incremento unitario, en el que se comprueba si el bit menos significativo (LSB) de (*uIndex)* es igual a 1. De ser así, se rota a la derecha, a nivel de bits, el contenido de (*uIndex)*, se pone en 0 el bit más significativo (MSB) de (*uIndex)* se realiza operación XOR entre y el valor hexadecimal el polinomio Modbus 0xA001 luego se almacena el resultado en (*modbus_serial_crc. b[])*.

  En caso contrario, es decir, si el LSB de (*uIndex)* es igual a 0, simplemente se rota a la derecha, a nivel de bits, el contenido de (*uIndex)* se pone en 0 el bit más significativo (MSB) de (*uIndex)*.
- Se incrementa el valor del índice (*i)* y decrementar el valor de (*byte_count)*. Cuando el valor de (*byte_count)* es igual a 0, se ha logrado procesar toda la trama, por lo que simplemente se retorna el valor de *(data)* y la subrutina llega a su fin.

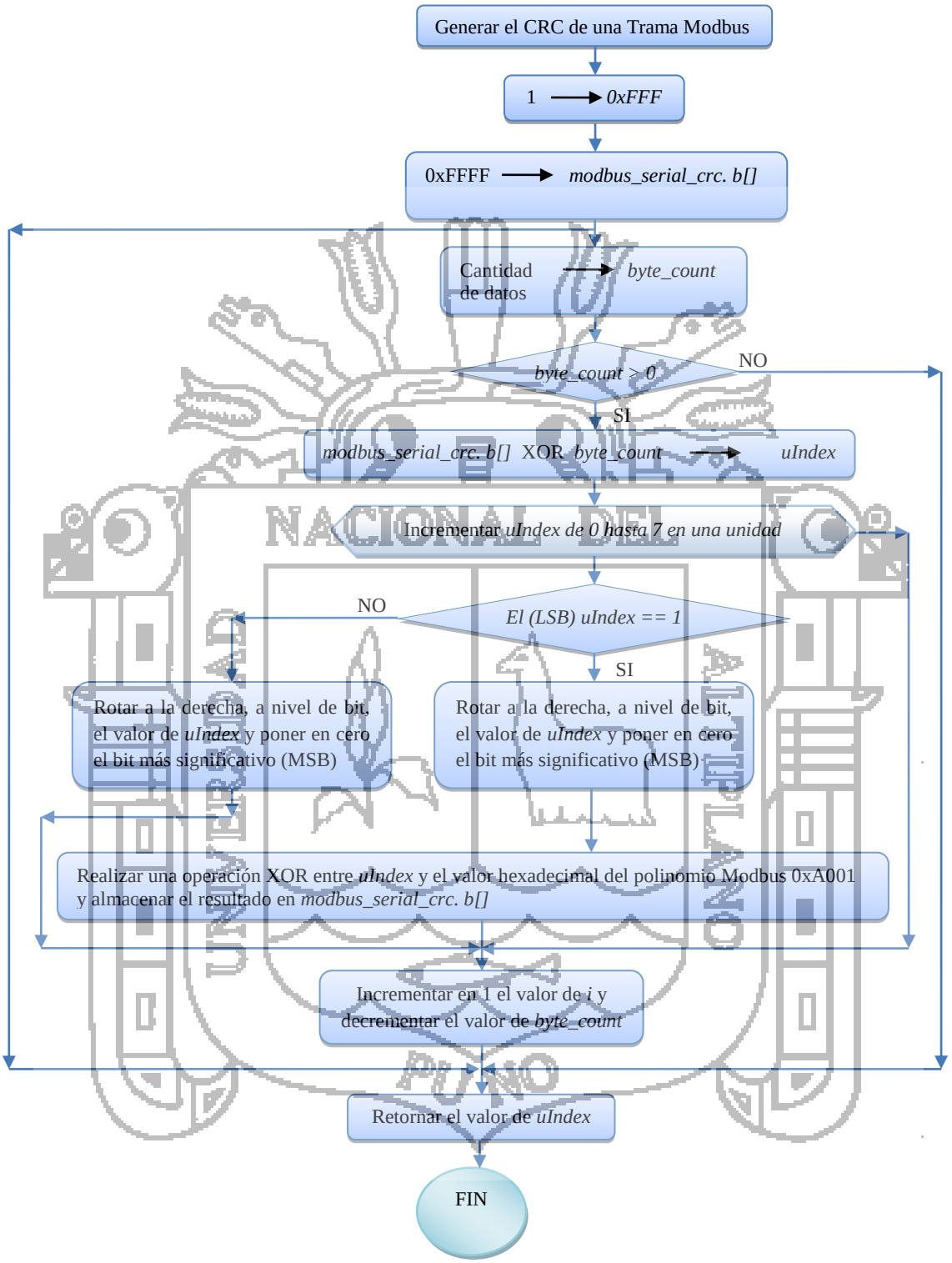



*Figura 97 - Diagrama de flujo de Generar el CRC.*

- **Dirección *Modbus***

  Se consulta si la dirección *Modbus* que se encuentra en la trama coincide con la dirección del Esclavo la cual ha sido previamente asignada, en caso de que no sean iguales se asume que la trama no es para este Esclavo y se descarta.

- **Funciones**

  Se consulta en la trama recibida qué función se debe realizar, que en este caso solo pueden ser la función 4, 2 y la función 1.

- **Respuesta en excepción**

  Se envía una respuesta al Maestro en caso de alguna inconsistencia por que la trama no es válida y/o función no soportada.

### 3.2.7.2. Códigos de Función Modbus

Para poder usar el hardware del módulo por medio del protocolo Modbus es preciso establecer una relación entre las funciones y las entradas del módulo, dado que este es un dispositivo de solo entradas análogas-digitales y digitales se utilizaron funciones del protocolo Modbus que se detallaran a continuación.

#### *3.2.7.2.1. Función 4(0x04) lectura de entradas análogas*

La Figura 98 muestra función 04 es una función para la adquisición de datos de entrada en que los datos se registran de 16 *Bits*, por medio de esta función se lee el valor que hay en las entradas análogas del módulo, cuya variación de 1 a 5 se digitaliza con valores discretos entre 0 a 255, puesto que el *ADC* es de 8 *Bits*. Luego se escalan de 0-100% se utilizó el código fuente que se detalla en el Anexo II.

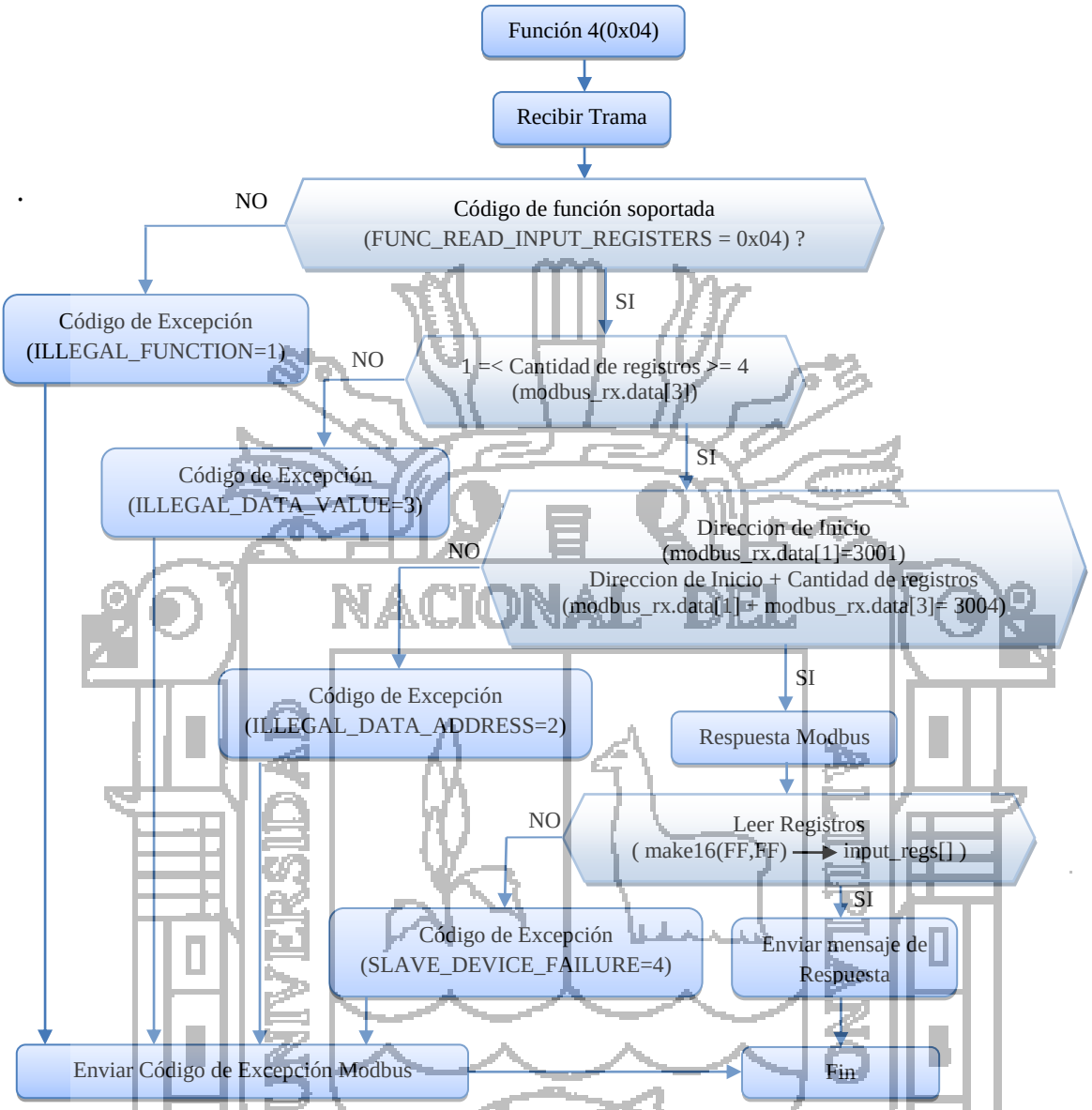



*Figura 98 - Diagrama de flujo de Función 4(0x04).*

### *3.2.7.2.2. Función 02(0x02) Lectura de Entradas Digitales*

Esta función captura la información del estado de entradas de datos discretos, cada entrada necesita un espacio de memoria de un *Bit*, el cual representara los estados *ON* (activa) u *OFF* (apagada).

La lectura del estado de una entrada digital por medio de esta función se lleva a cabo leyendo el *Bit* de la correspondiente entrada digital ver Figura 99.

Para mejorar el rendimiento se ha decidido implementar un registro que permita conocer el estado de las 8 entradas digitales por medio de esta función, de esta forma, de una sola petición se adquiere información de todo el estado de todas las entradas digitales se utilizó el código fuente que se detalla en el Anexo II.

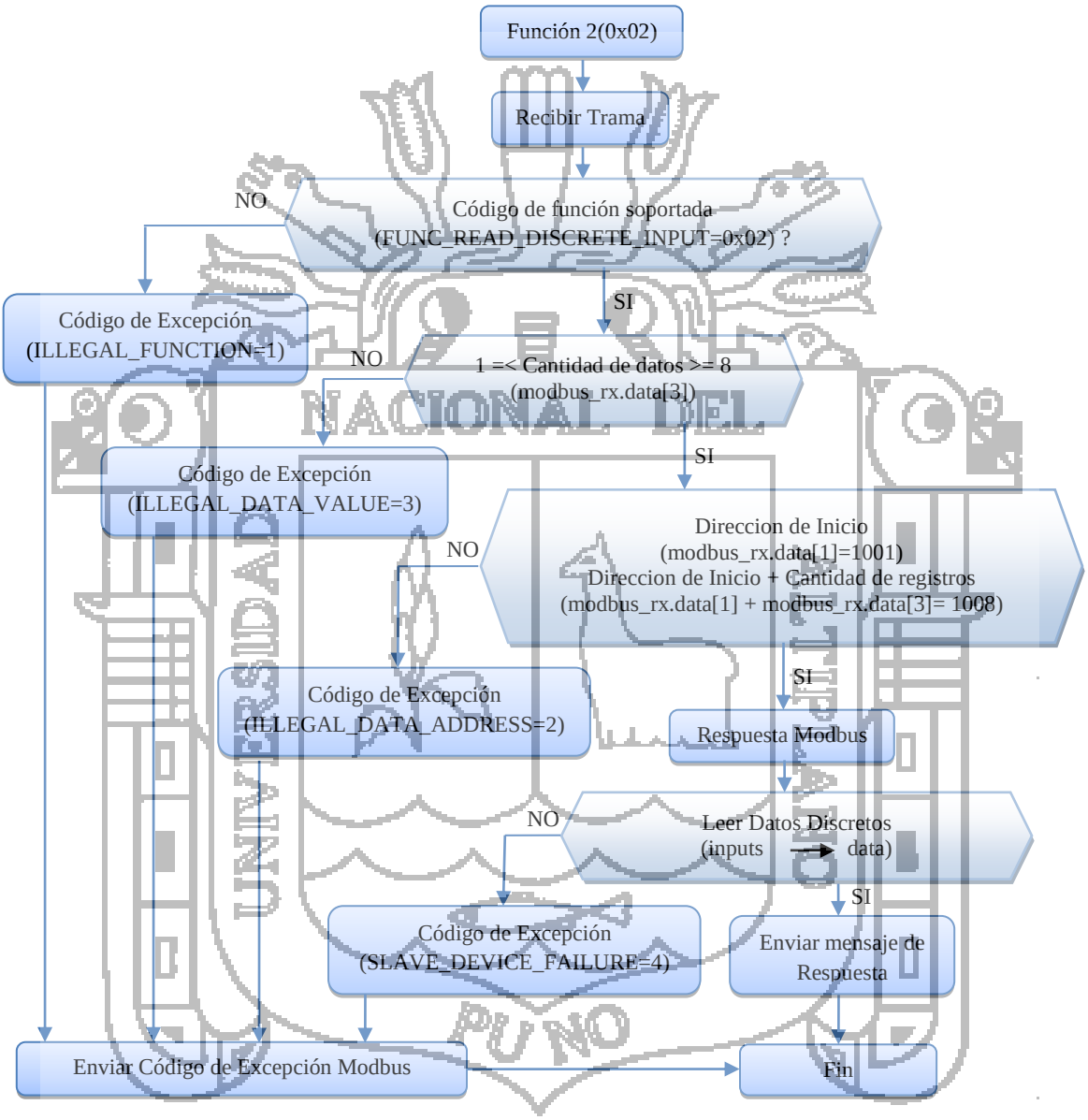



*Figura 99 - Diagrama de flujo de Función 02(0x02).*

### *3.2.7.2.3. Función 05(0x05) Escritura de Salidas Digitales*

Se puede observar en el diagrama de flujo que el esclavo debe verificar que en el campo de datos esta el valor hexadecimal 0000 o 00FF para poder forzar una bobina al estado ON o OFF ver Figura 100.

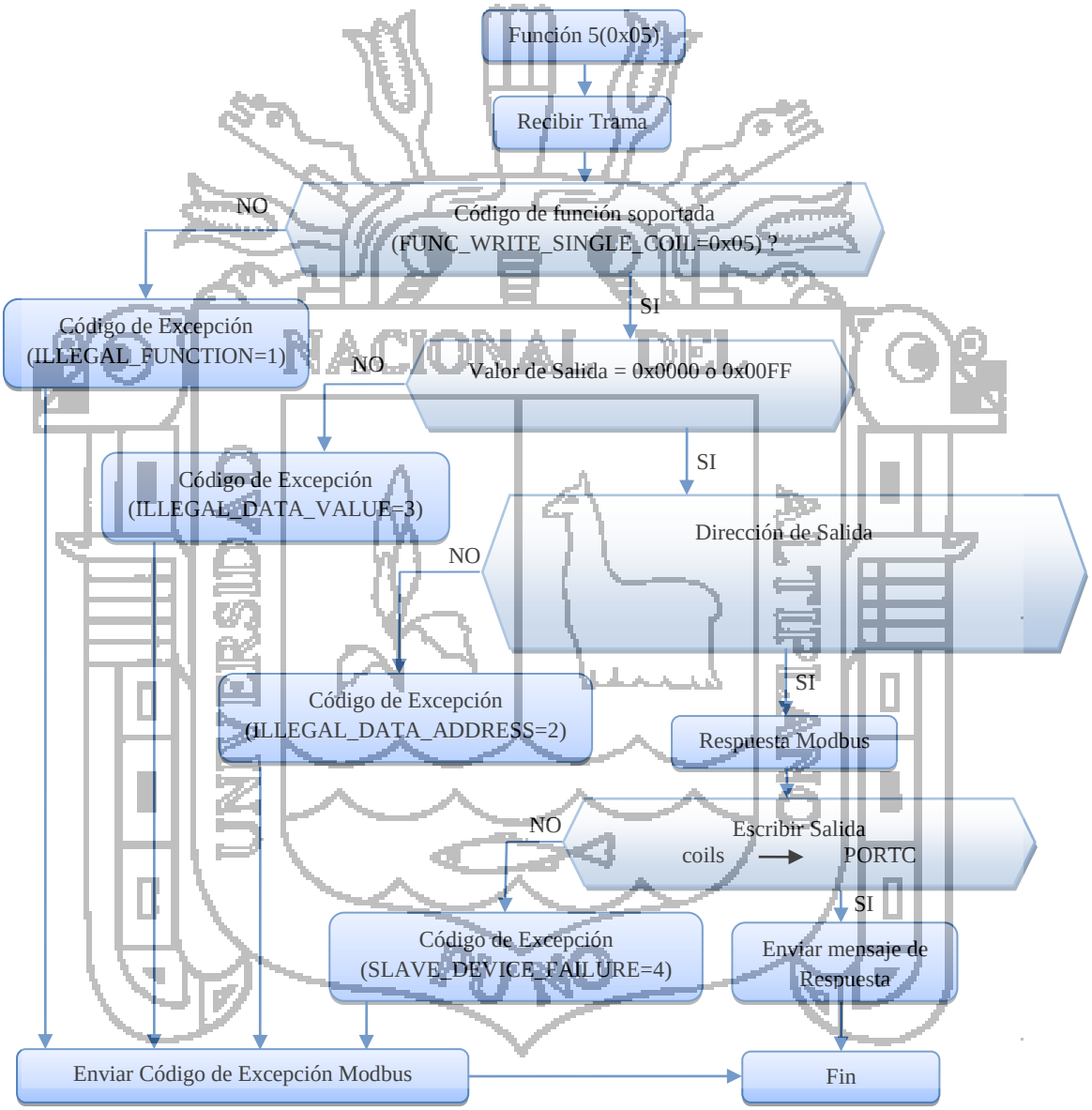



*Figura 100 - Diagrama de flujo de Función 05(0x05).*

### 3.3. Desarrollo del Terminal Maestro

En esta sesión se desarrolla el programa que reside en el computador y que se encargara de la descarga y la representación de los datos provenientes de la estación meteorológica en donde se implementar el maestro modbus.

#### 3.3.1. OPC SERVER de National Instruments.

Los Servidores OPC de National Instruments ofrecen una sola interfaz consistente con múltiples dispositivos, para no tener que aprender nuevos protocolos de comunicación. La combinación de los Servidores NI OPC y LabVIEW brinda una sola plataforma para brindar medidas y control de alto rendimiento a sistemas industriales nuevos y existentes en sistema HMI/SCADA con PLCs, PACs y sensores inteligentes. Tiene por características principales las siguientes ver Figura 101.

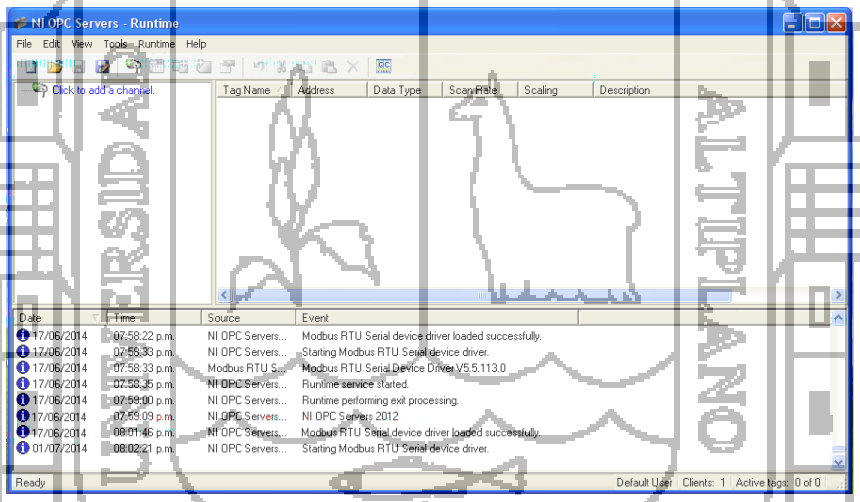



*Figura 101 - Entorno pantalla principal de NI-OPC server.*

- Una sola interfaz consistente para maximizar OPC
- Añade múltiples complementos de controlador en un solo servidor OPC
- Diagnósticos OPC para rápida depuración
- Incluye controladores para PLC legado basado en serial hasta lo último en PLC basado en Ethernet.

### 3.3.1.1. Configuración y uso de adquisición de datos en NI-OPC Server

Una vez abierto el programa NI-OPC Server, en el explorador de dispositivos nos aparece un mensaje el cual nos indica añadir un nuevo canal, damos click en add a Channel en el explorador de dispositivos y nos aparece la ventana en la cual debemos escribir el nombre del canal por medio del cual el dispositivo que vamos a adjuntar va a comunicarse y luego presionamos Next ver Figura 102.

Nuestro canal se va a llamar master_modbus.

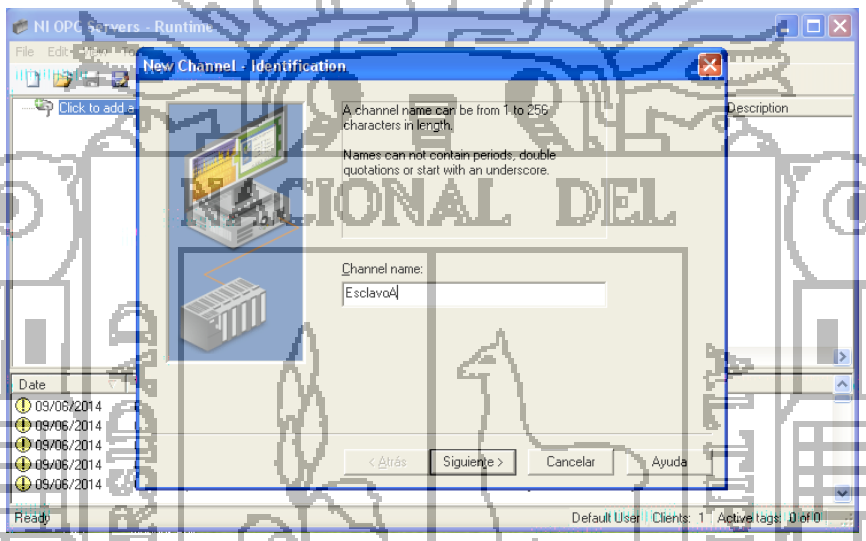



*Figura 102 - Canal en el OPS Server.*

Aparece la ventana para seleccionar el driver del tipo de dispositivo que vamos a utilizar, en la lista aparece un sinnúmero de drivers para protocolos de comunicación y varias marcas de.

Para nuestro caso vamos a utilizar el driver de Modbus RTU Serial, además dispone de drivers para Modbus Plus y Modbus ASCII.

Luego de elegir el driver Modbus que vamos a utilizar, damos click en Next para poder continuar con la configuración ver Figura 103.

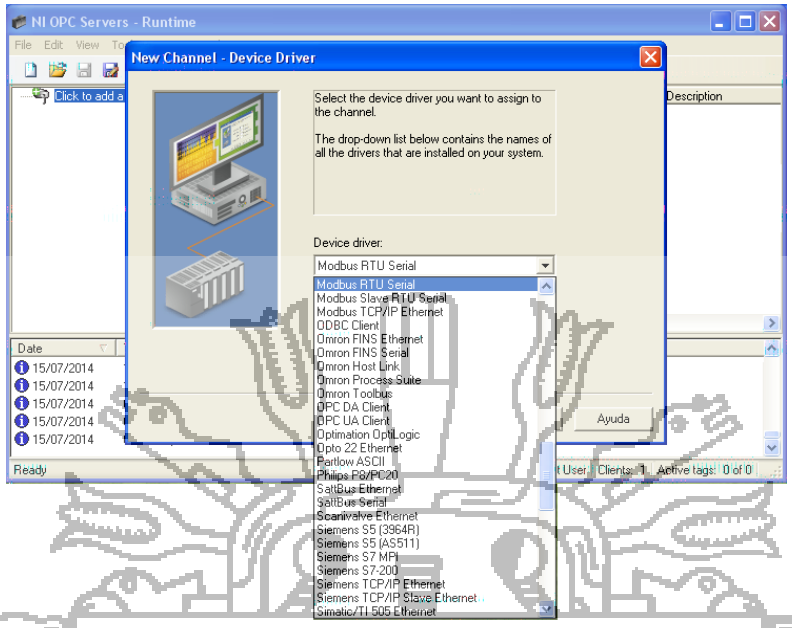



*Figura 103 - Driver para asignar el canal.*

Para la configuración se tienen que seguir estos pasos ver Figura 104.

- Asignamos el puerto COM1
- Establecemos Paridad
- Número de bits de datos
- Cantidad de bits de parada
- Control de flujo

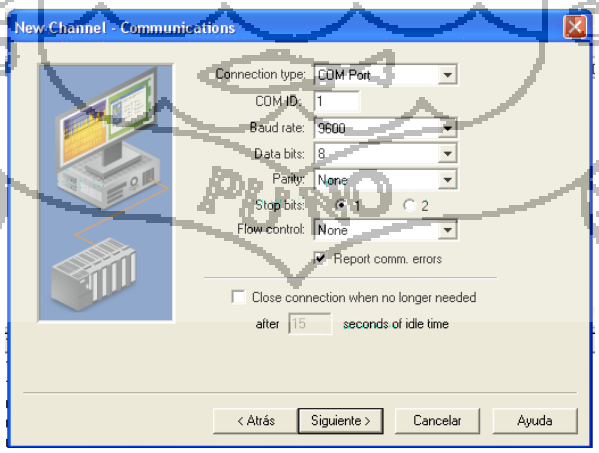



*Figura 104 - Configuración para la comunicación serial.*

La siguiente ventana nos muestra las configuraciones que antes realizamos para el canal que estamos creando. Aceptamos dando click en Finalizar ver Figura 105.

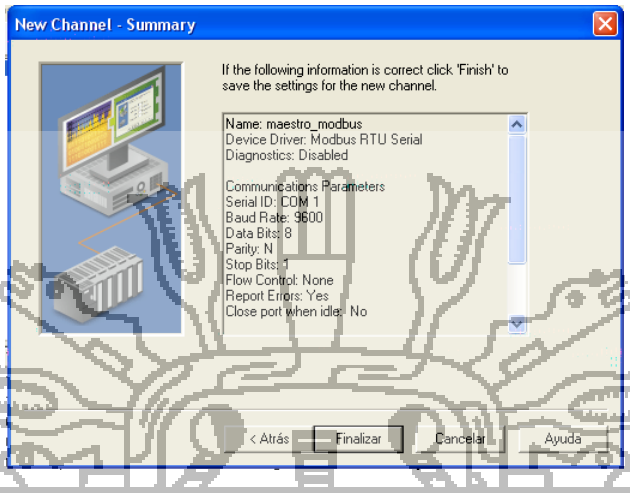



*Figura 105 - Configuración serial.*

Ahora en el explorador de dispositivos se ha creado nuestro Módulo master_modbus, el cual muestra una opción que indica crear un nuevo dispositivo ver Figura 106

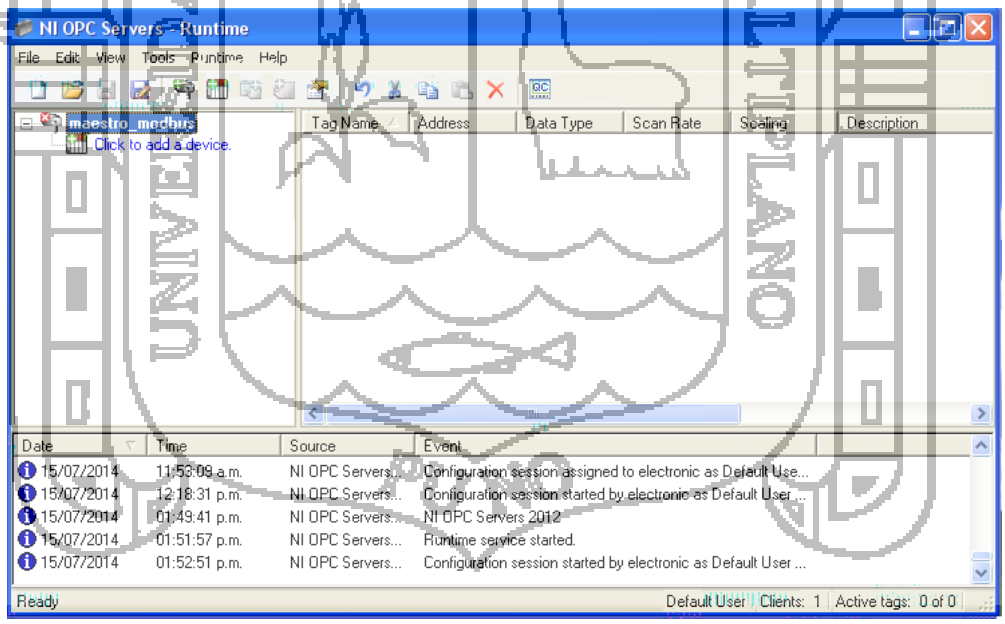



*Figura 106 - Entorno pantalla principal de NI-OPC server.*

Agregamos un nuevos dispositivo que vamos a adjuntar se llamará EsclavoA y EsclavoB, este será el módulo de adquisición de datos que trabajará como Modbus Esclavo, y luego presionamos Next ver Figura 107.

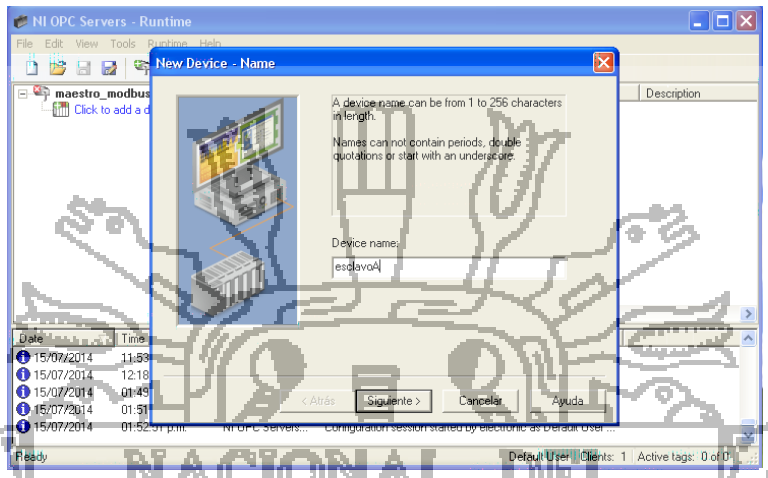



*Figura 107 - Creación de dispositivos para el canal de comunicación.*

A continuación aparecerá la siguiente ventana en la cual debemos seleccionar el protocolo Modbus como modelo de dispositivo ver Figura 108.

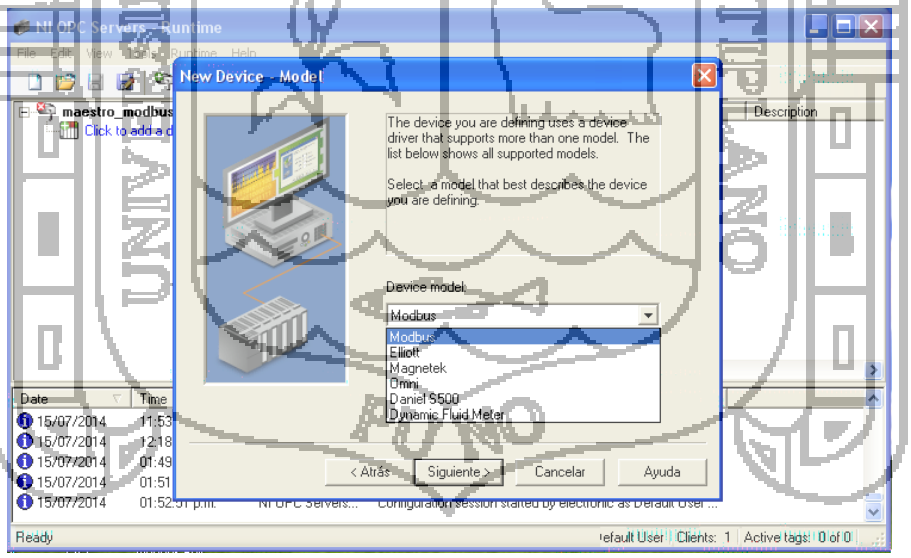



*Figura 108 - Configuración para modbus creado en el OPS Server.*

Configuramos la dirección Modbus del dispositivo, tomando en cuenta que hay 2 esclavos el cual vamos a asignarle la dirección Modbus. Asignamos la dirección 10 y 11 ver Figura 109.

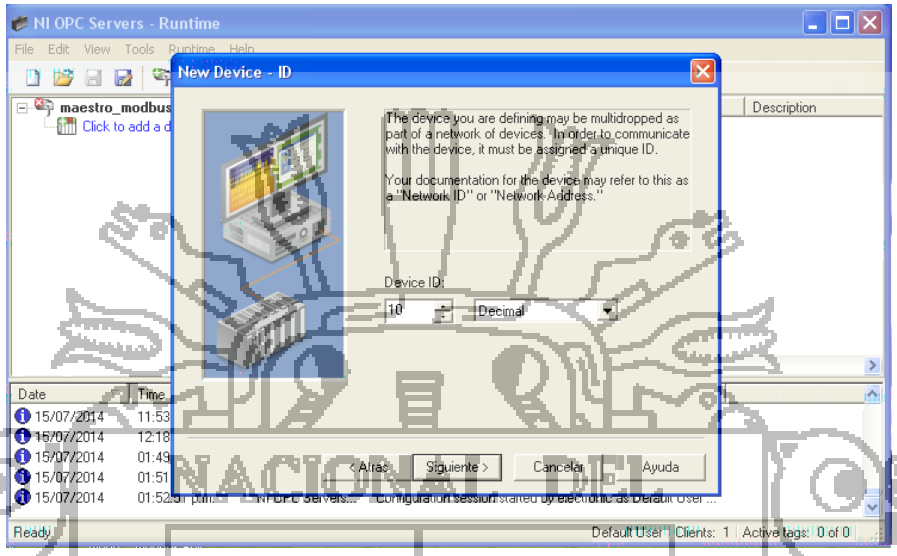



*Figura 109 - Asignación de dirección esclavo Modbus.*

Luego establecemos los tiempos necesarios para establecer la comunicación Modbus tales como Timeout, y tiempo entre solicitud de datos al esclavo ver Figura 110.

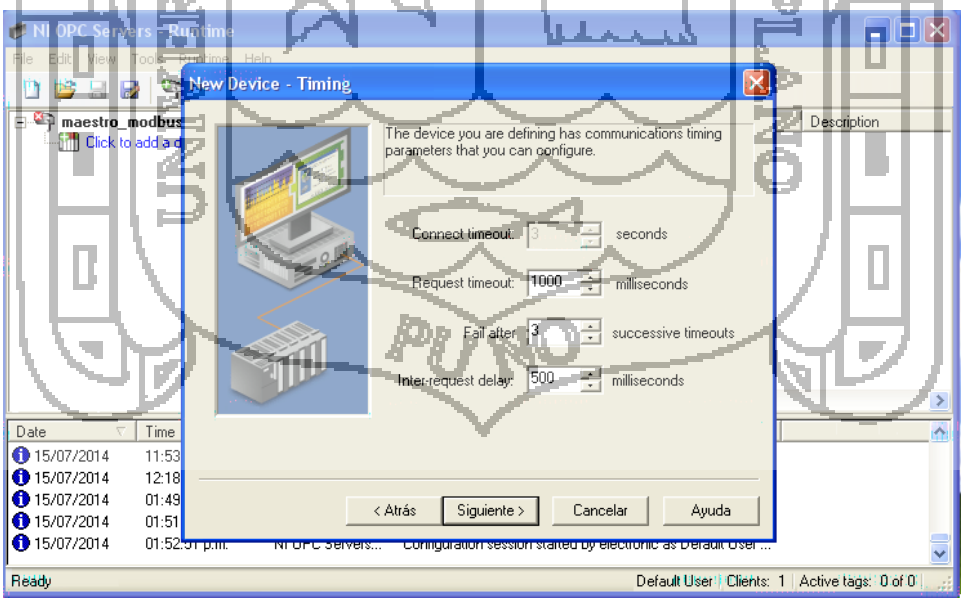



*Figura 110 - Configuraciones de tiempos para la comunicación Modbus.*

Ahora definimos la función 4 de Modbus para leer los datos de los registros del esclavo ver Figura 111.

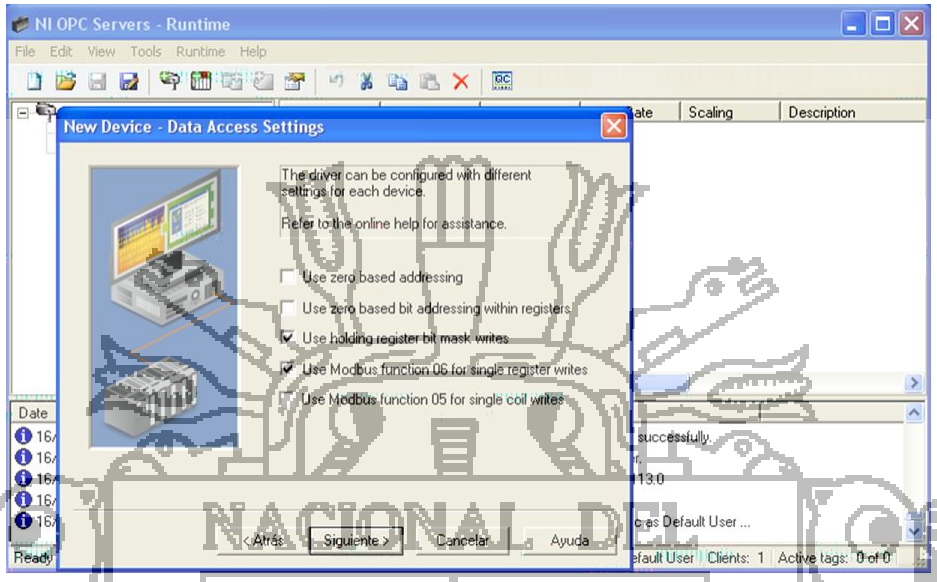



*Figura 111 - Funciones para escribir datos vía Modbus.*

Usamos el orden por defecto de la codificación de los datos recibidos ver Figura 112.

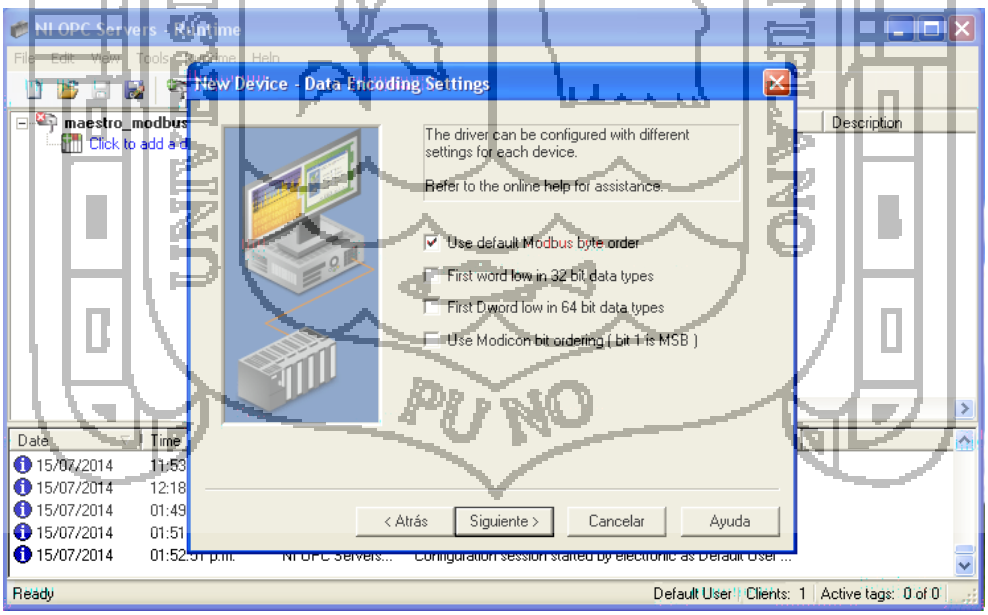



*Figura 112 - Configuración para establecer la codificación de los datos.*

A continuación configuramos el tamaño de los registros que vamos a leer ver Figura 113.

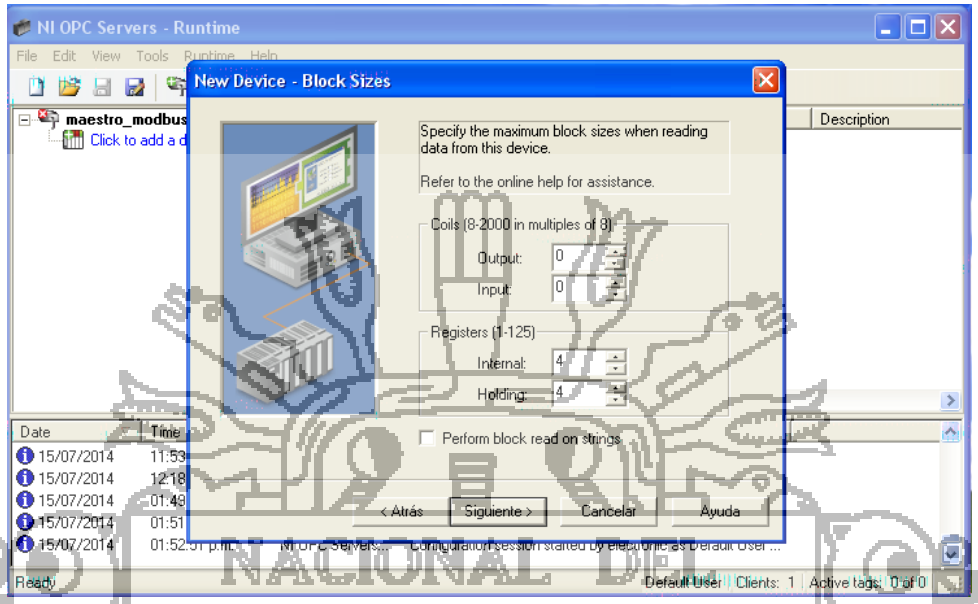



*Figura 113 - Tamaño de los bloques de datos.*

La siguiente pantalla va a mostrar las configuraciones que realizamos y nos permite finalizar la configuración dando clic en Finish ver Figura 114.

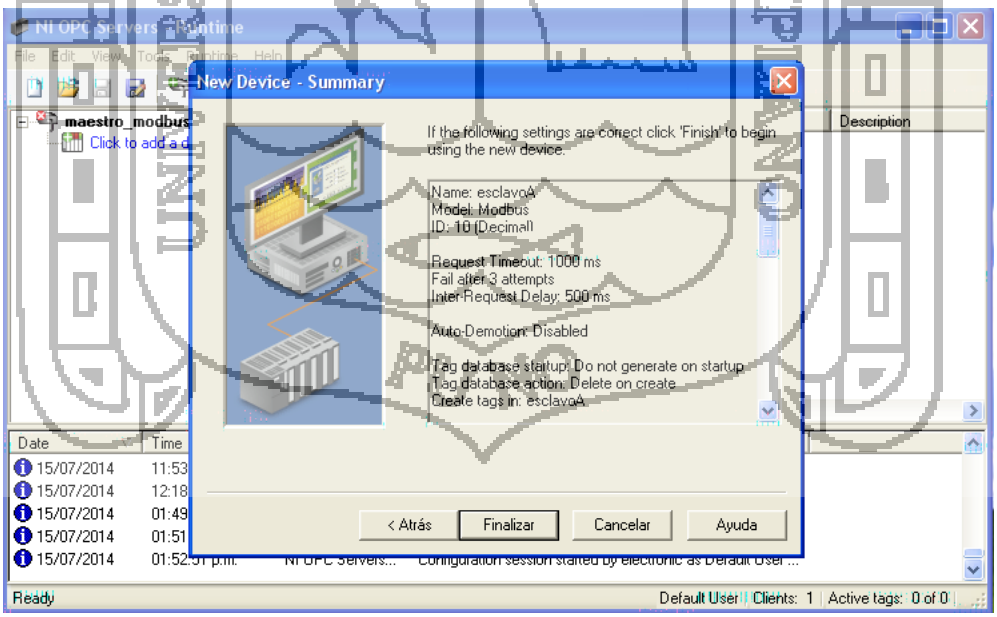



*Figura 114 - Datos configurado en el dispositivo Modbus.*

En este punto se ha generado el esclavoA esclavoB en el explorador, y aparece en la pantalla de tags la opción para ir creando variables a base de las direcciones que estemos utilizando los diferentes registros internos de los esclavos de datos.

Cabe señalar que los registros llamados input registers, el OPC Server los identifica como registros solo de lectura y los ubica a partir de la dirección 30001, mientras que los registros llamados holding registers, los ubica en la dirección 40001 e identifica como registros de lectura y escritura. Un registro solo de lectura, no puede ser escrito no modificado por comunicación ver Figura 115.

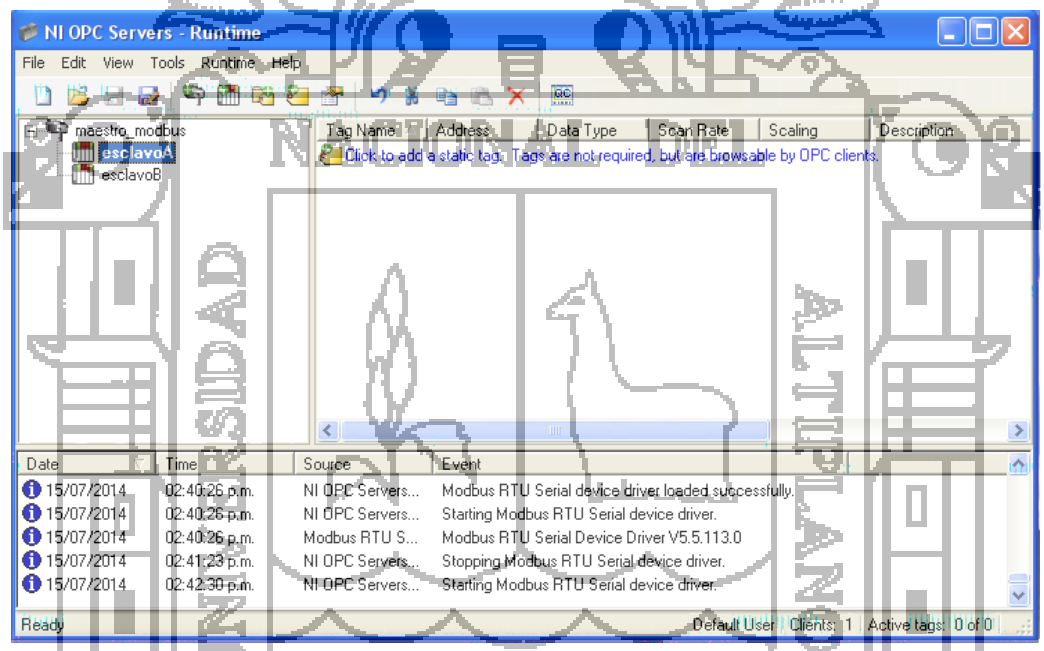



*Figura 115 - El explorador general que muestra el canal y los esclavos.*

Luego de dar clic en la opción add a static tag, se abre la siguiente ventana en la cual le damos un nombre a la variable, le asignamos la dirección correspondiente y le indicamos de que tipo es y si se trata de una variable de lectura y escritura o solo lectura, dando el Nombre de ana1_in, como se dijo antes este registro es el 00h ó 30001 para el OPC server, de los registros de entrada por lo que es una variable solo de lectura ver Figura 115

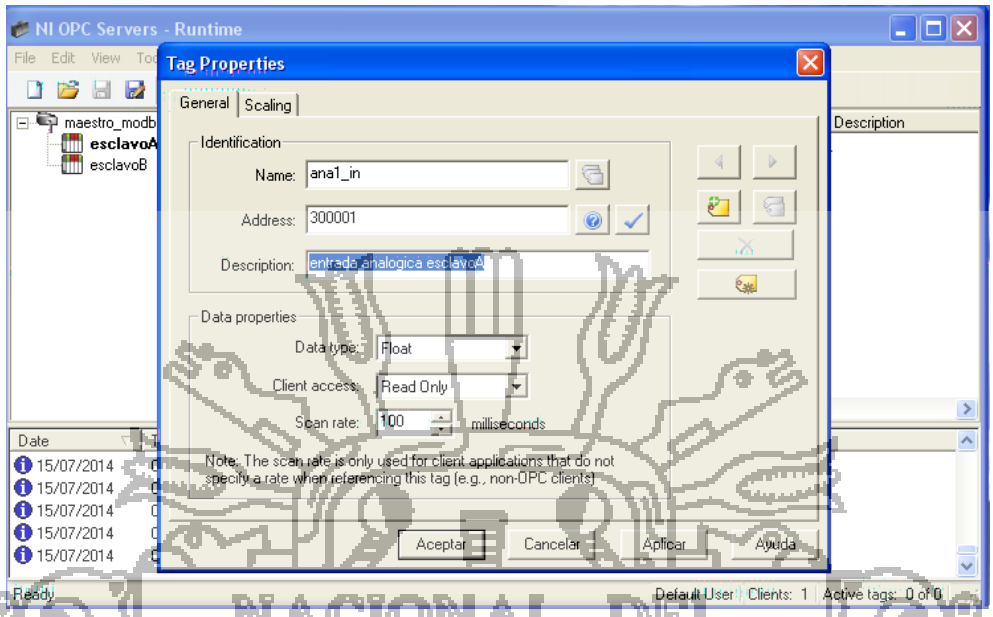



*Figura 116 - Creación de tags asignándoles direcciones.*

Una vez que hayamos creado los Tags correspondientes a las variables a ser leídas, vamos a tener los siguiente, para lo cual, damos clic en el Botón. OPC Client para poder empezar con la lectura de datos desde el Módulo externo ver Figura 117.

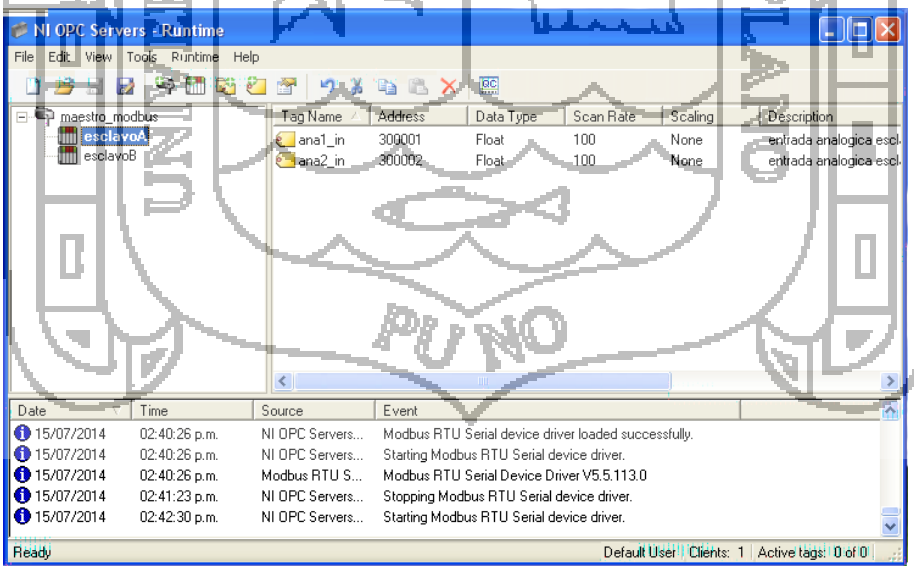



*Figura 117 - Explorador principal, con tags creados.*

Ahora bien, una vez realizadas todas las configuraciones, conectamos los esclavos en el Puerto configurado, COM1 y empezamos a recibir los datos correspondientes a cada variable. En esta ventana se realizan las lecturas ver Figura 118.

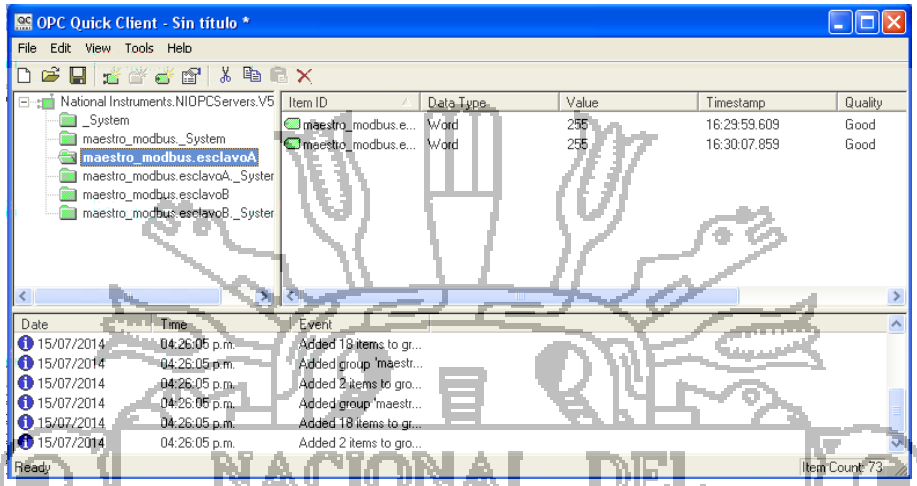



*Figura 118 - OPC cliente obteniendo los valores del dispositivo Modbus esclavo.*

### 3.3.1.2. Enlazando Labview con OPC Server de National Instruments

En Labiview debemos crear un proyecto nuevo para obtener la siguiente ventana y a partir de esta, poder realizar las demás configuraciones para poder establecer un enlace entre NI-OPC Server y el proyecto de Labview. En My Computer hacemos clic derecho y seleccionamos la opción; nuevo I/O Server ver Figura 119.

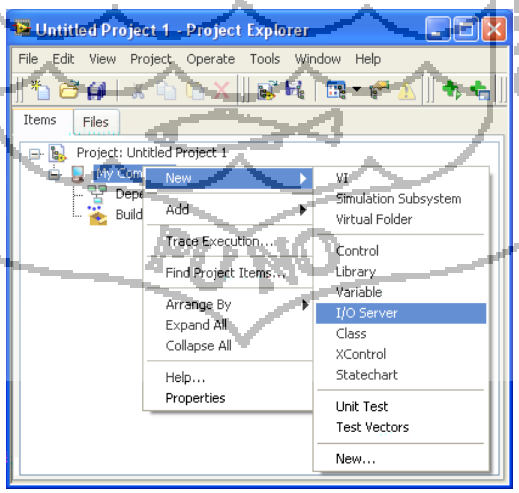



*Figura 119 - Explorador de proyecto Labview.*

Con esta opción se realiza una búsqueda de servidores de datos, entre los cuales debemos seleccionar OPC SERVER de National Instruments para poder adquirir los datos que anteriormente los configuramos con tags. Seleccionamos la opción OPC Cliente damos clikc en continuar. Entre los I/O server se dispone de drivers para Modbus tanto master como slave ver Figura 120.

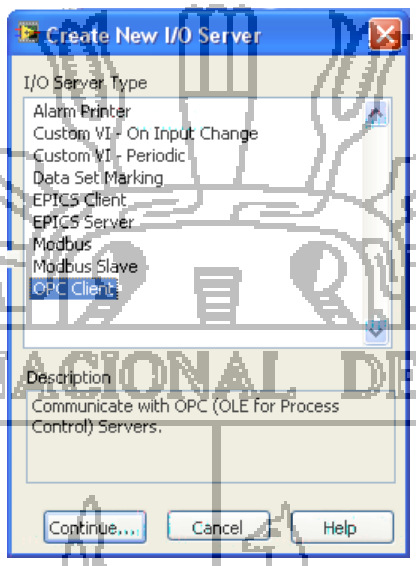



*Figura 120 - Creación de I/O Server.*

Luego se obtiene la siguiente ventana, donde indicamos que tipo de OPC vamos ausar. Aquí elegimos NI OPC SERVER ya que es donde configuramos la conexión con el Interfaz de esclavos de datos. Esta ventana nos permite también configurar datos de tiempos de actualización de datos, al dar click sobre el botón Browse, nos permite buscar servidores de datos ubicados en otras computadoras las cuales forman parte de la misma red dentro de un área de trabajo, es decir se puede usar la aplicación Labview como un HMI cliente y tener el servidor en otra máquina la cual contenga el OPC Server ver Figura 121.

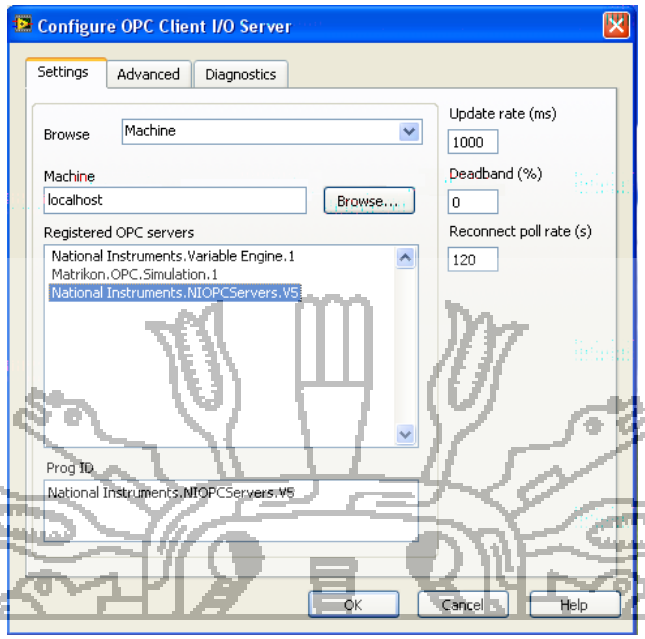



*Figura 121 - Opciones de tipo de OPC Server.*

Una vez que hemos seleccionado el OPC Server, se genera en el proyecto de Labview dentro de una librería, de la siguiente manera ver Figura 122.

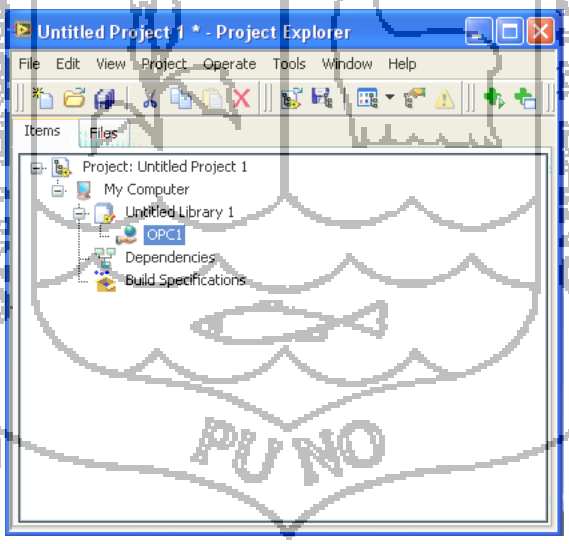



*Figura 122 - Explorador de Proyectos de Labview con e OPC creado.*

Ahora le damos click derecho sobre OPC1 generado y escogemos la opción View I/O Items ver Figura 123.

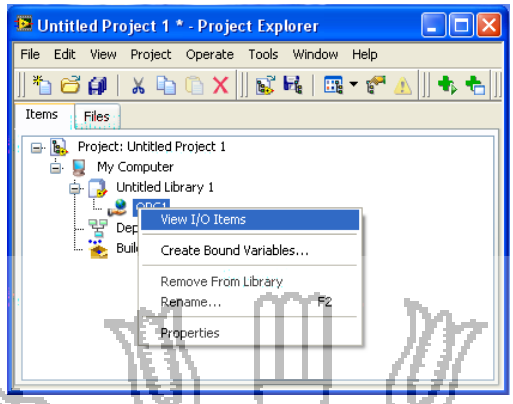



*Figura 123 - OPC dentro del proyecto de Labview.*

Ahora se abre el siguiente cuadro el cual ya posee las variables que creamos en el OPC Server. Y a parecen los tags que anteriormente habíamos creado, como controles de Labview, por lo que podemos usarlos dentro de la pantalla de programación gráfica y asignarle cualquier tipo de aplicación ver Figura 124.

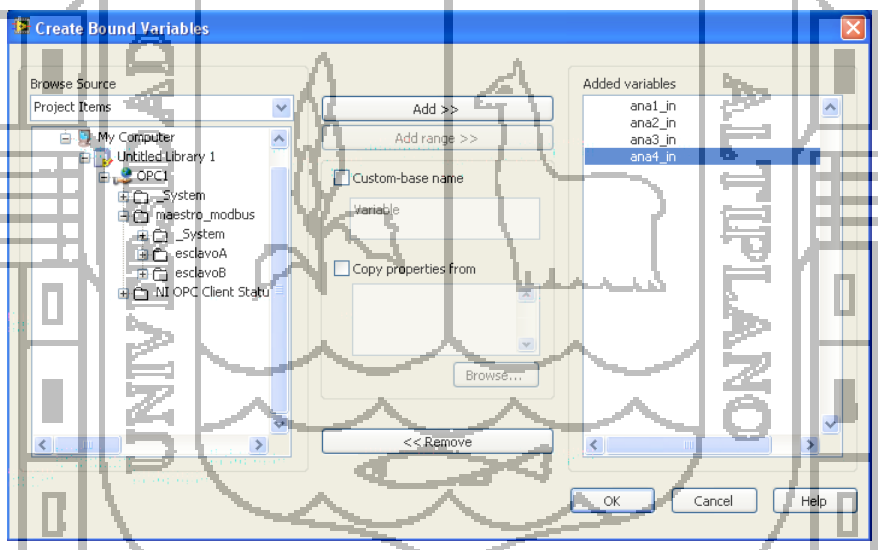



*Figura 124 - Variables del OPC server agregados en el proyecto de Labview.*

Y ahora es necesario ir añadiendo las variables que vamos a utilizar en el proyecto de Labview usando el botón ADD, para posteriormente utilizarlas en la pantalla de programación gráfica de Labview y de esta manera tener el control y monitoreo sobre ellas. Presionamos OK y se obtiene lo siguiente: las variables se agregan al proyecto y se muestran las siguientes ventanas ver Figura 125.

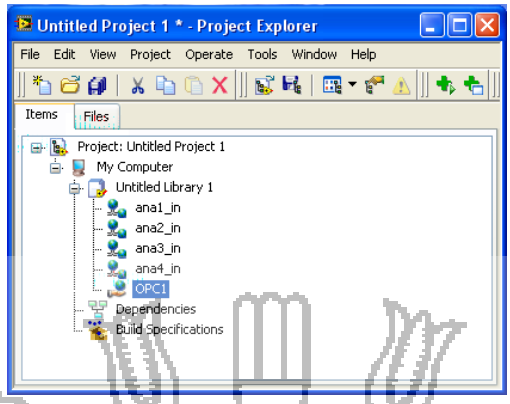



*Figura 125 - Variables creadas se agregan en el explorador del proyecto.*

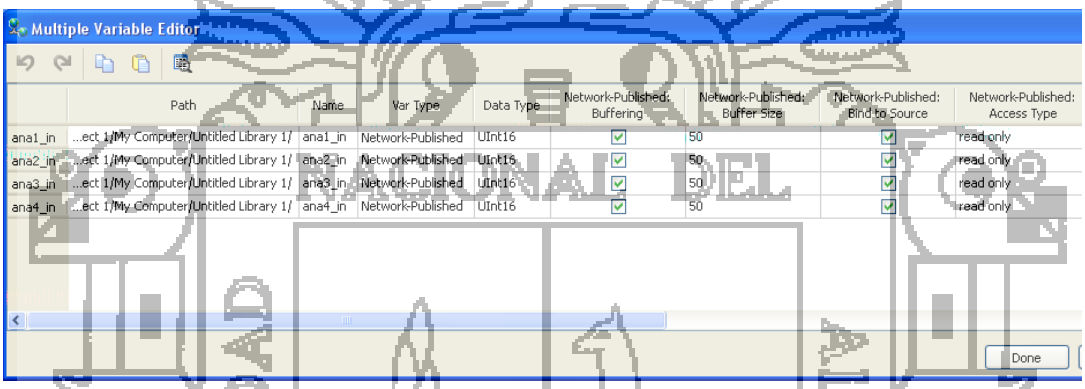



*Figura 126 - Cuadro que indica en resumen las características de cada variable.*

Ahora hay que presionar el botón Done para indicar que hemos concluido con la configuración del enlace entre el proyecto de Labview y el OPC server que previamente creamos ver Figura 126.

### 3.3.1.3. Aspecto Final

Al ejecutar el programa se encuentra la pantalla principal la cual está dividida en 2 paneles, en el panel superior se encuentran los indicadores de las variables climáticas que son monitoreadas por la estación meteorológica y que son actualizados al finalizar cada ciclo de muestreo de la misma. En el panel inferior se observa una caja de texto la cual contiene la información referente al estatus de comunicación entre el computador y la estación. En caso de que la comunicación no haya sido exitosa se muestra un mensaje de error para que se proceda a la conexión física entre la interfaz en la figura se muestra como

luce la pantalla principal del maestro modbus que reside dentro del programa ver Figura 127.

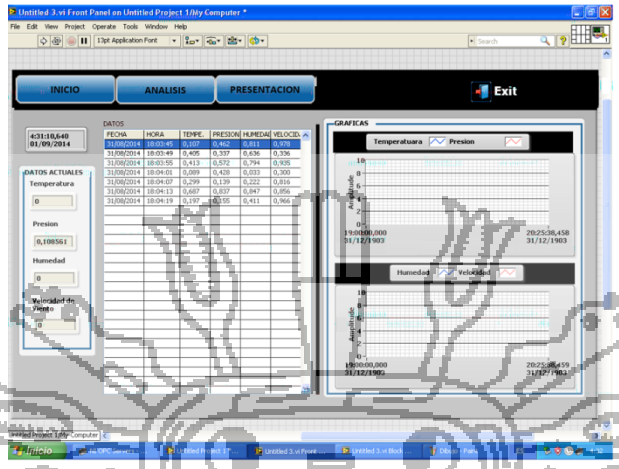



*Figura 127 - Pantalla Principal monitoreo de datos.*

Con los datos obtenidos de la estación meteorológica el maestro modbus los organiza de la siguiente forma para su. Almacenamiento y posterior representación, según cada variable climática

- Los datos del día actual hora fecha.
- Los datos en mes en curso.

Como se muestra en la representación de datos históricos ver Figura 128.

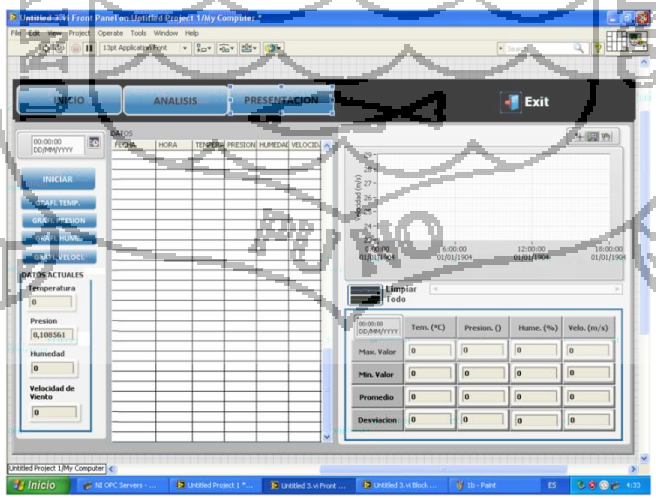



*Figura 128 - Reporte de datos.*

Al iniciar el monitoreo de datos, los valores de las variables de temperatura, humedad, presión y velocidad de viento, se almacenan en un registro, para después presentar un reporte en formato Excel, este reporte se genera con la fecha de uso de la estación de monitoreo, donde se muestra en la primer columna, el tiempo de monitoreo de datos con un periodo de muestreo de 1 segundos y en las .Con esta aplicación se genera el reporte de las variables de la columna en formato

Excel, esto permite contar con un reporte accesible para un posterior análisis fuera de línea

# CAPÍTULO IV

## EXPOSICIÓN Y ANÁLISIS DE LOS RESULTADOS

### 4.1. Diseño e implementación de las tarjetas

Al tratarse de cada una de las tarjetas realizadas son conexionadas con los esclavos con la tarjeta de acondicionamiento de señal. Los esclavos son conectados a al bus modbus como se muestra en la Figura 129.

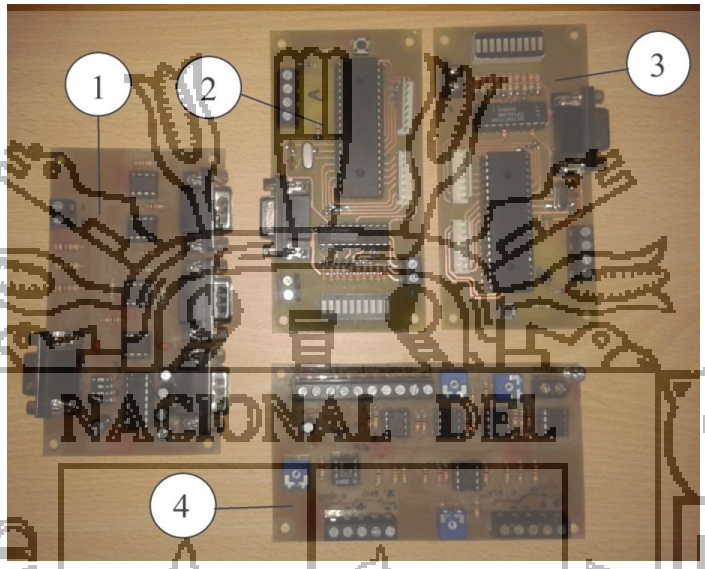



*Figura 129 - Tarjetas.*

| Num. | Descripción | Características |
|---|---|---|
| 1 | Bus Modbus | Interfaz conversión RS232 a RS485 |
| 2 | Esclavo 01 | Esclavo modbus con dirección 10 |
| 3 | Esclavo 02 | Esclavo modbus con dirección 11 |
| 4 | Tarjeta de datos | Acondicionamiento de Variables físicas |

*Tabla 14 - Características de las tarjetas.*

#### 4.1.1. Especificación del interfaz eléctrico al bus serial

Para su montaje se emplea un par trenzado que permite un enlace unidireccional, semiduplex, compartido tanto para trasmitir como para recibir datos, pero no ambas operaciones a la vez. El RS-485 solo define el medio[12] físico, no establece el formato de tramas ni cuáles serán las señales de control ni el protocolo de enlace como se muestra Figura 130.

[12] Las especificación eléctricas mencionadas fueron tomadas de MODBUS over Serial Line Specification y Imlemetation guide V1.0 Pag. 24.

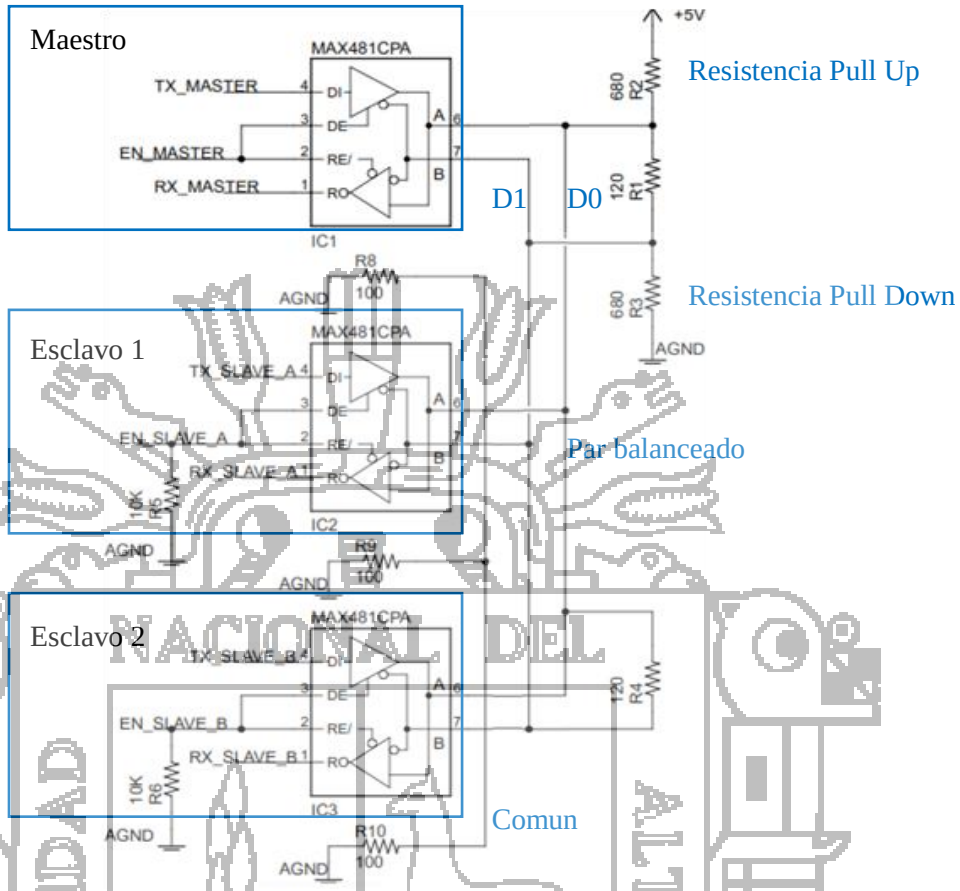



*Figura 130 - Interfaz eléctrica.*

| Interfaz eléctrica | | Para el Dispositivo | Estándar EIA/TIA-488 | Descripción |
|---|---|---|---|---|
| D1 | D1 | I/O | B/B` | Al transmitir un 1, la señal A tiene una tensión +V y la señal B una –V |
| D0 | D0 | I/O | A/A` | Al transmitir un 0, la señal A tiene una tensión –V y la señal B Una +V |
| Común | Común | .. | C/C` | Señal común |

*Tabla 15- Características interfaz electrica.*

### 4.1.2. Especificación del interfaz habilitación del maestro y esclavos

Para la conexión entre la computador el Dispositivo Master se cuenta con un MAX485. En la Figura 131 se muestra los pines utilizados por dicho conector

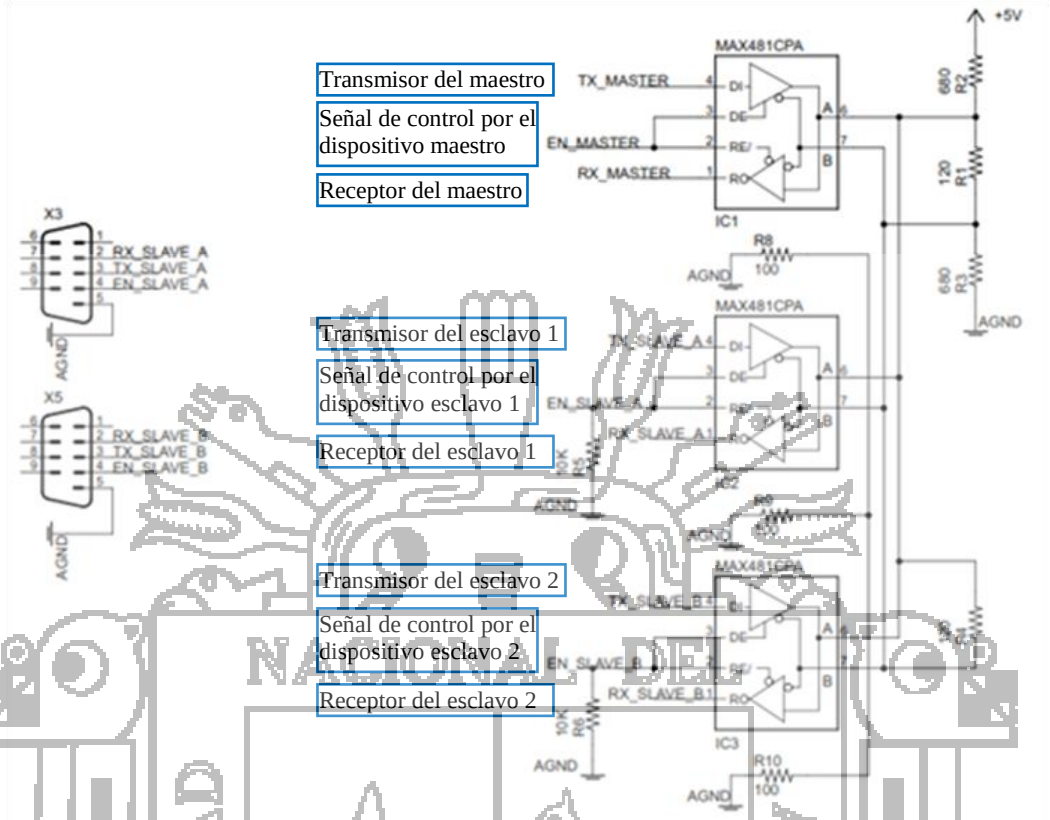



*Figura 131 - Características de conexión.*

Como muestra la Figura 131 el maestro se encarga de, transmisión recepción serial, procesarlos y enviar a todos los Dispositivos Esclavos sobre RS 485 ver Tabla 16.

| Pin Max485 | Señal de control | Estado |
|---|---|---|
| 2 y 3 | 0 | Habilita Recepción del Max485 |
| 2 y 3 | 1 | Habilita Transmisión del Max485 |

*Tabla 16 - Estado de control maestro.*

Mientras se transmite la trama, desde el maestro Figura 131 hacia los Dispositivos Esclavos, la señal de control se encuentra en estado de alta impedancia.

| Esclavo | Pin Max485 | Señal de control | Estado |
|---|---|---|---|
| Esclavo 1 | 2 y 3 | 0 | Habilita Recepción del Max485 |
| | 2 y 3 | 1 | Habilita Transmisión del Max485 |
| Esclavo 2 | 2 y 3 | 0 | Habilita Recepción del Max485 |
| | 2 y 3 | 1 | Habilita Transmisión del Max485 |

*Tabla 17 - Estado de control de los esclavos.*

## 4.2. Herramientas para desarrollar el protocolo Modbus

Para poder desarrollar la comunicación Modbus con los esclavos Microcontrolador PIC se usaron herramientas que actúan como maestro, una vez concluida su construcción. De igual manera, se analizarán los resultados obtenidos en dichas pruebas.

El propósito de esta herramienta es la correcta elaboración y transmisión de las tramas MODBUS de petición, desde maestro computador hasta los Dispositivos Esclavos, pasando Una vez procesada la información por los dispositivos Esclavos, estos elaboran la trama MODBUS de respuesta para enviarla hacia el maestro computador, obviamente, todos los esclavos deben estar conectados y seleccionados en un mismo modo de comunicación. Para comprobar que no existan fallas de comunicación, los Esclavos deben enviar la trama MODBUS de respuesta hacia el maestro computador

### 4.2.1. Programa *MODSCAN32*

Para las pruebas se utilizó el software ***MODSCAN32*** elaborado por Win-Tech, cuyo propósito fundamental es el de simular un Dispositivo Master.

Este software envía Tramas MODBUS (RTU) y valida las tramas de respuesta si estas cumplen con el protocolo MODBUS. Además permite visualizar dichas tramas.

Las tramas MODBUS de respuesta pueden ser visualizadas en formatos tales como: decimal, hexadecimal, binario y notación de punto flotante, tanto en modo RTU ver Figura 132.

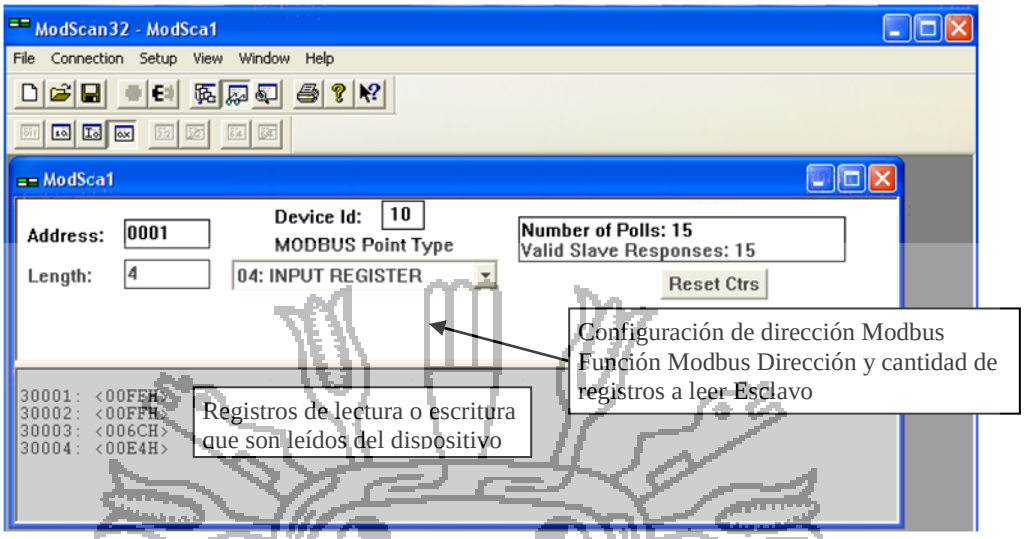



*Figura 132 - Entorno de trabajo ModScan.*

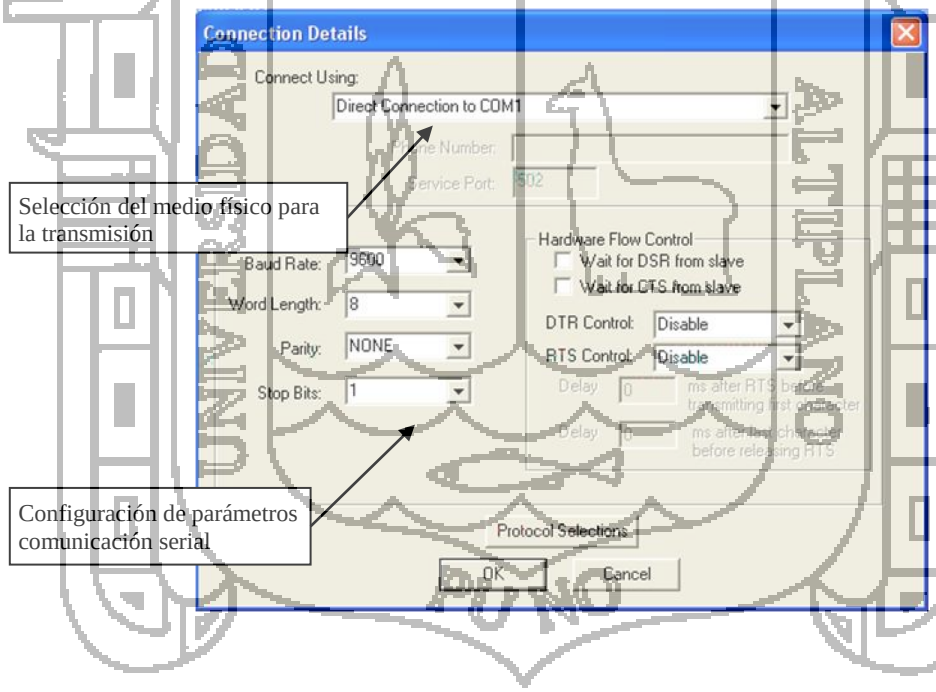



*Figura 133 - Menú para establecer los parámetros de comunicación.*

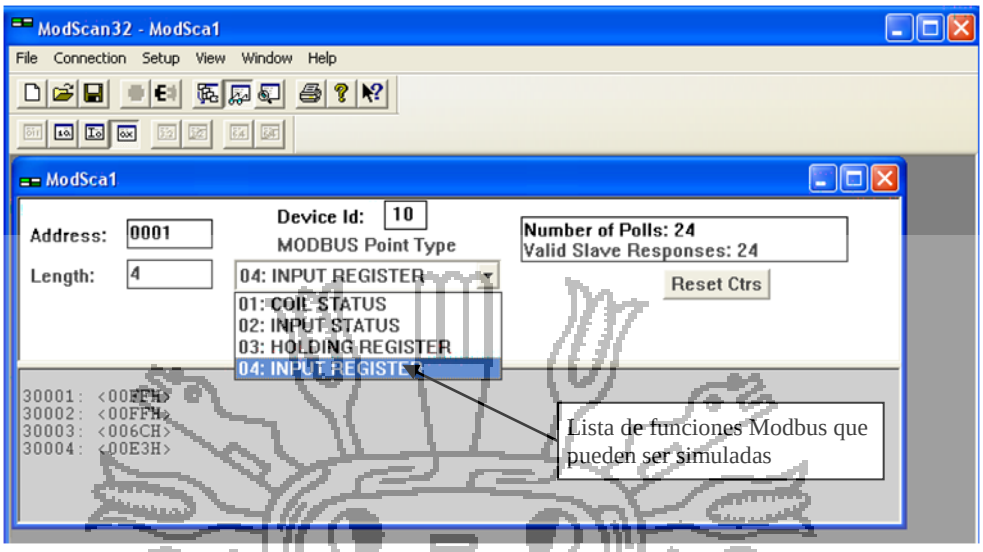



*Figura 134 - Menú para establecer la función Modbus que va a simular el programa.*

### 4.2.2. Pruebas de recepción y envió de mensajes MODBUS

La realización de las pruebas de recepción y envió de mensajes MODBUS. Para la activación de señales y verlas en el programa Modscan32 mencionado anteriormente, a base de dip-switches. Además la captura de señales, En los cuadros rojos se muestra cada parte que componen la red.

El diseño del impreso cuenta de tres tarjetas impresos formando. El esclavo 1 cuenta con conexión para 8 entradas, 8 salidas, 2 leds indicadores de estado de modbus y 4 entradas analógicas también el esclavo cuenta la misma conexión. El circuito 3, cuenta con una terminal DB9 hembra, para la comunicación serial ya sea con la PC o con el convertidor de protocolos, que es necesario para la conexión, bus modbus RTU, conversión RS232 a RS485 como se muestra en la Figura 135.

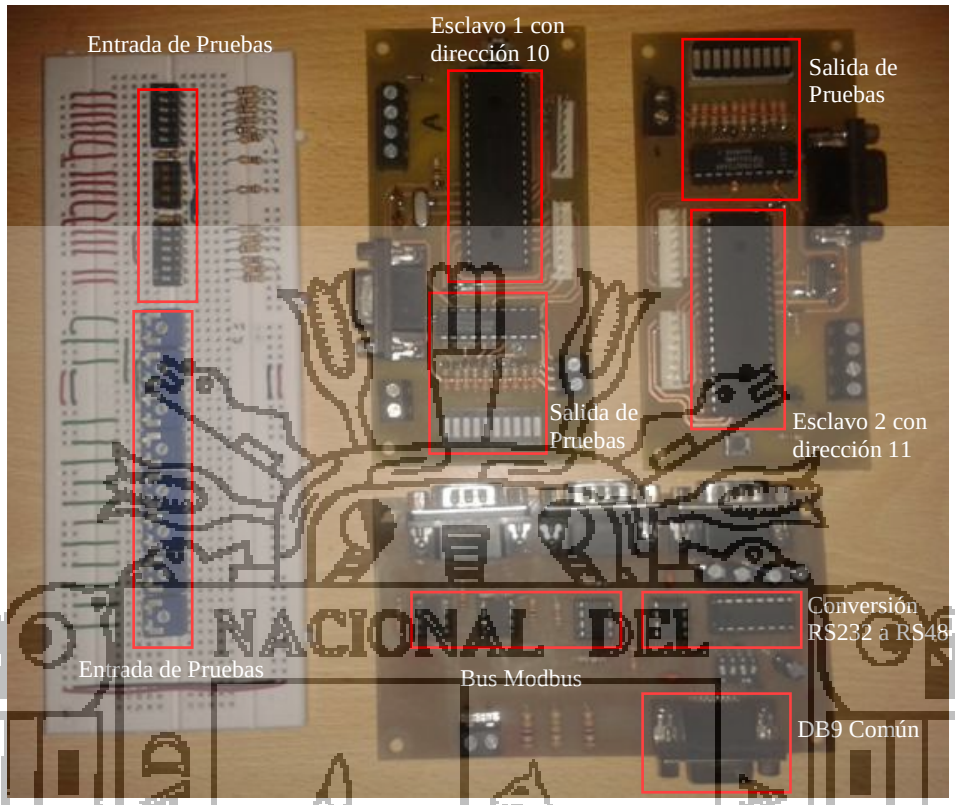



*Figura 135 - Tarjetas MODBUS.*

### 4.2.3. Pruebas de velocidad transmisión de mensajes Modbus

La realización para pruebas de transmisión de mensajes Modbus. Para diferentes velocidades y verlas en el programa Modscan32 mencionado anteriormente se muestran en el siguiente cuadro ver Tabla 18.

| Nº | Velocidad de Transmisión ( bps) | Tiempo de respuesta del esclavo(ms) | Cumple | |
|---|---|---|---|---|
| | | | SI | NO |
| 1 | 110 | 100 | | X |
| 2 | 300 | 100 | | X |
| 3 | 600 | 100 | | X |
| 4 | 1200 | 100 | | X |
| 5 | 2400 | 100 | X | |
| 6 | 4800 | 100 | X | |
| 7 | 9600 | 100 | X | |
| 8 | 14400 | 100 | X | |
| 9 | 19200 | 100 | X | |
| 10 | 38400 | 100 | | X |
| 11 | 56000 | 100 | | X |
| 12 | 57600 | 100 | | X |
| 13 | 115200 | 100 | | X |
| 14 | 128000 | 100 | | X |
| 15 | 256000 | 100 | | X |

*Tabla 18 - Velocidades de Transmisión.*

Obteniéndose los siguientes mensajes por el programa Modscan32

- Para las velocidades de 110 bps, 300 bps, 600 bps, 1200 bps se comunica pero la respuesta del mensaje se produce tiempo de espera es muy lenta ver Figura 136.

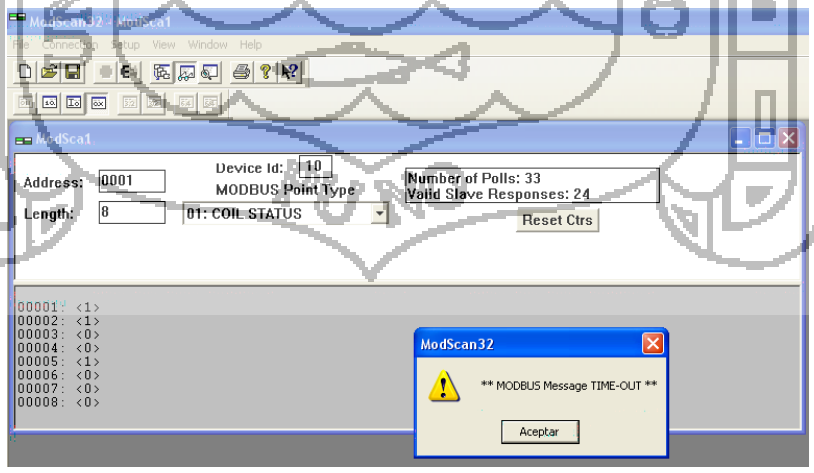



*Figura 136 - Mensaje de TIME-OUT.*

- Para la velocidad de transmisión 2400 bps se comunica el esclavo pero hay mensaje de error ver Figura 137

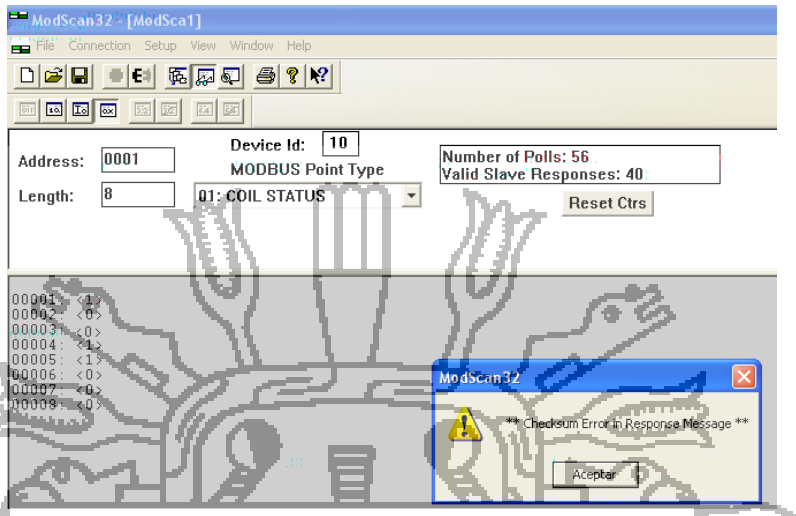



*Figura 137 - Error de mensaje MODBUS.*

- Para las velocidades de 38400 bps, 56000 bps, 115200, 128000 bps, bps, 256000 bps el es esclavo no se puede comunicar mostrando el tiempo de espera ver Figura 138.

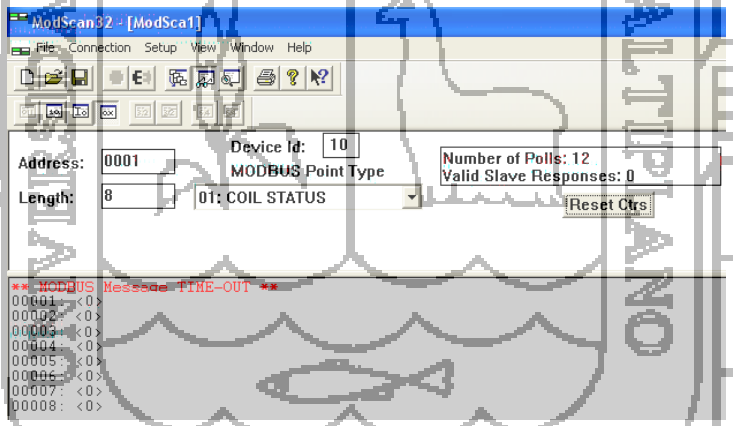



*Figura 138 - Mensaje TIME-OUT Modbus.*

### 4.2.4. Resultado de la prueba de comunicación utilizando la función 04

Se realiza la prueba de comunicación para la Función 04, enviando una trama modbus de petición con la dirección y el número de la entradas a leer, para que el dispositivo esclavo, al cual esté dirigida la trama, construya una trama MODBUS de respuesta con el estado actual de la entrada en la figura se muestran los resultados de la prueba mencionada la trama, modbus de petición en color blanco y la trama MODBUS de respuesta en color negro ver Figura 139.

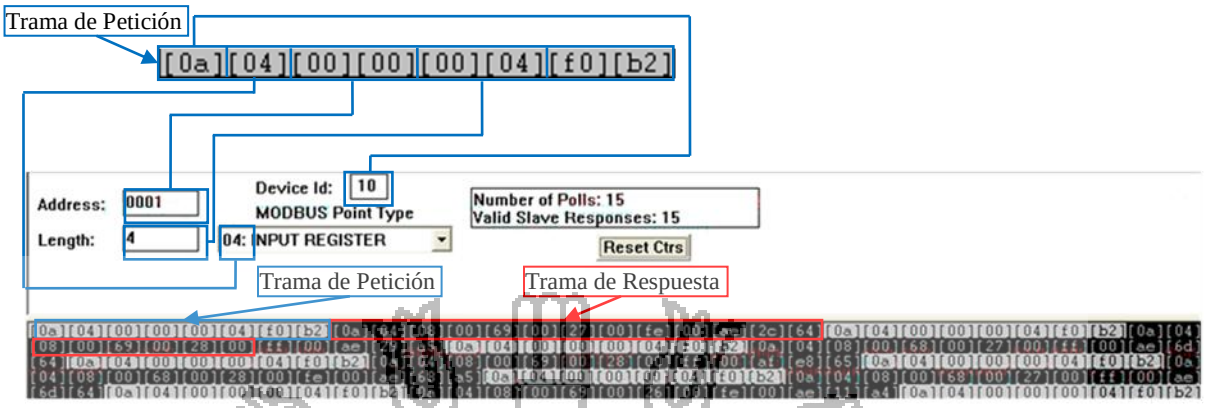



*Figura 139 - Visualización de tráfico de datos válidos de la función 04.*

Obteniendo 4 valores de los registros del esclavo ver Tabla 19.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 30001 | 0069H | Entrada Variable |
| 2 | 30002 | 0028H | Entrada Variable |
| 3 | 30003 | 00FFH | Entrada Variable |
| 4 | 30004 | 00AEH | Entrada Variable |

*Tabla 19 - Estado de registros del esclavo.*

Leemos 4 registros a partir del registro con dirección 1, con dirección esclavo 10 ver Figura 140.

| Dirección del esclavo | Código de función lectura | Dirección alta de lectura | Dirección baja de lectura | Registro alto del número de lectura | Registro bajo del número de lectura | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 04 | 00 | 00 | 00 | 04 | b0 | b2 |

*Figura 140 - Trama de Petición función 04.*

| Dirección del esclavo | Código de función lectura | Número registros de bytes leídos | Valor alto del primer registro | Valor bajo del primer registro | ... | Valor alto del último registro | Valor bajo del último registro | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|---|---|
| 0a | 04 | 08 | 00 | 69 | . | 00 | ae | 2c | 64 |

*Figura 141 - Trama de Respuesta función 04.*

### 4.2.4.1. Respuestas de excepción error dirección función 04

Se efectuó la prueba enviando una trama modbus con una dirección de lectura no válida; con lo cual el Dispositivo Esclavo construirá un mensaje de excepción con un código de dirección ilegal y la enviará hacia el Dispositivo Master ver Figura 142 y Figura 143.


*Figura 142 - Respuestas de excepción (error en dirección).*

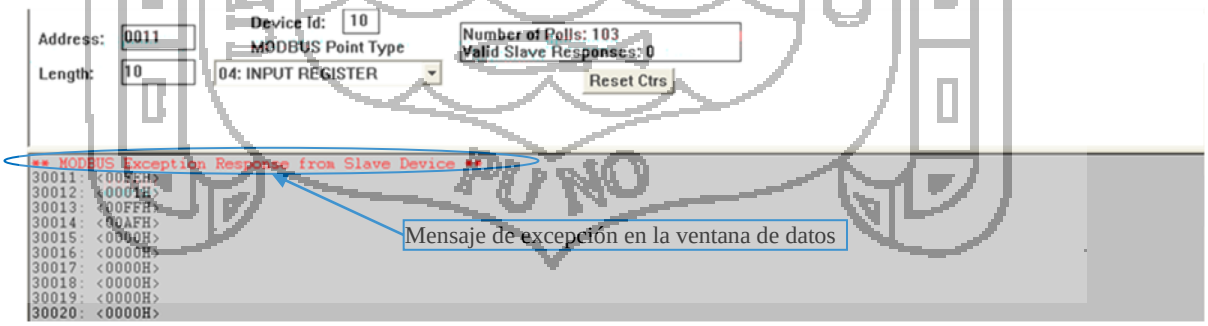



*Figura 143 - Mensaje de excepción (error en dirección).*

Obteniendo 10 registros del esclavo ver Tabla 20.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 30011 | 00FFH | Entrada Variable no valida con error dirección |
| 2 | 30012 | 0001H | Entrada Variable no valida con error dirección |
| 3 | 3013 | 00FDH | Entrada Variable no valida con error dirección |
| 4 | 30014 | 00AFH | Entrada Variable no valida con error dirección |
| 5 | 30015 | 0000H | Entrada Variable no valida con error dirección |
| 6 | 30016 | 0000H | Entrada Variable no valida con error dirección |
| 7 | 30017 | 0000H | Entrada Variable no valida con error dirección |
| 8 | 30018 | 0000H | Entrada Variable no valida con error dirección |
| 9 | 30019 | 0000H | Entrada Variable no valida con error dirección |
| 10 | 30020 | 0000H | Entrada Variable no valida con error dirección |

*Tabla 20 - Estado de registros con error de dirección.*

Leemos 10 registros a partir del registro con dirección 11, con dirección esclavo 10, mostrando datos anteriores y esclavo nos envía la respuesta excepción como se muestra en las trama.

| Dirección del esclavo | Código de función lectura | Dirección alta de lectura | Dirección baja de lectura (dirección incorrecta) | Registro alto del número de lectura | Registro bajo del número de lectura | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 04 | 00 | 0a | 00 | 0a | 51 | 74 |

*Figura 144 - Trama de Petición con dirección diferente.*

| Dirección del esclavo | Función (función de la trama de petición + 80 Hexadecimal) | Código de excepción | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|
| 0a | 84 | 02 | b3 | 03 |

*Figura 145 - Trama de Respuesta con excepción.*

### 4.2.4.2. Respuestas de excepción error en datos función 04

Se efectuó la prueba enviando una trama modbus con 20 datos de lectura no válida; con lo cual el Dispositivo Esclavo construirá un mensaje de excepción con un código de valor ilegal y la enviará hacia el Dispositivo Master ver Figura 146 y Figura 147.

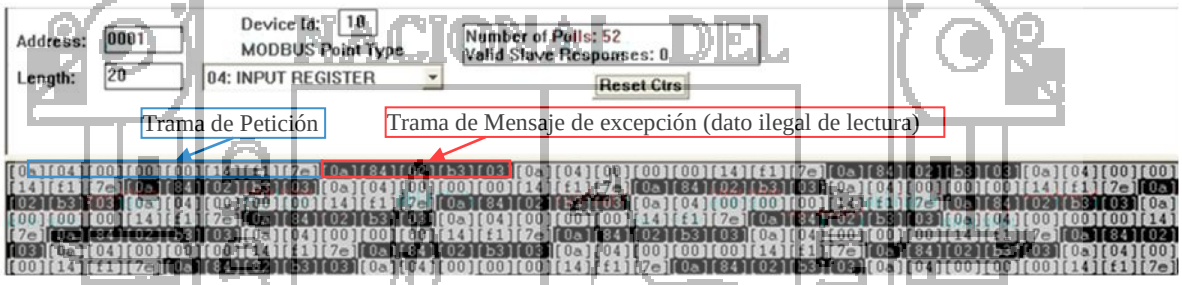



*Figura 146 - Respuestas de excepción (error en datos).*

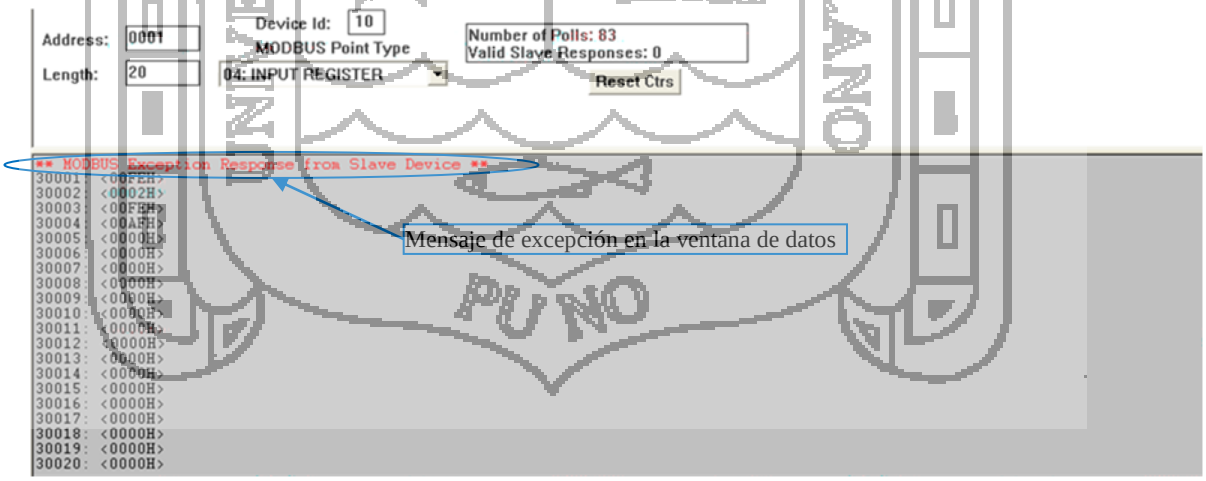



*Figura 147 - Mensaje de excepción (error en datos).*

Obteniendo 20 registros del esclavo ver Tabla 21.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 30001 | 00FFH | Entrada Variable |
| 2 | 30002 | 0001H | Entrada Variable |
| 3 | 30003 | 00FDH | Entrada Variable |
| 4 | 30004 | 00AFH | Entrada Variable |
| 5 | 30005 | 0000H | Entrada Variable |
| 6 | 30006 | 0000H | Entrada Variable |
| 7 | 30007 | 0000H | Entrada Variable |
| 8 | 30008 | 0000H | Entrada Variable |
| 9 | 30009 | 0000H | Entrada Variable |
| 10 | 30010 | 0000H | Entrada Variable |
| 11 | 30011 | 0000H | Entrada Variable no valida con error dato |
| 12 | 30012 | 0000H | Entrada Variable no valida con error dato |
| . . | . . | . . | . . |

*Tabla 21 - Estado de registros error de datos.*

Leemos 20 registros a partir del registro con dirección 1, con dirección esclavo 10, mostrando datos anteriores y esclavo nos envía la respuesta excepción como se muestra en las trama.

| Dirección del esclavo | Código de función lectura | Dirección alta de lectura | Dirección baja de lectura (dirección incorrecta) | Registro alto del número de lectura | Registro bajo del número de lectura (número de registro incorrecto) | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 04 | 00 | 00 | 00 | 14 | f1 | 7e |

*Figura 148 - Trama de Petición con 20 registros.*

| Dirección del esclavo | Función (función de la trama de petición + 80 Hexadecimal) | Código de excepción | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|
| 0a | 84 | 03 | b3 | 03 |

*Figura 149 - Trama de Respuesta con excepción.*

### 4.2.5. Resultado de la prueba de comunicación utilizando la función 02

Se muestran los resultados de esta prueba. La trama modbus de petición aparece en color blanco y la trama modbus de respuesta en color negro ver Figura 150.

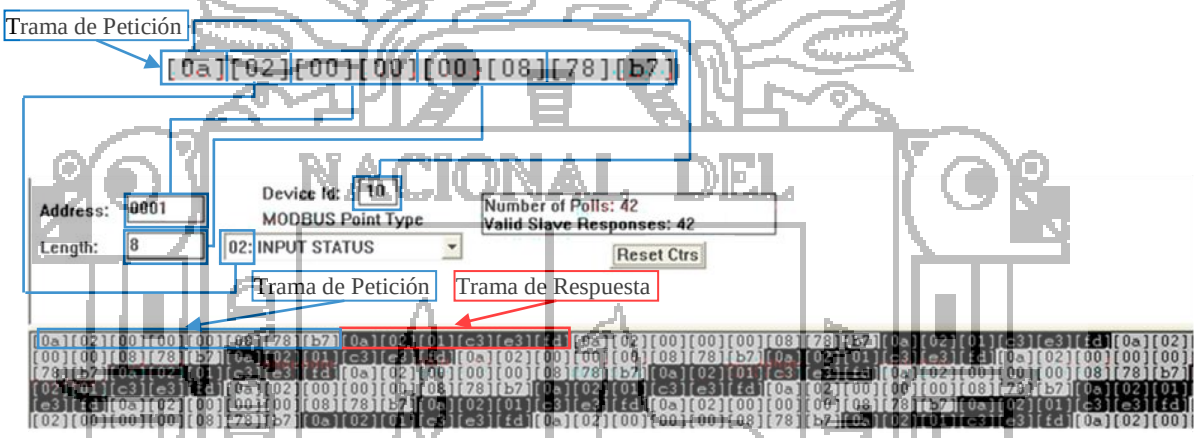



*Figura 150 - Visualización de tráfico de datos válidos de la función 02.*

Obteniendo 8 bit del esclavo ver Tabla 22.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 10001 | 1 | Entrada ON/OFF |
| 2 | 10002 | 1 | Entrada ON/OFF |
| 3 | 10003 | 0 | Entrada ON/OFF |
| 4 | 10004 | 0 | Entrada ON/OFF |
| 5 | 10005 | 0 | Entrada ON/OFF |
| 6 | 10006 | 0 | Entrada ON/OFF |
| 7 | 10007 | 1 | Entrada ON/OFF |
| 8 | 10008 | 1 | Entrada ON/OFF |

*Tabla 22 - Estado de bits del esclavo.*

Leemos 8 bit a partir de la dirección 1, con dirección esclavo 10.

| Dirección del esclavo | Código de función lectura digital | Dirección alta de lectura digital | Dirección baja de lectura digital | Registro alto del número de lectura digital | Registro bajo del número de lectura digital | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 02 | 00 | 00 | 00 | 08 | 78 | b7 |

*Figura 151 - Trama de Petición función 02.*

| Dirección del esclavo | Código de función lectura | Número de bytes de estado de las entradas digitales | Estado de la lectura Digital | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|
| 0a | 02 | 01 | C3 | e3 | fd |

*Figura 152 - Trama de Respuesta función 02.*

### 4.2.5.1. Respuestas de excepción error dirección función 02

Se efectuó la prueba enviando una trama modbus con una dirección de lectura no válida; con lo cual el Dispositivo Esclavo construirá un mensaje de excepción con un código de dirección ilegal y la enviará hacia el Dispositivo Master ver Figura 153 y Figura 154.

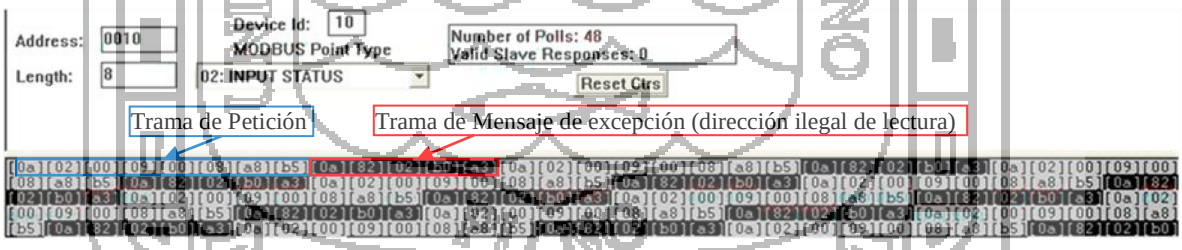



*Figura 153 - Respuestas de excepción (error en dirección) función 02.*

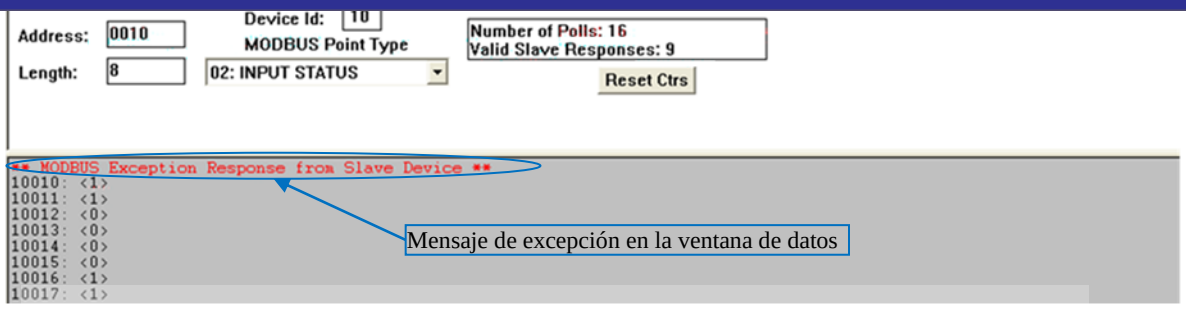



*Figura 154 - Mensaje de excepción (error en dirección) función 02.*

Obteniendo 8 bit del esclavo ver Tabla 23.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 10010 | 1 | Entrada ON/OFF con error a dirección |
| 2 | 10011 | 1 | Entrada ON/OFF con error a dirección |
| 3 | 10012 | 0 | Entrada ON/OFF con error a dirección |
| 4 | 10013 | 0 | Entrada ON/OFF con error a dirección |
| 5 | 10014 | 0 | Entrada ON/OFF con error a dirección |
| 6 | 10015 | 0 | Entrada ON/OFF con error a dirección |
| 7 | 10016 | 1 | Entrada ON/OFF con error a dirección |
| 8 | 10017 | 1 | Entrada ON/OFF con error a dirección |

*Tabla 23- Estado de bits con error de dirección.*

Leemos 8 bit a partir de la dirección 1, con dirección esclavo 10, mostrando datos anteriores y esclavo nos envía la respuesta excepción de dirección como se muestra en las trama.

| Dirección del esclavo | Código de función lectura | Dirección alta de lectura | Dirección baja de lectura (dirección incorrecta) | Registro alto del número de lectura | Registro bajo del número de lectura | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 02 | 00 | 09 | 00 | 08 | a8 | b5 |

*Figura 155 - Trama de Petición con dirección diferente función 02.*

| Dirección del esclavo | Función (función de la trama de petición + 80 Hexadecimal | Código de excepción | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|
| 0a | 82 | 02 | b0 | a3 |

*Figura 156 - Trama de Respuesta con excepción.*

### 4.2.5.2. Respuestas de excepción error en datos función 02

Se efectuó la prueba enviando una trama modbus con 10 datos de lectura no válida; con lo cual el Dispositivo Esclavo construirá un mensaje de excepción con un código de valor ilegal y la enviará hacia el Dispositivo Master ver Figura 157 y Figura 158.

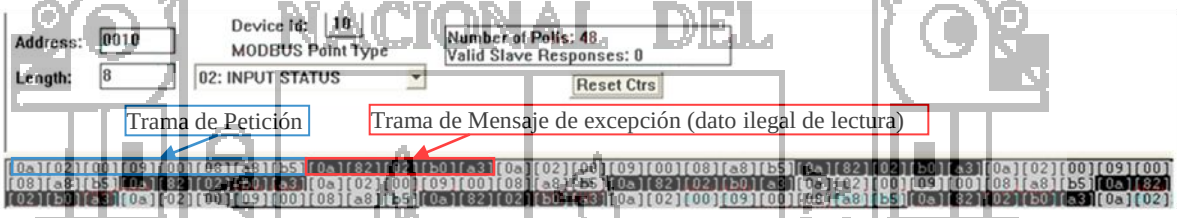



*Figura 157 - Respuestas de excepción (error en datos).*

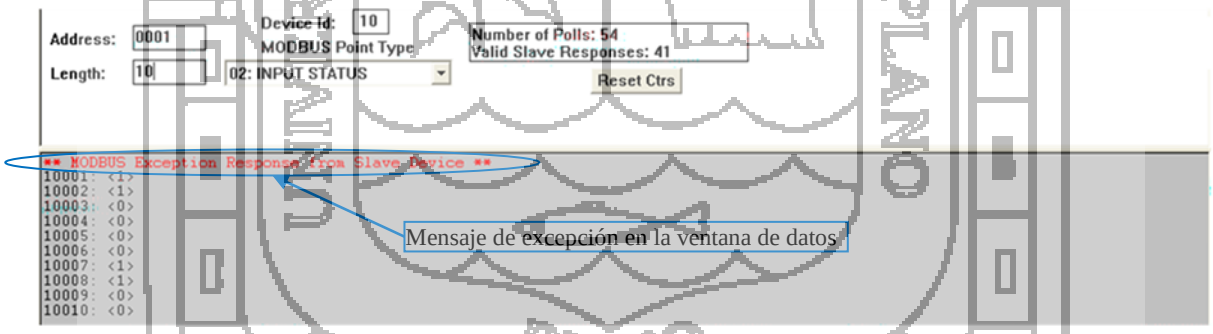



*Figura 158 - Mensaje de excepción (error en datos) función 02.*

Obteniendo 8 bit del esclavo como se muestra la Tabla 24.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 10001 | 1 | Entrada ON/OFF |
| 2 | 10002 | 1 | Entrada ON/OFF |
| 3 | 10003 | 0 | Entrada ON/OFF |
| 4 | 10004 | 0 | Entrada ON/OFF |
| 5 | 10005 | 0 | Entrada ON/OFF |
| 6 | 10006 | 0 | Entrada ON/OFF |
| 7 | 10007 | 1 | Entrada ON/OFF |
| 8 | 10008 | 1 | Entrada ON/OFF |
| 9 | 10009 | 0 | Entrada ON/OFF con error a datos |
| 10 | 10010 | 0 | Entrada ON/OFF con error a datos |

*Tabla 24 - Estado de bits con error de datos.*

Leemos 10 bit a partir del registro con dirección 1, con dirección esclavo 10, mostrando datos anteriores y esclavo nos envía la respuesta excepción de dirección como se muestra en las trama

| Dirección del esclavo | Código de función lectura | Dirección alta de lectura | Dirección baja de lectura (dirección incorrecta) | Bit alto del número de lectura | Bit bajo del número de lectura (número de bit incorrecto) | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 02 | 00 | 09 | 00 | 08 | a8 | b5 |

*Figura 159 - Trama de Petición con 10 bits de datos.*

| Dirección del esclavo | Función (función de la trama de petición + 80 Hexadecimal) | Código de excepción | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|
| 0a | 83 | 03 | B0 | a3 |

*Figura 160 - Trama de Respuesta con excepción de error de datos.*

### 4.2.6. Resultado de la prueba de comunicación utilizando la función 05

Se efectuó la prueba enviando una trama modbus de petición el esclavo responderá activando la salida digital se muestran los resultados de esta prueba. La trama modbus de petición aparece en color blanco y la trama modbus de respuesta en color negro ver Figura 161.

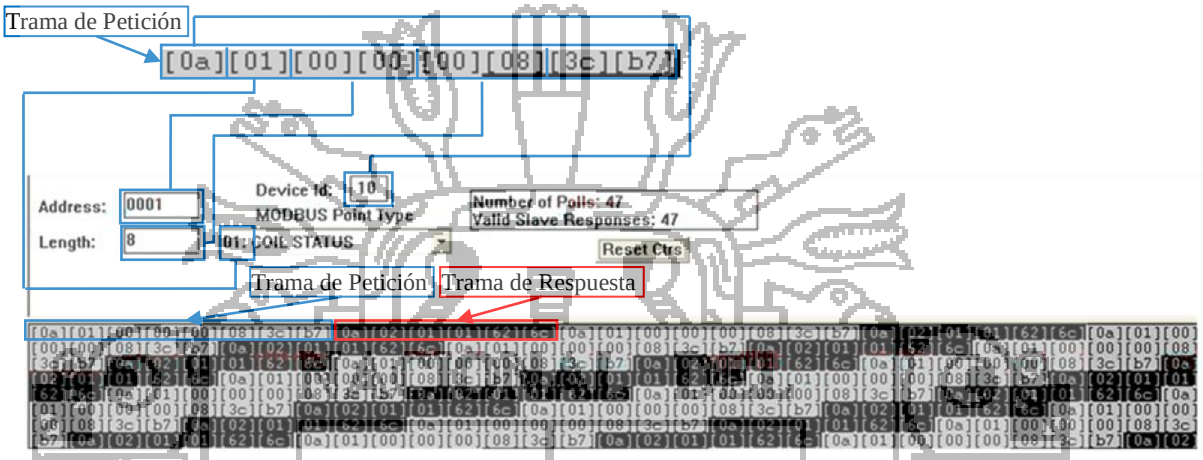



*Figura 161 - Visualización de tráfico de datos válidos de la función 05.*

Escribimos 8 bit del esclavo ver Tabla 25.

| Nº | Dirección Modbus | Valor (Hex) | Descripción |
|---|---|---|---|
| 1 | 00001 | 1 | Salida ON/OFF |
| 2 | 00002 | 1 | Salida ON/OFF |
| 3 | 00003 | 0 | Salida ON/OFF |
| 4 | 00004 | 0 | Salida ON/OFF |
| 5 | 00005 | 0 | Salida ON/OFF |
| 6 | 00006 | 0 | Salida ON/OFF |
| 7 | 00007 | 0 | Salida ON/OFF |
| 8 | 00008 | 0 | Salida ON/OFF |

*Tabla 25 - Estado de bits de salida.*

Escribimos 8 bit a partir del registro de la dirección 1, con dirección esclavo 10.

| Dirección del esclavo | Código de función salida digital | Dirección alta de salida digital | Dirección baja de salida digital | Registro alto del número de salida digital | Registro bajo del número de salida digital | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|---|---|
| 0a | 01 | 00 | 00 | 00 | 08 | 3c | b7 |

*Figura 162 - Trama de Petición función 05.*

| Dirección del esclavo | Código de función salida | Número de bytes de estado de las salida digitales | Estado de la salida Digital | CRC alta de la trama | CRC baja de la trama |
|---|---|---|---|---|---|
| 0a | 01 | 01 | 01 | 62 | 6c |

*Figura 163 - Trama de Respuesta función 05.*

De igual manera se realizaron las excepciones de error de dirección y datos, enviando una trama MODBUS para salidas digitales de petición para esclavo construya una trama MODBUS de excepción.

### 4.2.7. Detección de errores CRC

Comprobación de Redundancia Cíclica[13] es de dos bytes, conteniendo un valor binario de 16 bits. El valor del CRC es calculado por el maestro emisor, el cual añade el CRC al mensaje, el esclavo receptor recalcula un CRC durante la recepción del mensaje y compara el valor calculado con el valor actual recibido en el campo de CRC. Si los dos valores no son iguales, resulta un error. Un procedimiento para generar un CRC es:

1. Cargar un registro de 16 bits que denominaremos registro CRC, con FFFF (todos 1).
2. XOR del primer byte - 8 bits - del mensaje con el byte de orden bajo del registro CRC de 16 bits, colocando el resultado en el registro CRC.
3. Desplazar el registro CRC un bit a la derecha (hacia el LSB- bit menos significativo), rellenando con un cero el MSB – bit más significativo -. Extraer y examinar el LSB.
4. (Si el LSB era 0): Repetir paso 3 (otro desplazamiento). (Si el LSB era 1): Hacer XOR entre el registro CRC y el valor polinómico 0xA001 hex (1010 0000 0000 0001).
5. Repetir los pasos 3 y 4 hasta que se hayan efectuado 8 desplazamientos. Una vez hecho esto, se habrá procesado un byte completo – 8 bits.

[13] Las especificación para CRC mencionadas fueron tomadas de MODBUS over Serial Line Specification y Imlemetation guide V1.0 Pag. 41.

6. Repetir los pasos 2 al 5 para el próximo byte – 8 bits – del mensaje. Continuar haciendo esto hasta que todos los bytes hayan sido procesados.
7. El contenido final del registro CRC es el valor CRC.
8. Cuando el CRC es situado en el mensaje, el bytes de orden bajo se sitúa primero, seguido por el byte de orden alto.

#### 4.2.7.1. Detección de errores CRC función 04

Se comprobó la detección de errores para la función 04 con la siguiente trama de petición del maestro modbus ver Figura 164 y la Tabla 26.

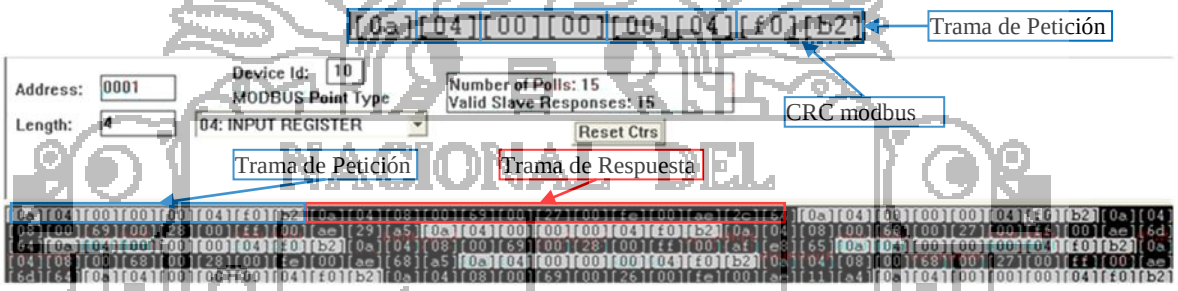



*Figura 164 - Detección de errores para la trama de petición.*

| N. bytes | Trama Modbus | crc modbus |
|---|---|---|
| 6 | 0A0400000004 | F0B2 |

Valor Hex. del polinomio 0xA001 =

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| byte N. | Hex | Cargar CRC | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0A | 0000000000001010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 04 | 0000000000000100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| | | xor 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | xor 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

3F47

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 4 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | xor 8 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | **06D3** |
| 3 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| | | xor 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | | xor 3 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | |
| | | xor 5 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | |
| | | xor 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | |
| | | xor 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | **5302** |
| 4 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 8 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | **423D** |
| 5 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | **BD31** |
| 6 | **04** | 0000000000000100 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | | |
| | | xor 2 tramas de arriba | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 2 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | |
| | | xor 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 7 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | |
| | | xor 8 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | **F0B2** |

*Tabla 26 - Detección de errores para la trama de petición.*

La trama de respuesta detección de errores es construida por el esclavo como se muestra en la ver Figura 165 y los resultados en la Tabla 27.

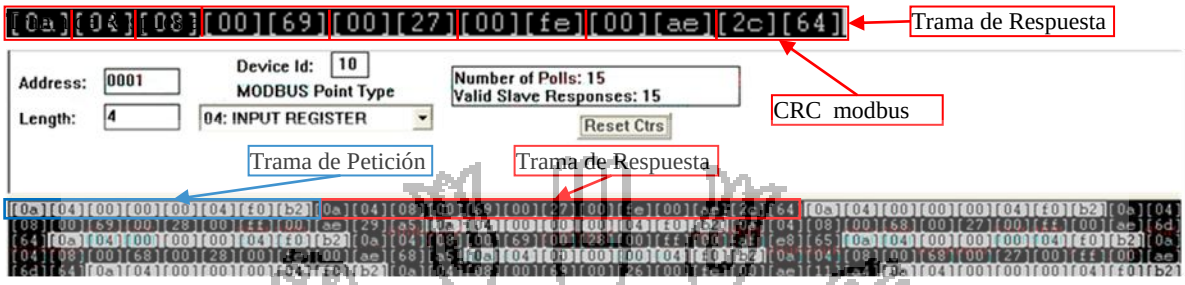



*Figura 165 - Detección de errores para la trama respuesta.*

| N. bytes | Trama Modbus | crc modbus |
|---|---|---|
| 11 | 0A04080069002700FE00AE | 2C64 |

Valor Hex. del polinomio 0xA001 = 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

| byte N. | Hex | cargar CRC | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0A | 0000000000001010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 3F47 |
| 2 | 04 | 0000000000000100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 4 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 06D3 |
| 3 | 08 | 0000000000001000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 3 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | **52C4** |
| 4 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 5 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 6 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 7 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 8 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | **45FD** |
| 5 | **69** | 0000000001101001 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **1** | |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | |
| | | xor 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | |
| | | xor 2 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 3 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 4 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | **FCDD** |
| 6 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 2 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 3 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | **DD41** |
| 7 | **27** | 0000000000100111 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | |
| | | xor 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 3 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | **C143** |
| 8 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **8290** |
| 9 | **FE** | 0000000011111110 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | |
| | | xor 2 tramas de arriba | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 3 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 8 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | **91E1** |
| 10 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | |
| | | xor 5 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | |
| | | xor 6 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 7 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | |
| | | xor 8 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **20AC** |
| 11 | **AE** | 0000000010101110 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **0** | |
| | | xor 2 tramas de arriba | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 2 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 3 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 7 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | |
| | | xor 8 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | **2C64** |

*Tabla 27 - Detección de errores para la trama respuesta.*

De igual manera se realizaron las detección de errores para las funciones 02 y 05 se muestra en el ANEXO III.

### 4.2.8. Control de Comunicación Modbus

A continuación se comprueba con el OPC Client se vea una correcta comunicación. Para ello se guarda el proyecto y se abre el OPC Quick Client, El software

Quick Client consiste en un software mediante el cual es posible acceder a todos los datos disponibles en la aplicación del server como por ejemplo los del sistema, diagnóstico, así como las etiquetas definidas por el usuario ver Figura 166.

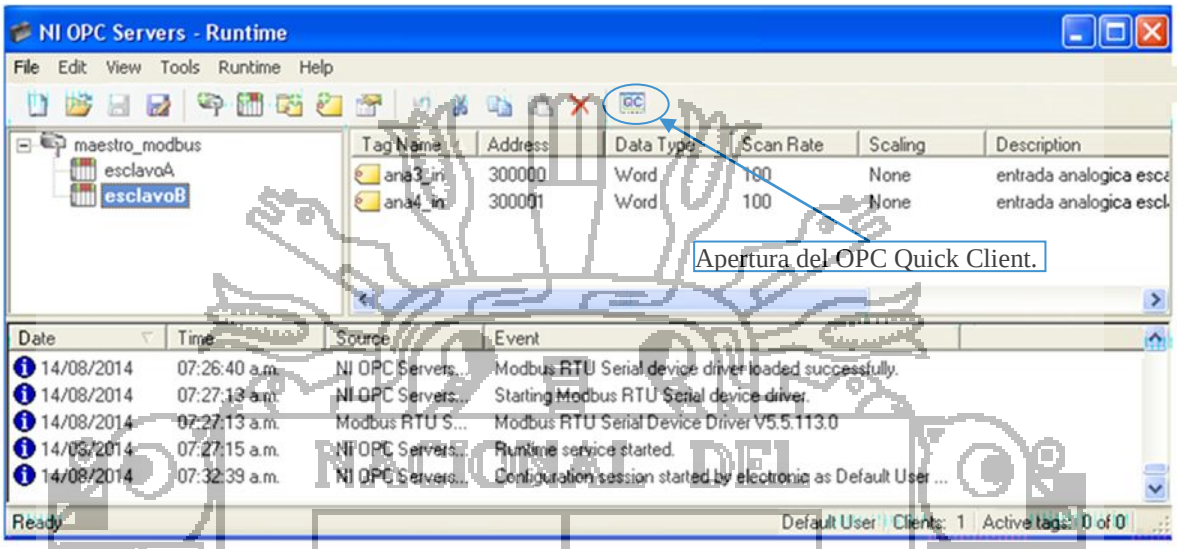



*Figura 166: Guardado del programa y apertura del OPC Quick Client.*

El Quick Client también permite leer y escribir datos con el OPC Quick Client se pueden realizar operaciones en un servidor OPC tanto a nivel de servidor, como de Grupo, como de Ítem. Esto permite a un usuario probar la confiabilidad de un servidor OPC antes de configurar los programas de aplicación que finalmente se conectarán con el servidor OPC. Esto convierte al programa en una herramienta de valor incalculable a la hora de realizar tests de configuraciones de servidor y para cerciorarse de que el servidor de comunicaciones del dispositivo funciona correctamente. De esta manera, cualquier solución de problemas necesario para configurar el software, puede ser reducido en los esclavos y no al OPC Server. Esto puede reducir considerablemente el tiempo empleado para ello.
Como se puede ver no se produce una correcta comunicación ya que no se puede ver el valor del Tag y la calidad de la comunicación es mala ver Figura 167 y la Tabla 28.

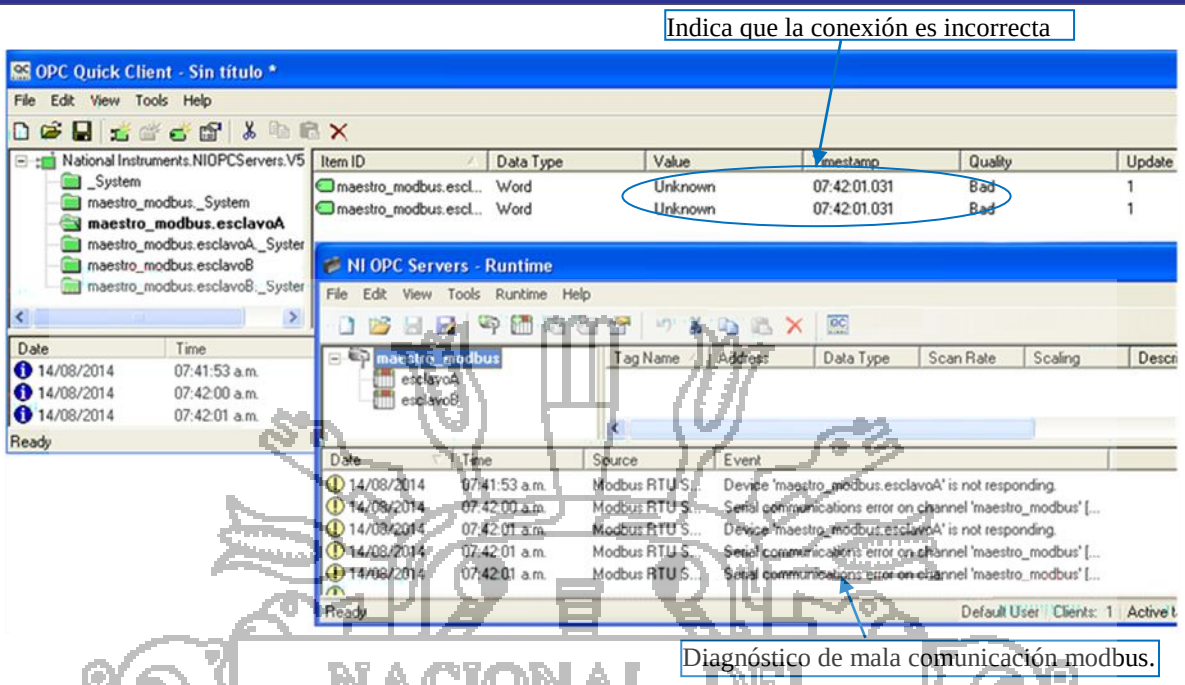



*Figura 167* - OPC Quick Client

| Nº | valor | Estado de comunicación |
|---|---|---|
| 1 | Unkown | Bad |
| 2 | Unkown | Bad |

*Tabla 28 - Estado de comunicación mala.*

Como se puede ver a continuación, el OPC está bien creado y configurado ya que visualizamos el valor de los Tags en tiempo real y sin errores como se muestra en la Tabla 29.

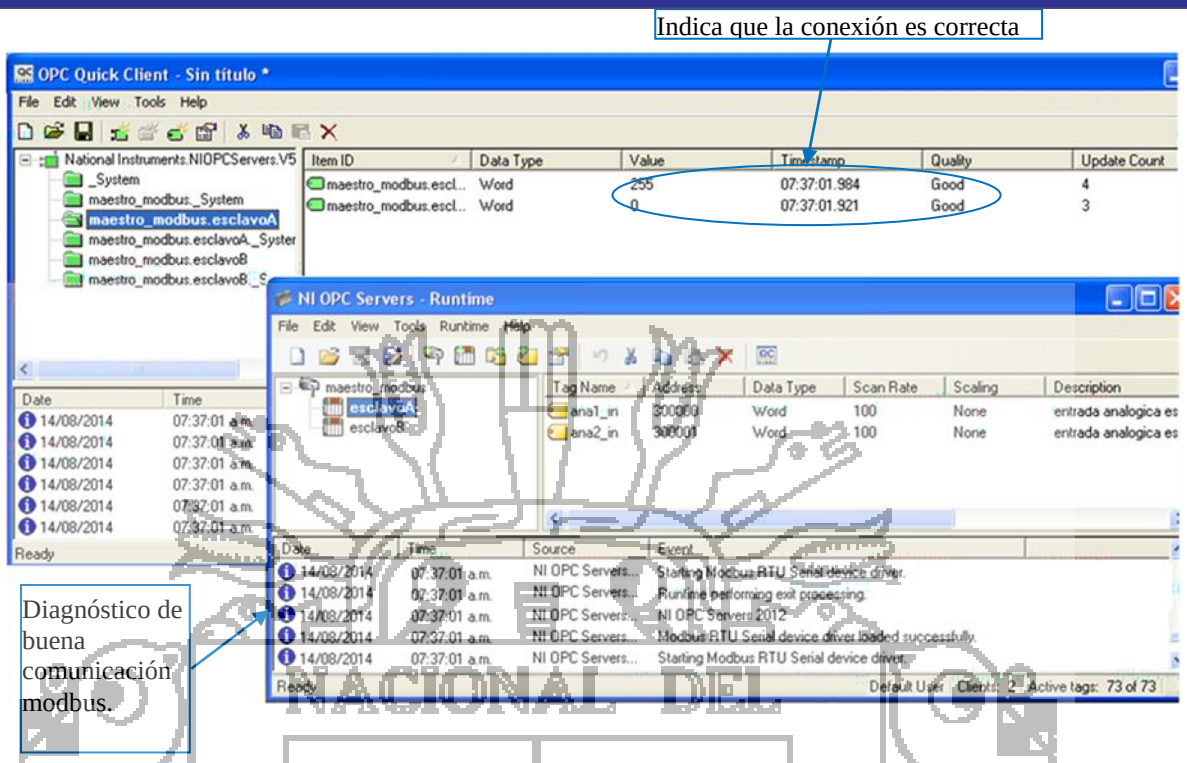



*Figura 168 - OPC Server creado y arranque del OPC Quick Client*

| Nº | valor | Estado de comunicación |
|---|---|---|
| 1 | 255 | Buena |
| 2 | 0 | Buena |

*Tabla 29 - Estado de comunicación buena.*

## CONCLUSIONES

**PRIMERO:** Se Implementó la estructura y la norma que rige el protocolo Modbus, que cumple en términos Generales con las Especificaciones propuestas para la implementación de Redes Industriales, y que además se adquirió una tarjeta Electrónica que se encarga de Convertir Señales RS232 a RS485 y viceversa, que permite conectar los dispositivos esclavos y el dispositivo Maestro en la misma Red Modbus.

**SEGUNDO:** El protocolo MODBUS es el encargado de conectar los dispositivos esclavos adquiriendo datos y de llevar la información hacia la PC, dispositivo maestro principal de la Red implementada.

**TERCERO:** Se implementó una Aplicación para un Maestro Modbus, que permite efectividad para el monitoreo de variables físicas, es confiabilidad porque no tiene errores, eficiencia buena velocidad de transmisión y facilidad (interface amigable con el usuario) que proporciona el uso del control de Comunicación Modbus utilizando la Tecnología NI OPC Server de labview en ella se encuentran las direcciones configuradas para los datos, canal, velocidad de transmisión, el bit de paridad, puerto de serie, control de flujo, comprobándose que las tramas generadas en la maestro modbus como respuestas a las peticiones simuladas por el ModScan32 cumplen con los requerimientos de dicho protocolo, por tanto se concluye que efectivamente la parte correspondiente a la generación de tramas modbus es correcta, dentro de un software de aplicación no necesita contar con los drivers para controlar los dispositivos esclavos.

.

## RECOMENDACIONES

**PRIMERO:** El Protocolo Modbus, ofrece dos Modos de Transmisión Serie de Datos, el Modo ASCII y el Modo RTU (Remote Terminal Unit), sin embargo, para este proyecto solo se implementó tanto en los dispositivos Esclavos, como en el Control de Comunicación modbus el modo de transmisión RTU, por ser el más usado por los dispositivos Modbus comerciales.

**SEGUNDO:** Al momento de diseñar un circuito electrónico para una red modbus se debe tomar muy en cuenta la velocidad de transmisión, modulación de los datos.

**TERCERO:** Si se desea implementar una red industrial de distancia a 1200m se deberá colocar repetidores para así evitar la atenuación de señal.

**CUARTO:** Se puede ampliar la cantidad de variables a supervisar, incrementando la cantidad de variables leídas por él NI OPC Server.

## BIBLIOGRAFIA

[1]. Carballar, José, (2003). El Libro de las Comunicaciones del PC, (1ra ed.), Alfaomega-Rama.

[2]. Jan Axelson, Designing RS-485 Circuits, de

http://janaxelson.com/rs-485_circuits.htm

[3]. Juan del C., (1998), Tendencias en Arquitecturas de Control Sistemas Abiertos y OPC, de

file:///C:/Users/electronic/Downloads/Sistemas%20Abiertos%20y%20OPC.pdf

[4]. Jose Soriano M., (2012), Diseño de un extensor de entradas y salidas analógicas por MODBUS RTU sobre RS – 485, de

http://openaccess.uoc.edu/webapps/o2/bitstream/10609/19273/6/jsorianomiTFC0113memoria.pdf

[5]. Kepware, (2014), About OPC - OPen Connectivity through Open Standards, de

http://www.kepware.com/Menu_items/industry_OPC_Foundation.asp

[6]. Luis Eduardo G., (2009). Control Digital Teoría y práctica, (2da ed.).

[7]. Modbus Akros, Comunicaciones Modbus Serie Akros Manual de instrucciones, de

http://www.ditel.es/manuales/obsoletos/reguladores/Modbus_Akros_Cas.pdf

[8]. Modbus Org., MODBUS over Serial Line Specification & Implementation guide V1.0, de

https://www.google.com.pe/#q=modbus%20protocol%20guide

[9]. Modbus Org., Modicon Modbus Protocol Reference Guide PI–MBUS–300 Rev. J, de

http://web.eecs.umich.edu/~modbus/documents/PI_MBUS_300.pdf

[10]. NI OPC Labview, (2012), How LabView UsesI/o Servers, de

http://www.ni.com/white-paper/13865/en/

[11]. NI OPC Labview, (2012), Connect LabVIEW to Any PLC With Modbus, de http://www.ni.com/white-paper/13911/en/

[12]. Thomas Floyd, (2000), Fundamentos DE SISTEMAS, (7ma ed.), Digitales Edit. Prentice Hall.

[13]. Roger Pressman, (2012). Ingeniería del Software, Un Enfoque Práctico, (6ta ed.), Editorial Mc Graw Hill.

[14]. Vicente Guerrero, Ramón L., Luis Martínez, (2009). Comunicaciones Industriales, (1ra ed.), Editorial Alfaomega. México.

# ANEXOS

# ANEXO I

**Datasheet**

# Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

## Pin Description

| PIN | | | | | NAME | FUNCTION |
|---|---|---|---|---|---|---|
| MAX481/MAX483/ MAX485/MAX487/ MAX1487 | | MAX488/ MAX490 | | MAX489/ MAX491 | | |
| DIP/SO | µMAX | DIP/SO | µMAX | DIP/SO | | |
| 1 | 3 | 2 | 4 | 2 | RO | Receiver Output: If A > B by 200mV, RO will be high; If A < B by 200mV, RO will be low. |
| 2 | 4 | — | — | 3 | $\overline{RE}$ | Receiver Output Enable. RO is enabled when $\overline{RE}$ is low; RO is high impedance when $\overline{RE}$ is high. |
| 3 | 5 | — | — | 4 | DE | Driver Output Enable. The driver outputs, Y and Z, are enabled by bringing DE high. They are high impedance when DE is low. If the driver outputs are enabled, the parts function as line drivers. While they are high impedance, they function as line receivers if $\overline{RE}$ is low. |
| 4 | 6 | 3 | 5 | 5 | DI | Driver Input. A low on DI forces output Y low and output Z high. Similarly, a high on DI forces output Y high and output Z low. |
| 5 | 7 | 4 | 6 | 6, 7 | GND | Ground |
| — | — | 5 | 7 | 9 | Y | Noninverting Driver Output |
| — | — | 6 | 8 | 10 | Z | Inverting Driver Output |
| 6 | 8 | — | — | — | A | Noninverting Receiver Input and Noninverting Driver Output |
| — | — | 8 | 2 | 12 | A | Noninverting Receiver Input |
| 7 | 1 | — | — | — | B | Inverting Receiver Input and Inverting Driver Output |
| — | — | 7 | 1 | 11 | B | Inverting Receiver Input |
| 8 | 2 | 1 | 3 | 14 | $V_{CC}$ | Positive Supply: 4.75V ≤ $V_{CC}$ ≤ 5.25V |
| — | — | — | — | 1, 8, 13 | N.C. | No Connect—not internally connected |

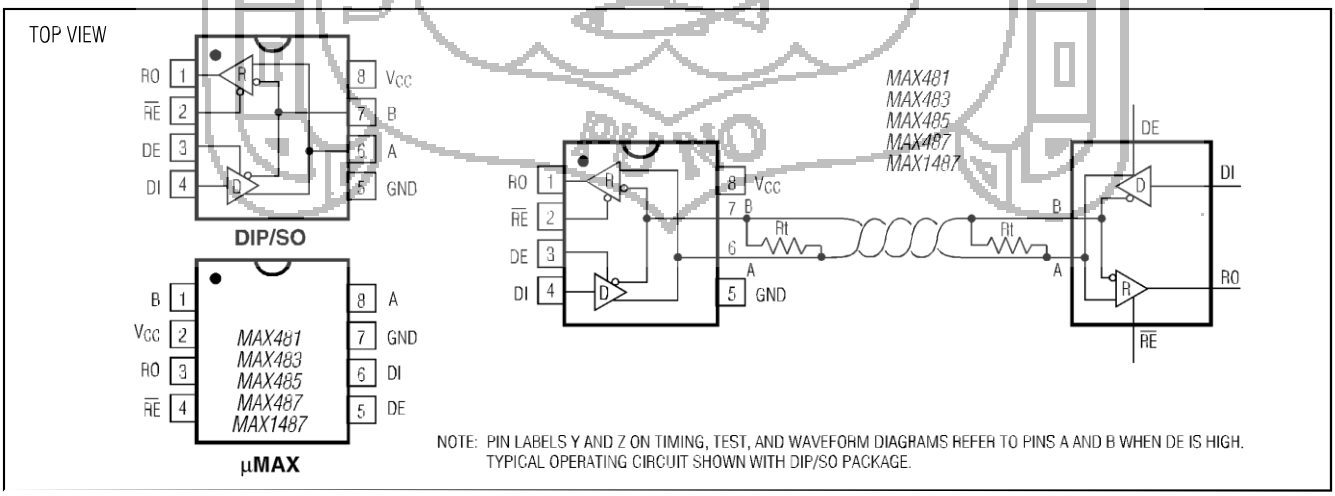



*Figure 1. MAX481/MAX483/MAX485/MAX487/MAX1487 Pin Configuration and Typical Operating Circuit*

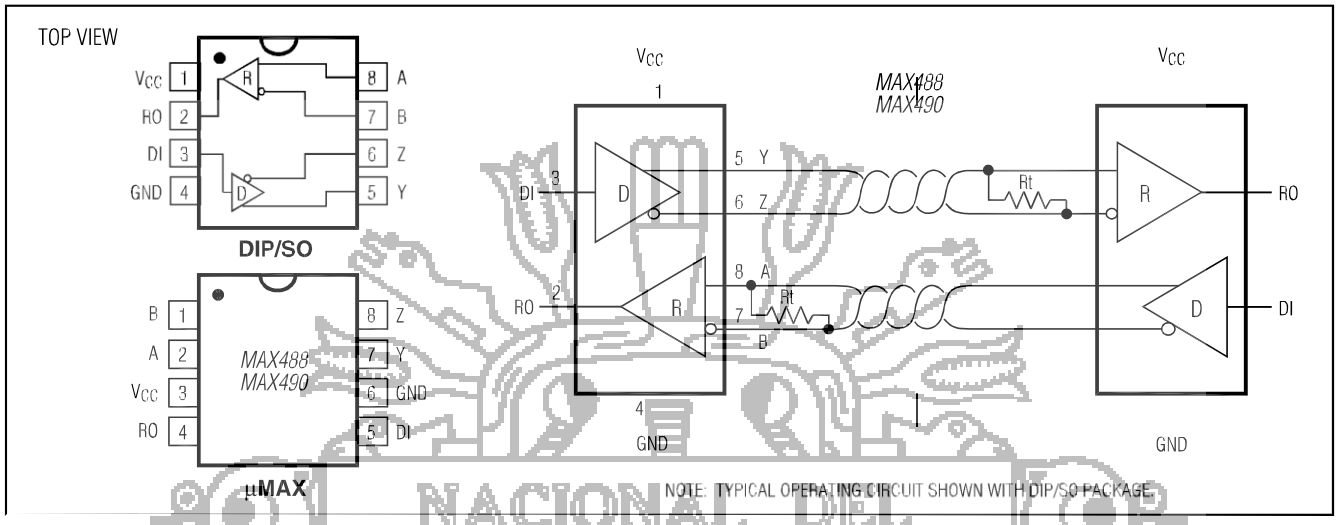



Figure 2. MAX488/MAX490 Pin Configuration and Typical Operating Circuit

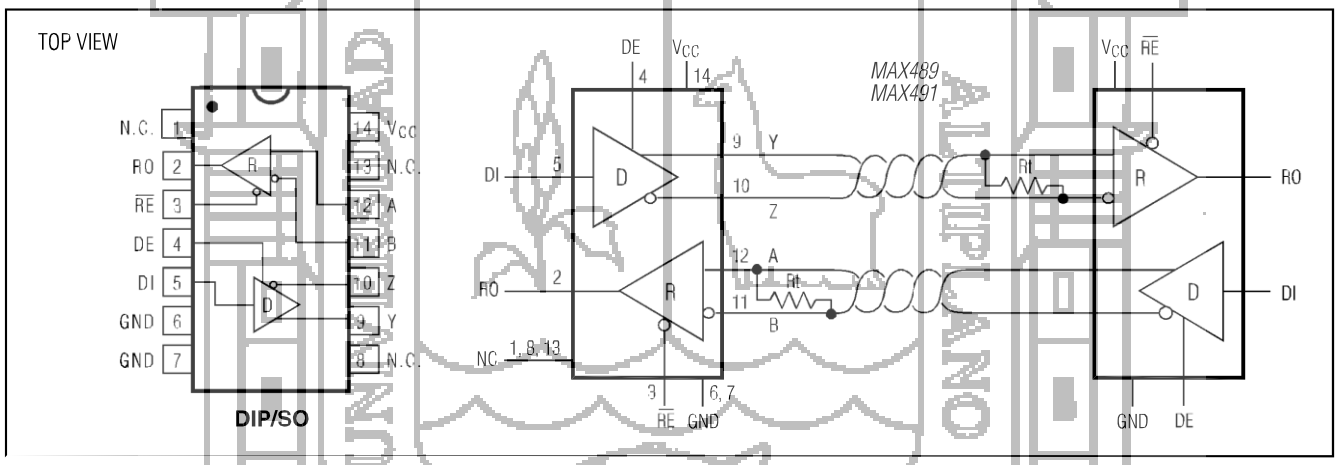



Figure 3. MAX489/MAX491 Pin Configuration and Typical Operating Circuit

## Applications Information

The MAX481/MAX483/MAX485/MAX487–MAX491 and MAX1487 are low-power transceivers for RS-485 and RS-422 communications. The MAX481, MAX485, MAX490, MAX491, and MAX1487 can transmit and receive at data rates up to 2.5Mbps, while the MAX483, MAX487, MAX488, and MAX489 are specified for data rates up to 250kbps. The MAX488–MAX491 are full-duplex transceivers while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are half-duplex. In addition, Driver Enable (DE) and Receiver Enable ($\overline{RE}$) pins are included on the MAX481, MAX483, MAX485, MAX487, MAX489, MAX491, and MAX1487. When disabled, the driver and receiver outputs are high impedance.

### MAX487/MAX1487: 128 Transceivers on the Bus

The 48kΩ, ¼-unit-load receiver input impedance of the MAX487 and MAX1487 allows up to 128 transceivers on a bus, compared to the 1-unit load (12kΩ input impedance) of standard RS-485 drivers (32 transceivers maximum). Any combination of MAX487/MAX1487 and other RS-485 transceivers with a total of 32 unit loads or less can be put on the bus. The MAX481/MAX483/MAX485 and MAX488–MAX491 have standard 12kΩ Receiver Input impedance.

## Test Circuits

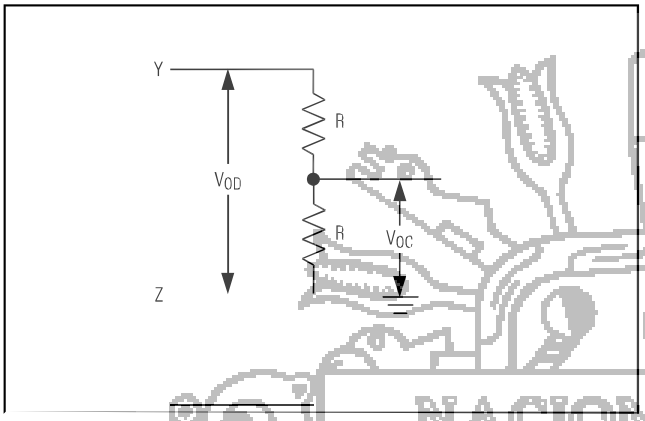



Figure 4. Driver DC Test Load

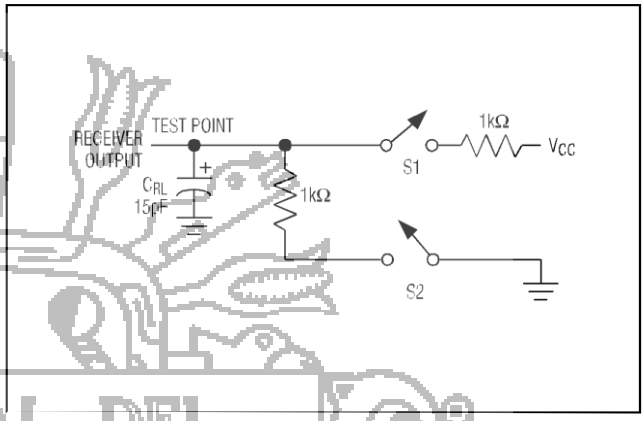



Figure 5. Receiver Timing Test Load

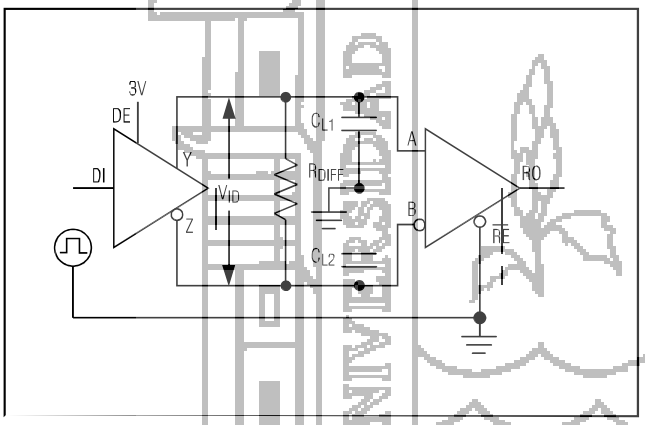



Figure 6. Driver/Receiver Timing Test Circuit

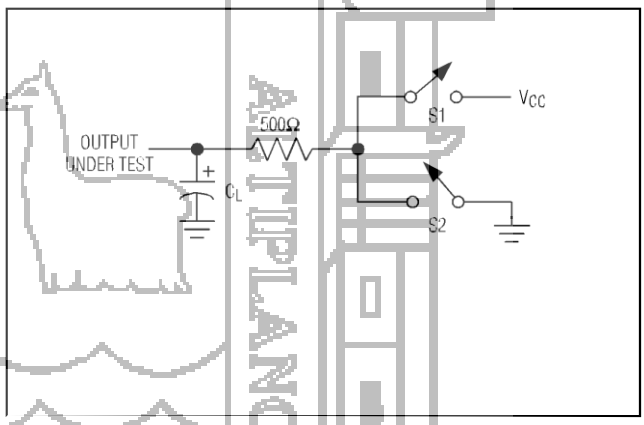


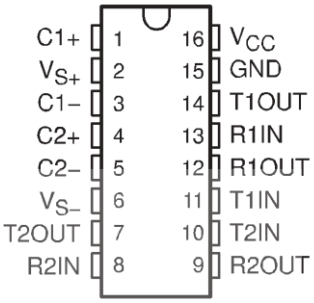

Figure 7. Driver Timing Test Load

### MAX483/MAX487/MAX488/MAX489: Reduced EMI and Reflections

The MAX483 and MAX487–MAX489 are slew-rate limited, minimizing EMI and reducing reflections caused by improperly terminated cables. Figure 12 shows the driver output waveform and its Fourier analysis of a 150kHz signal transmitted by a MAX481, MAX485, MAX490, MAX491, or MAX1487. High-frequency harmonics with large amplitudes are evident. Figure 13 shows the same information displayed for a MAX483, MAX487, MAX488, or MAX489 transmitting under the same conditions. Figure 13's high-frequency harmonics have much lower amplitudes, and the potential for EMI is significantly reduced.

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates Up To 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202
- Applications
  - TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)




## description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

**ORDERING INFORMATION**

| | PACKAGE† | | ORDERABLE ART NUMBER | SIDE M RKING |
|---|---|---|---|---|
| 0 C to 7 C | PDI (N) | Tube of 25 | M X232N | M X232N |
| | SOIC (D) | Tube of 40 | M X232D | M X232 |
| | SOIC (DW) | | M | X232 |
| | | | X232IN | |
| | SOIC (D) | | X232ID | |
| | SOIC (DW) | | X232IDW | X232I |
| | | | X232IDWR | |

† Package drawings, standard packing quantities, thermal data, symbolization, anc PCB design guidelines are available at www.ti.com/sc/package.



TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

## Function Tables

**EACH DRIVER**

| INPUT TIN | OUTPUT TOUT |
|---|---|
| L | H |
| H | L |

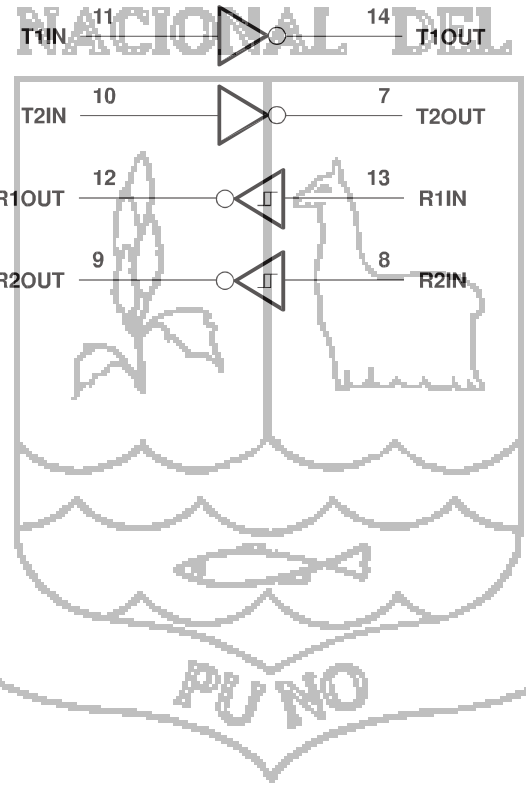
H high evel L low level

**EACH RECEIVER**

| IN UT RIN | OUTPUT ROUT |
|---|---|
| L | H |
| H | L |

H high evel L low level

## logic d iagram (positive logic)




TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

# LM555
# Timer

## General Description

The LM555 is a highly stable device for generating accurate time delays or oscillation. Additional terminals are provided for triggering or resetting if desired. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For astable operation as an oscillator, the free running frequency and duty cycle are accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output circuit can source or sink up to 200mA or drive TTL circuits.

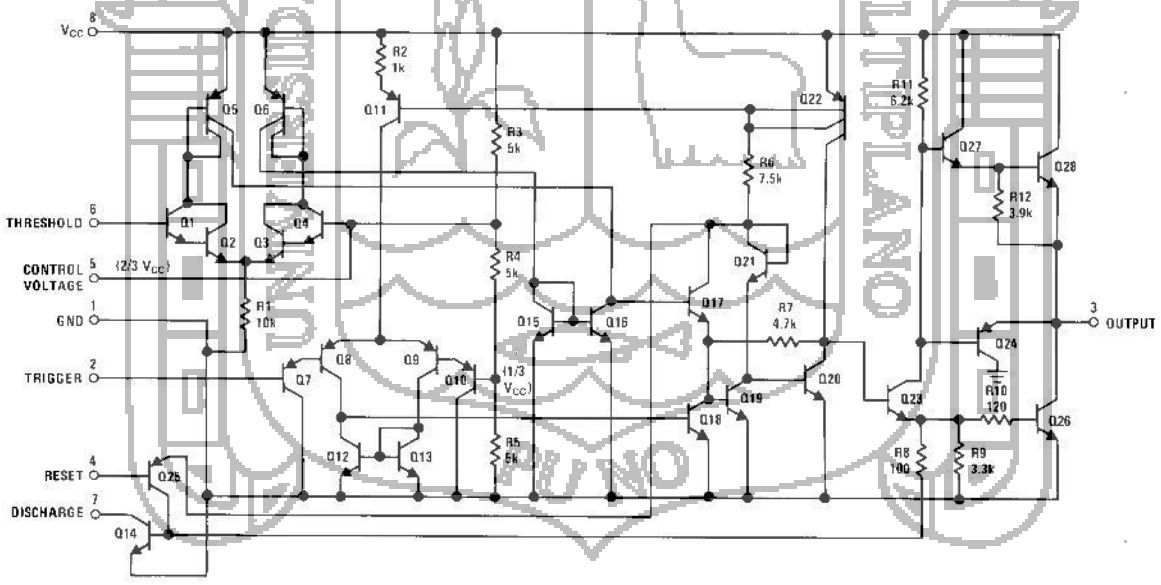
## Features

- Direct replacement for SE555/NE555
- Timing from microseconds through hours
- Operates in both astable and monostable modes
- Adjustable duty cycle
- Output can source or sink 200 mA
- Output and supply TTL compatible
- Temperature stability better than 0.005% per °C
- Normally on and normally off output
- Available in 8-pin MSOP package

## Applications

- Precision timing
- Pulse generation
- Sequential timing
- Time delay generation
- Pulse width modulation
- Pulse position modulation
- Linear ramp generator

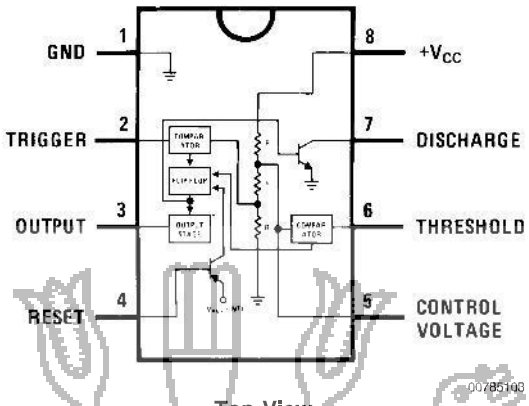
## Schematic Diagram




00785101

 DS007851

Dual-In-Line, Small Outline
and Molded Mini Small Outline Packages




Top View

## Ordering Information

| Package | Part Number | Package Marking | Media Transport | NSC Drawing |
|---|---|---|---|---|
| 8-Pin SOIC | LM555CM | LM555CM | Rails | M08A |
| | LM555CMX | LM555CM | 2.5k Units Tape and Reel | |
| 8-Pin MSOP | LM555CMM | Z55 | 1k Units Tape and Reel | MUA08A |
| | LM555CMMX | Z55 | 3.5k Units Tape and Reel | |
| 8-Pin MDIP | LM555CN | LM555CN | Rails | N08E |

**please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.**

| | |
|---|---|
| Supply Voltage | +18V |
| Power Dissipation (Note 3) | |
| LM555CM, LM555CN | 1180 mW |
| LM555CMM | 613 mW |
| Operating Temperature Ranges | |
| LM555C | 0˚C to +70˚C |
| Storage Temperature Range | −65˚C to +150˚C |

| | |
|---|---|
| Soldering (10 Seconds) | 260˚C |
| Small Outline Packages | |
| (SOIC and MSOP) | |
| Vapor Phase (60 Seconds) | 215˚C |
| Infrared (15 Seconds) | 220˚C |

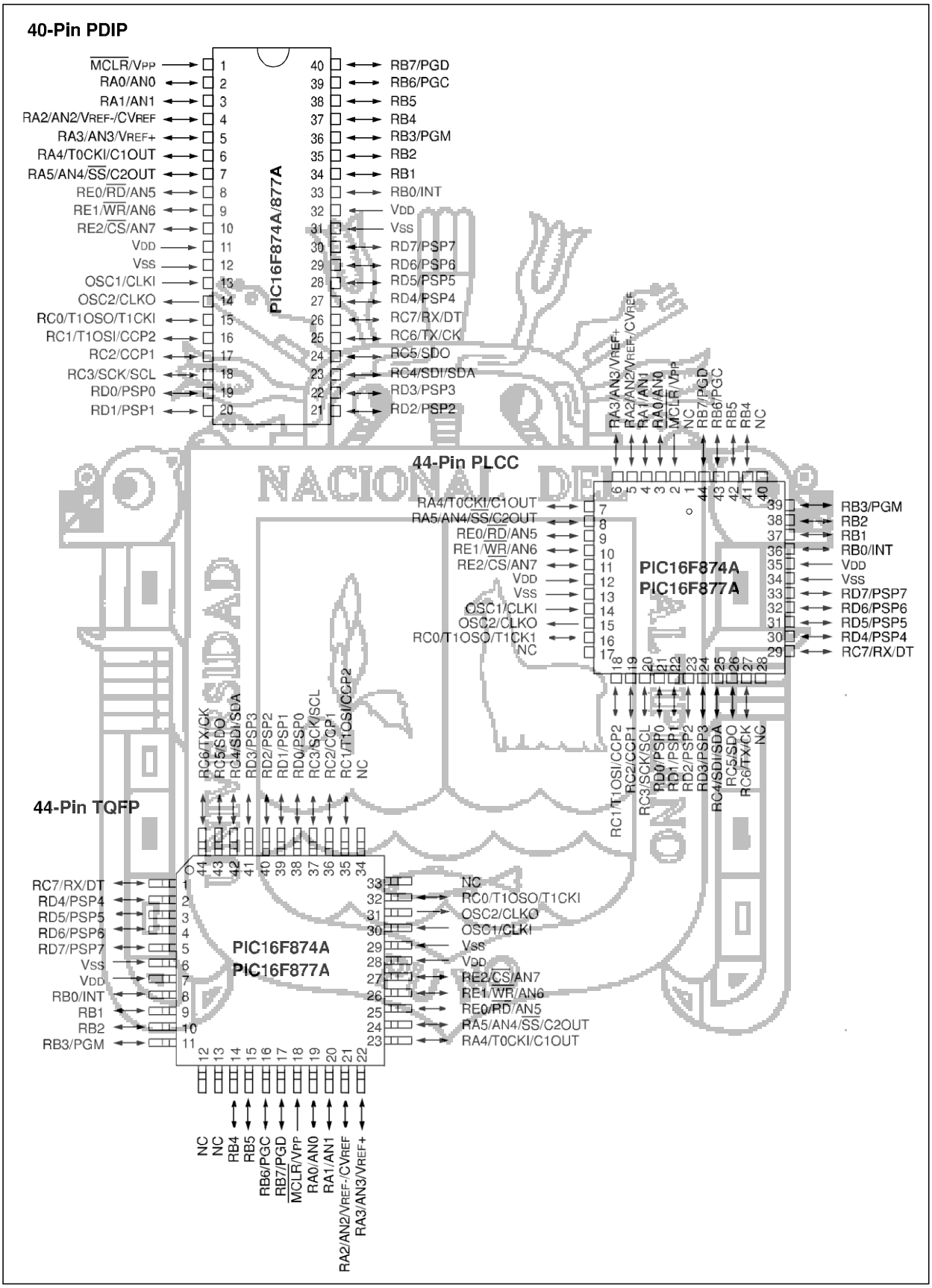
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.

## Electrical Characteristics (Notes 1, 2)

($T_A$ = 25˚C, $V_{CC}$ = +5V to +15V, unless otherwise specified)

| Parameter | Conditions | Limits LM555C | | | Units |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| Supply Voltage | | 4.5 | | 16 | V |
| Supply Current | $V_{CC}$ = 5V, $R_L$ = ∞ | | 3 | 6 | mA |
| | $V_{CC}$ = 15V, $R_L$ = ∞ (Low State) (Note 4) | | 10 | 15 | mA |
| Timing Error, Monostable | | | | | |
| Initial Accuracy | | | 1 | | % |
| Drift with Temperature | $R_A$ = 1k to 100kΩ, C = 0.1µF, (Note 5) | | 50 | | ppm/˚C |
| Accuracy over Temperature | | | 1.5 | | % |
| Drift with Supply | | | 0.1 | | %/V |
| Timing Error, Astable | | | | | |
| Initial Accuracy | | | 2.25 | | % |
| Drift with Temperature | $R_A$, $R_B$ = 1k to 100kΩ, C = 0.1µF, (Note 5) | | 150 | | ppm/˚C |
| Accuracy over Temperature | | | 3.0 | | % |
| Drift with Supply | | | 0.30 | | %/V |
| Threshold Voltage | | | 0.667 | | x $V_{CC}$ |
| Trigger Voltage | $V_{CC}$ = 15V | | 5 | | V |
| | $V_{CC}$ = 5V | | 1.67 | | V |
| Trigger Current | | | 0.5 | 0.9 | µA |
| Reset Voltage | | 0.4 | 0.5 | 1 | V |
| Reset Current | | | 0.1 | 0.4 | mA |
| Threshold Current | (Note 6) | | 0.1 | 0.25 | µA |
| Control Voltage Level | $V_{CC}$ = 15V | 9 | 10 | 11 | V |
| | $V_{CC}$ = 5V | 2.6 | 3.33 | 4 | V |
| Pin 7 Leakage Output High | | | 1 | 100 | nA |
| Pin 7 Sat (Note 7) | | | | | |
| Output Low | $V_{CC}$ = 15V, $I_7$ = 15mA | | 180 | | mV |
| Output Low | $V_{CC}$ = 4.5V, $I_7$ = 4.5mA | | 80 | 200 | mV |

**Pin Diagrams (Continued)**



## Table of Contents



### TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our Web site at **www.microchip.com/cn** to receive the most current information on all of our products.

## 1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implementec only on the 40/44-pin devices

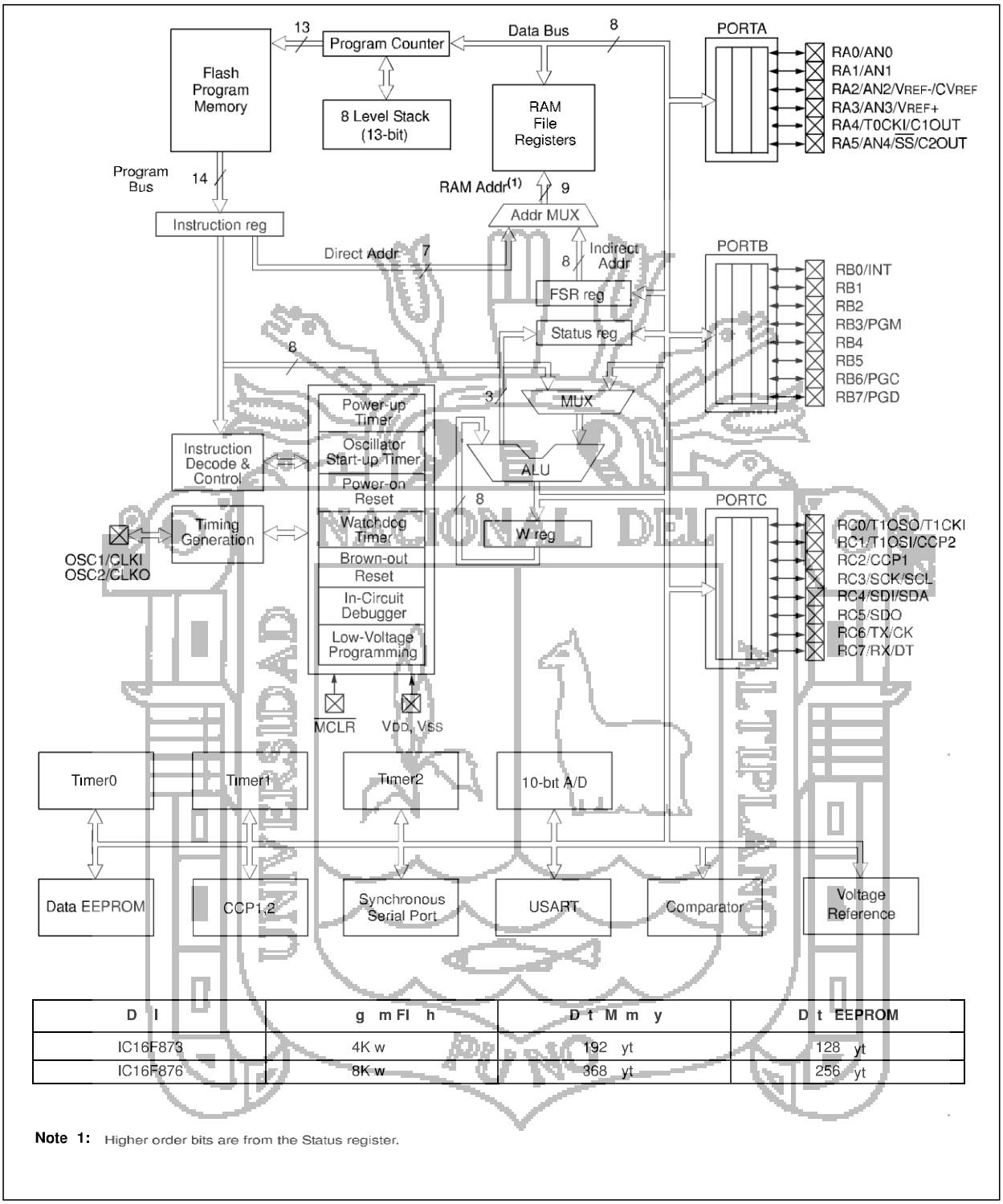
The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

**TABLE 1-1: PIC16F87XA DEVICE FEATURES**

| Key Features | IC16F873A | | | |
|---|---|---|---|---|
| O er ting Frequency | DC – 20 | MHz | | |
| Resets (and Delays) | POR BO (PWRT OST) | POR BOR (PWRT OST) | | POR B (PWRT OST) |
| Fl sh Progr m Memory (14 bit words) | | | | |
| Data Memory (bytes) | | | | |
| | | | | |
| | | | | |
| | | E | | |
| | | | | |
| arp/PWM modules | | | | |
| Serial Communic | | | | |
| rallel Communications | | | | |
| | 5 input channels | | 5 input channels | 5 input channels |
| log Comparators | | | | |
| | 35 Instructions | 35 Instructions | 35 Instructions | 35 Instructions |
| | 28<br>28<br>28<br>28 | 40 pin PDIP<br>44 pin PLCC<br>44 pin TQFP<br>44 in QFN | 28<br>28<br>28<br>28 | 40 pin PDIP<br>44 pin PLCC<br>44 in TQFP<br>44 pin FN |

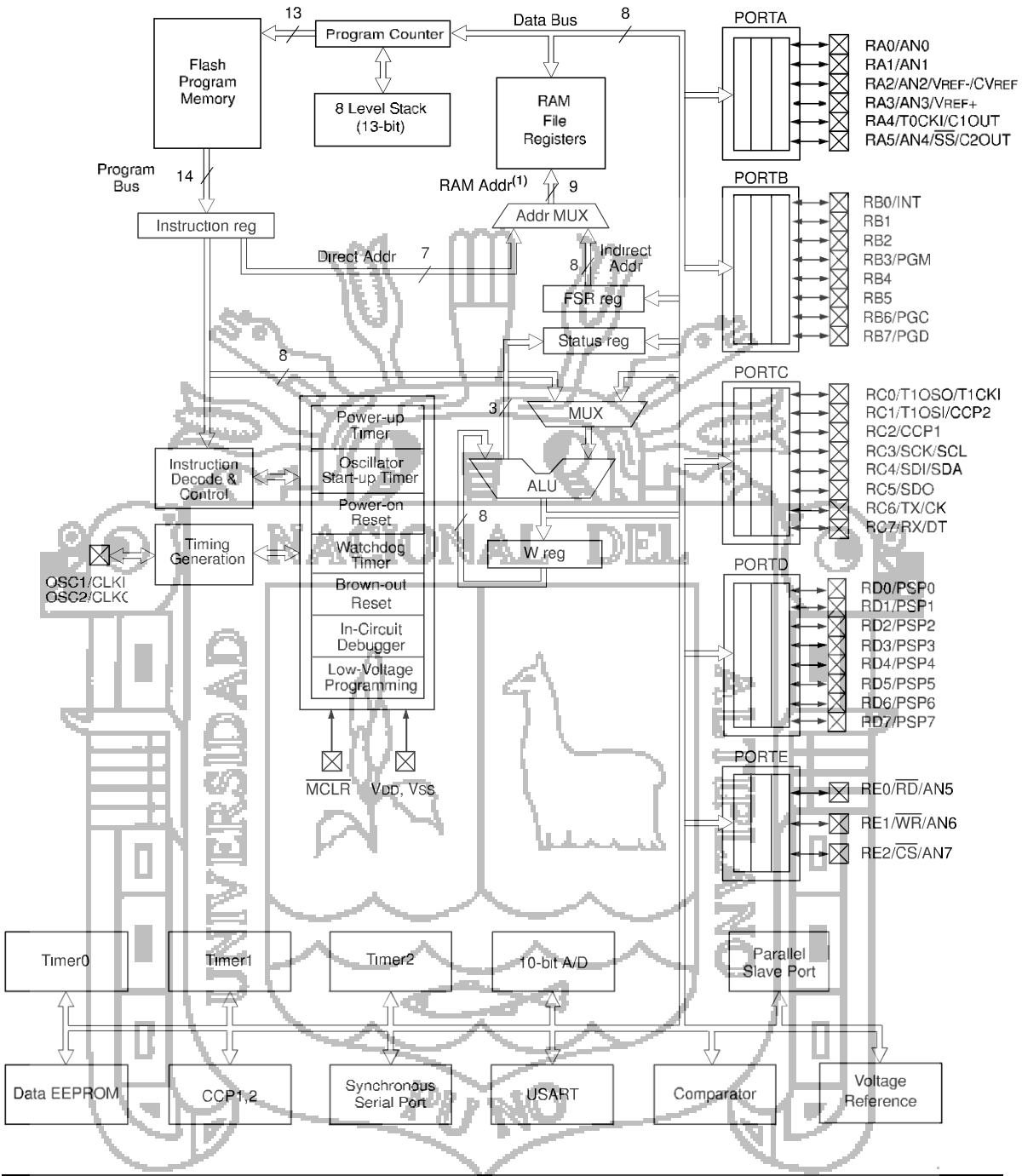
**FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM**




| D I | g m Fl h | D t M m y | D t EEPROM |
|---|---|---|---|
| IC16F873 | 4K w | 192 yt | 128 yt |
| IC16F876 | 8K w | 368 yt | 256 yt |

**Note 1:** Higher order bits are from the Status register.

**FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM**




| D i | P g m Fl h | D t M m y | D t EE ROM |
|---|---|---|---|
| IC16F874 | 4K w d | 192 yt | 128 yt |
| IC16F877 | 8K w d | 368 Byt | 256 Byt |

**Note 1:** Higher order bits are from the Status register.

# HIH-4000 Series

## Humidity Sensors

### DESCRIPTION

The HIH-4000 Series Humidity Sensors are designed specifically for high volume OEM (Original Equipment Manufacturer) users.

Direct input to a controller or other device is made possible by this sensor's near linear voltage output. With a typical current draw of only 200 µA, the HIH-4000 Series is often ideally suited for low drain, battery operated systems.

Tight sensor interchangeability reduces or eliminates OEM production calibration costs. Individual sensor calibration data is available.

The HIH-4000 Series delivers instrumentation-quality RH (Relative Humidity) sensing performance in a competitively priced, solderable SIP (Single In-line Package).

Available in two lead spacing configurations, the RH sensor is a laser trimmed, thermoset polymer capacitive sensing element with on-chip integrated signal conditioning.

The sensing element's multilayer construction provides excellent resistance to most application hazards such as wetting, dust, dirt, oils and common environmental chemicals.

### FEATURES

- Molded thermoset plastic housing
- Near linear voltage output vs. % RH
- Laser trimmed interchangeability
- Low power design
- Enhanced accuracy
- Fast response time
- Stable, low drift performance
- Chemically resistant

### POTENTIAL APPLICATIONS

- Refrigeration equipment
- HVAC (Heating, Ventilation and Air Conditioning) equipment
- Medical equipment
- Drying
- Metrology
- Battery-powered systems
- OEM assemblies

# HIH-4000 Series

**Table 1 Performance S ecifications (At 5 Vdc su ply and 25 C [77 F] un ess otherwise notec )**

| Parameter | Minimum | Typica | Maximum | Unit | Specific Note |
|---|---|---|---|---|---|
| nterchangeabi ity (first order curve) | – | – | – | – | – |
| 0% RH to 59% RH | 5 | – | 5 | % H | – |
| 60% RH to 100% RH | 8 | – | 8 | % H | – |
| Accuracy (best fit straight ine) | 3 5 | – | +3 5 | % H | 1 |
| Hysterisis | – | 3 | – | % H | – |
| Repe tabi ity | – | +0 5 | – | % H | – |
| Sett ing time | – | – | 70 | ms | – |
| Response time (1/e in s ow moving ir) | – | 5 | – | s | – |
| Stabi ity (at 50% RH) | – | 1 2 | – | % H | – |
| Vo t ge sup y | 4 | – | 5 8 | Vdc | 2 |
| Current supp y | – | 200 | 500 | µA | – |
| Vo t ge output (1 order curve fit) | $V_{OU}$ =($V_{SU}$ )(0 0062(sensor RH) + 0 16) typica t 25 C<br>True RH = (Sensor RH)/(1 0546 – 0 00216T) T in C | | | | |
| Output voltage 50% RH 5 V | – | 4 | – | mV/ C | |
| | 40[ 40] | See Figure 1 | 85[185] | C[ F] | – |
| | 0 | See Figure 1 | 100 | % H | 3 |
| | 50[ 58] | – | 125[257] | C[ F] | – |
| | | See Figure 2 | | % H | 3 |

**Specific Notes:**

Can only be achieved with the supplied slope and offset. For HIH-4000-003 and HIH-4000-004 catalog listings only. Device is calibrated at 5 Vdc and 25 °C.

3. Non-condensing environment.

**General Notes:**

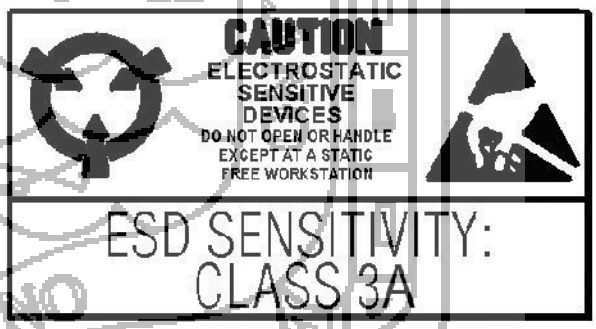
- Sensor is ratiometric to supply voltage.
- Extended exposure to >90% RH causes a reversible shift of 3% RH.
- Sensor is light sensitive. For best performance, shield sensor from bright light.

**FACTORY CALIBRATION DATA**

HIH-4000 Sensors may be ordered with a calibration and data printout. See Table 2 and the order guide on the back page.

**Table 2. Example Data Printout**

| | |
|---|---|
| Model | HIH 4000 003 |
| Ch nnel | |
| Wafer | 0309 |
| MRP | 337313 |
| ted values at 5 V | |
| accuracy at 25<br>Zero offset | 31 483 mV/%RH<br>zero offset)/slope<br>0 826)/0 0315 |
| Ratiometric response<br>0% RH to 100% RH | $V_{SU}$ |



# Integrated Silicon Pressure Sensor Altimeter/Barometer Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

**MPX4115 SERIES**

**OPERATING OVERVIEW**
**INTEGRATED PRESSURE SENSOR**
**15 to 115kPa (2.18 to 16.7 psi)**
**0.2 to 4.8 Volts Output**

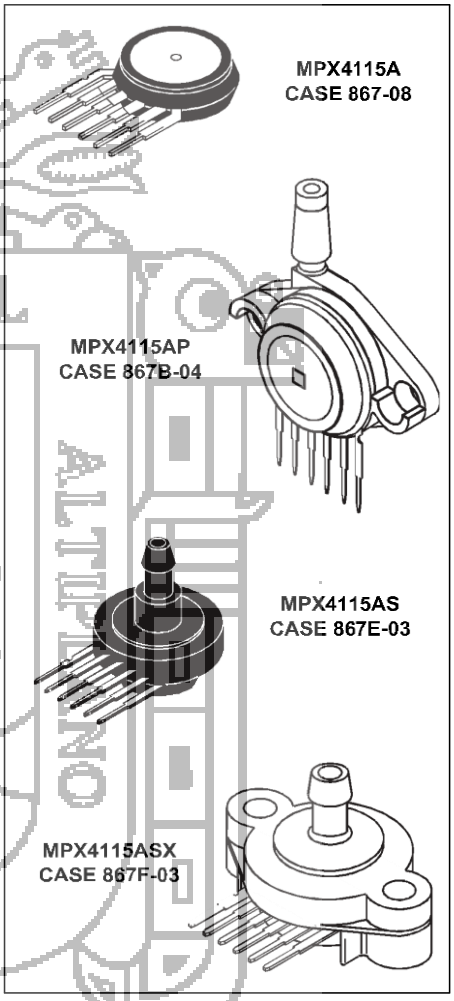
The MPX4115 series is designed to sense absolute air pressure in an altimeter or barometer (BAP) applications. Freescale's BAP sensor integrates on-chip, bipolar op amp circuitry and thin film resistor networks to provide a high level analog output signal and temperature compensation. The small form factor and high reliability of on-chip integration makes the Freescale BAP sensor a logical and economical choice for application designers.

### Features

- 1.5% Maximum Error over 0° to 85°
- Ideally suited for Microprocessor or Microcontroller-Based Systems
- Available in Absolute, Differential and Gauge Configurations
- Durable Epoxy Unibody Element
- Easy-to-Use Chip Carrier Option

### Typical Applications

- Altimeter
- Barometer




**ORDERING INFORMATION**[1]

| Device | | | MPX Series Order No | |
|---|---|---|---|---|
| Basic Element | | Case 867-08 | | |
| Ported Elements | | Case 867B-04 | | |
| | | Case 867E-03 | | |
| | Absolute Axial Port | Case 867F-03 | | |

1. The MPX4115A BAP Sensor is available in the Basic Element package or with pressure port fittings that provide mounting ease and barbed hose connections.

**PIN NUMBERS**

| | | | |
|---|---|---|---|
| 1 | $V_{OUT}$[1] | 4 | N/C[2] |
| 2 | GND | 5 | N/C[2] |
| 3 | $V_S$ | 6 | N/C[2] |

1. Pin 1 is noted by the notch in the lead.
2. Pins 4, 5, and 6 are internal device connections. Pin 1 is noted by the notch in the Lead. Do not connect to external circuitry or ground.



freescale™

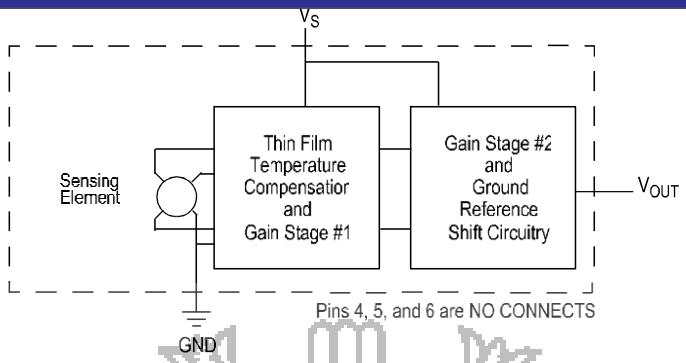



**Figure 1. Integrated Pressure Sensor Schematic**

**Table 1. Maximum Ratings(1)**

| Parametrics | Sym ol | Valu | Unit |
|---|---|---|---|
| Overpressure(2) ( | $P_m$ | 400 | k a |
| ressure(2) (P1 | $P_b$ t | 1000 | k a |
| | $T_{tg}$ | 40 to 125 | C |
| Operating Tem | $T_A$ | 40 to 125 | C |

1. $T_C$ = 25°C unless otherwise noted.
2. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

## INTRODUCTION

Linear temperature sensors have a major advantage. The output can be easily conditioned to achieve a desired voltage output span over a particular temperature range. A linear output voltage allows ease of interface to data acquisition systems and programmable controllers. By adjusting the circuit gain, the sensitivity of the output can be adjusted over the total range such as 10° to +40° C.

## INTERFACING WITH 1-5V CIRCUIT

If more than 1 mA of current flows through the TD, self-heating will occur. The self-heating effect is typically 0.2° C/milliwatt. The circuits in **Figure 1** and **Figure 2** provide a maximum current flow of 1 mA.

## SETTING DESIRED SPAN

The circuit gain depends on the temperature range you want to sense. The offset adjustment is, in turn, dependent on the chosen gain. The transfer function for both circuits (Figures 1 and 2) is as follows:

$$(R_5/R_4 + 1) \bullet V[R_{TD}/(R_{TD} + R_7)] - (R_5/R_4)(1 + R_3/R_2)V_1 = v_O$$

Only two elements are unknown: the offset ($v_1$), and the circuit gain ($R_5/R_4 + 1$). To set the desired span, two equations for the two unknowns must be created and solved. To simplify these calculations, the following assumption is made:

$$R_5/R_4 = R_2/R_3$$

The second assumption is that no self-heating of the TD element will occur: the values of V and $R_7$ are constant at the values indicated.

$$V[R_{TD}/(R_{TD} + R_7)] = 5[R_{TD}/(R_{TD} + 5110)]$$

These assumptions reduce the transfer function to:

$$(R_5/R_4 + 1) \bullet 5[R_{TD}/(R_{TD} + 5110)] - (R_5/R_4 + 1)v_1 = v_O$$

To create the first of the two simultaneous equations, the value of $R_{TD}$ for the desired minimum temperature is taken from **Table 1**. ($R_{TD}$ at 20° C equals 2000 Ohms, and $v_O$ = 1V.) For the second equation, the value of $R_{TD}$ for the desired maximum temperature is taken from the table, and $v_O$ = 5 V.

The two equations are then solved for the gain ($R_5/R_4 + 1$) and the offset ($v_1$). The following example shows how this is accomplished.

Desired temperature range: 0° to 60° C.
Voltage output over range: 1 to 5 V.

Equation 1: $R_{TD}$ at 0° C is 1854 Ohms.

$$(R_5/R_4 + 1) \bullet 5[1854/(1854 + 5110)] - (R_5/R_4 + 1)v_1 = 1 \text{ V}$$

Equation 2: $R_{TD}$ at 60° C is 2314 Ohms.

$$(R_5/R_4 + 1) \bullet 5[2314/(2314 + 5110)] - (R_5/R_4 + 1)v_1 = 5 \text{ V}$$

Step 1: subtract equation 1 from equation 2.

$$(R_5/R_4 + 1)(1.558) - (R_5/R_4 + 1)V_1 = 5$$
$$(R_5/R_4 + 1)(1.331) - (R_5/R_4 + 1)v_1 = 1$$
$$(R_5/R_4 + 1)(.227) - 0 = 4$$
$$(R_5/R_4 + 1) = 4(1/.227)$$
$$(R_5/R_4 + 1) = 17.62 = \text{GAIN}$$

Step 2: substitute $(R_5/R_4 + 1) = 17.62$ into equation 1 and solve for $V_1$.

$$(17.62)(1.331) - (17.62)v_1 = 1$$
$$23.454 - 17.62v_1 = 1$$
$$22.452 = 17.62v_1$$
$$1.274 = v_1 = \text{OFFSET}$$

In order to transfer this information into the circuit in **Figure 1**, choose appropriate values for $R_4$ and $R_5$ such that:

$$(R_5/R_4 + 1) = \text{GAIN}$$

For this example, $R_4$ = 1 K Ohm and $R_5$ = 16.62 K Ohm would be appropriate.

Choose $R_2$ and $R_3$ based on $R_2/R_3 = R_5/R_4$. For this example, choose $R_2 = R_5$ = 16.62 K Ohm, and $R_3 = R_4$ = 1 K Ohm.

To set the offset $v_1$ using potentiometer $R_1$, temporarily insert an equivalent discrete resistor in place of the TD element. It should be equal to the TD resistance at the minimum desired temperature (1854 Ohms from the example). Adjust $R_1$ until the output voltage is 1 V. Replace the discrete resistor with the TD element. The circuit is now set and ready to give 1 V to 5 V output over the chosen temperature range.

Reference/Index

**Figure 1**
**5.0 V Regulated Circuit**

1. LM358 is a general purpose operational amplifier.
2. 2N2222 is a general purpose NPN transistor.
3. Resistor accuracy should be within ±1%.
4. $v_o$ is measured with respect to ground.

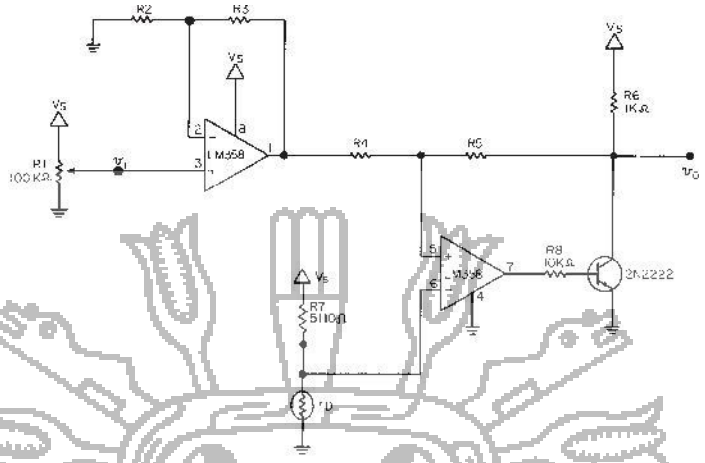



**Figure 2**
**6.5-30 V Supply Voltage**

**Note:** Any error on the 5.0 V regulator will be seen directly on $v_o$. This error can be reduced when setting the span by assuming that V equals the actual output of the regulator.

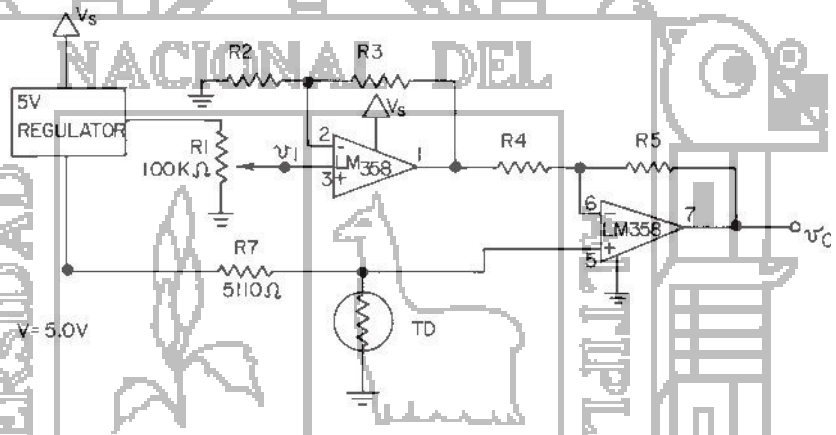
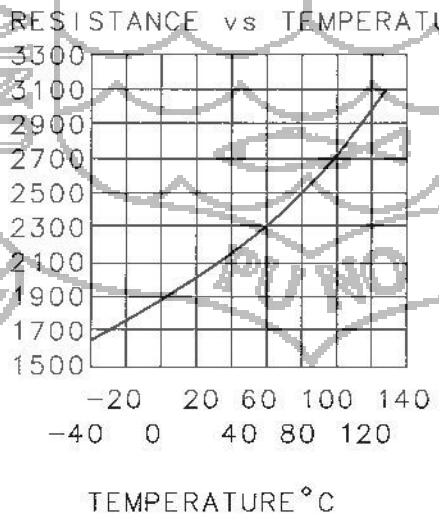



**Figure 3**
**TD Series Resistance vs Temperature**




## ABSOLUTE MAXIMUM RATINGS

| | |
|---|---|
| Operating temperature range | −40 to +150°C (−40 to +302°F) |
| Storage temperature range | −55 to +170°C (−67 to +338°F) |
| Voltage | 10 VDC Continuous (24 hours) |

**Linearity**
±2% (−25 to 85°C)
±3% (−40 to 150°C)
TD sensors can be linearized to within ±0.2%.

**Repeatability**

## ELECTRICAL INTERFACING

The high nominal resistance, positive temperature coefficient and linear sensitivity characteristics of TD Series temperature sensors simplify designing the electrical interface.

Figure 4 is a simple circuit that can be used to linearize the voltage output to within 0.2% or a ±0.4°C error over a range of −40° to +150°C (−40° to +302°F).

Figure 5 illustrates an interface for applications requiring a voltage that varies linearly with temperature. In the example shown, the current regulator sensor resistance can be affected by temperature, so only the temperature sensor should be exposed to thermal changes.

In some applications, it may be desirable to detect one particular temperature. Figure 6 illustrates one way this can be accomplished. In the comparator circuit shown, the potentiometer can be adjusted to correspond to the desired temperature.

**Figure 4**
**Linear Output Voltage Circuit**

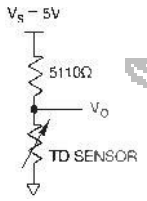



**Table 1 – INTERCHANGEABILITY** (with 1 mA maximum current)

| Temperature | Resistance (Ohms) | Temperature | Resistance (Ohms) |
|---|---|---|---|
| −40°C (−40°F) | 1584 ± 12 (1.9°C) | +60°C (140°F) | 2314 ± 9 (1.1°C) |
| −30°C (−22°F) | 1649 ± 11 (1.7°C) | +70°C (158°F) | 2397 ± 10 (1.2°C) |
| −20°C (−4°F) | 1715 ± 10 (1.5°C) | +80°C (176°F) | 2482 ± 12 (1.4°C) |
| −10°C (14°F) | 1784 ± 9 (1.3°C) | +90°C (194°F) | 2569 ± 14 (1.6°C) |
| 0°C (32°F) | 1854 ± 8 (1.1°C) | +100°C (212°F) | 2658 ± 16 (1.8°C) |
| +10°C (50°F) | 1926 ± 6 (0.8°C) | +110°C (230°F) | 2748 ± 18 (2.0°C) |
| +20°C (68°F) | 2000 ± 5 (0.7°C) | +120°C (248°F) | 2840 ± 19 (2.0°C) |
| +30°C (86°F) | 2076 ± 5 (0.7°C) | +130°C (266°F) | 2934 ± 21 (2.2°C) |
| +40°C (104°F) | 2153 ± 6 (0.8°C) | +140°C (284°F) | 3030 ± 23 (2.4°C) |
| +50°C (122°F) | 2233 ± 7 (0.9°C) | +150°C (302°F) | 3128 ± 25 (2.5°C) |

**Equation for computing resistance:**

$R_T = R_O + (3.84 \times 10^{-3} \times R_O \times T) + (4.94 \times 10^{-6} \times R_O \times T^2)$

$R_T$ = Resistance at temperature T
$R_O$ = Resistance at 0°C
T = Temperature in °C

**Figure 5**
**Simple Current Regulator Interface**

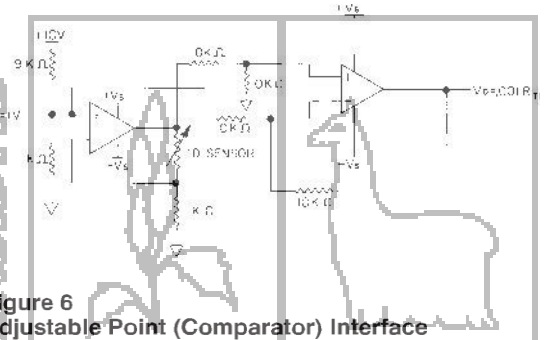
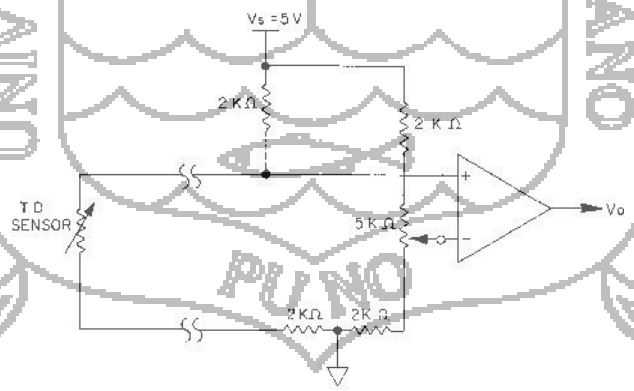



**Figure 6**
**Adjustable Point (Comparator) Interface**




Reference/Index

# 74HC244; 74HCT244

## Octal buffer/line driver; 3-state

Rev. 4 — 24 September 2012 Product data sheet

## 1. General description

The 74HC244; 74HCT244 is an 8-bit buffer/line driver with 3-state outputs. The device can be used as two 4-bit buffers or one 8-bit buffer. The device features two output enables ($1\overline{OE}$ and $2\overline{OE}$), each controlling four of the 3-state outputs. A HIGH on $n\overline{OE}$ causes the outputs to assume a high-impedance OFF-state. Inputs include clamp diodes that enable the use of current limiting resistors to interface inputs to voltages in excess of $V_{CC}$.

## 2. Features and benefits

- Input levels:
  - For 74HC244: CMOS level
  - For 74HCT244: TTL level
- Octal bus interface
- Non-inverting 3-state outputs
- Complies with JEDEC standard no. 7 A
- ESD protection:
  - HBM JESD22-A114F exceeds 2000 V
  - MM JESD22-A115-A exceeds 200 V
- Multiple package options
- Specified from –40 °C to +85 °C and –40 °C to +125 °C

## 3. Ordering information

Table 1. Ordering information

| | Temperature range | | | |
|---|---|---|---|---|
| 74HC244N | | | plastic dual in-line package; 20 leads (300 mil) | SOT146-1 |
| | | | plastic small outline package; 20 leads; body width 7.5 mm | SOT163-1 |
| | –40 °C to +125 °C | SSOP20 | plastic shrink small outline package; 20 leads; body width 5.3 mm | SOT339-1 |
| | | TSSOP20 | plastic thin shrink small outline package; 20 leads; body width 4.4 mm | SOT360-1 |
| | | DHVQFN20 | | SOT764-1 |

## 4. Functional diagram

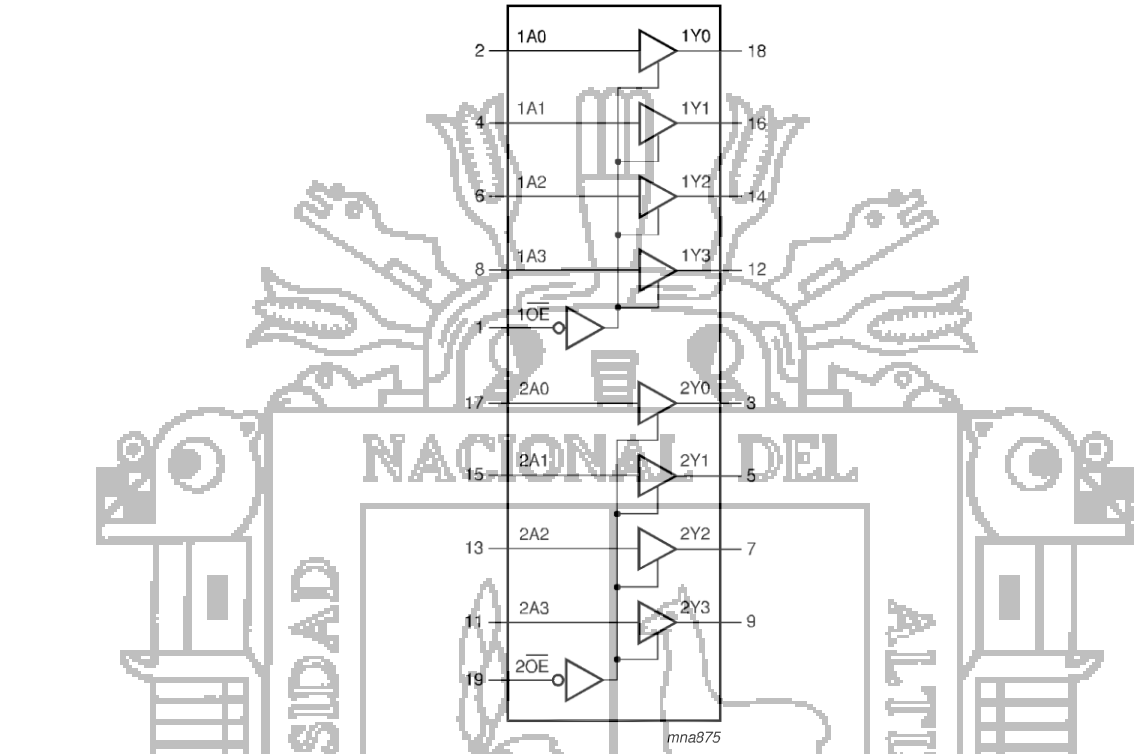



**Fig 1.** Functional diagram

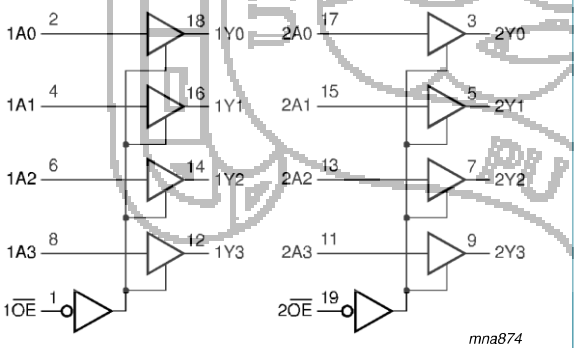



**Fig 2.** Logic symbol

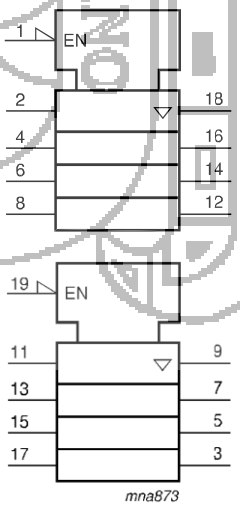



**Fig 3.** IEC logic symbol

# 5. Pinning information

## 5.1 Pinning

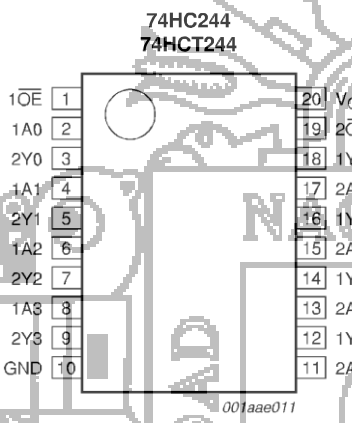
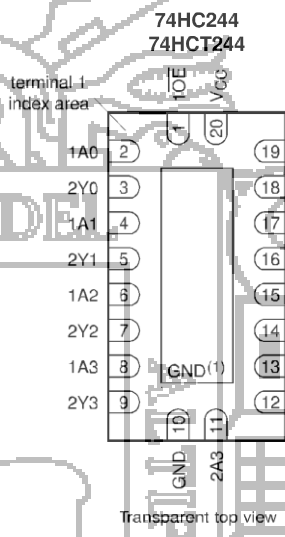



Fig 4. Pin configuration DIP20, SO20, (T)SSOP20




(1) This is not a supply pin. The substrate is attached to this pad using conductive die attach material. There is no electrical or mechanical requirement to solder this pad. However, if it is soldered, the solder land should remain floating or be connected to GND.

Fig 5. Pin configuration DHVQFN20

## 5.2 Pin description

Table 2. Pin description

| Symbol | Pin | Description |
|---|---|---|
| 1$\overline{OE}$, 2$\overline{OE}$ | 1, 19 | output enable input (active LOW) |
| 1A0, 1A1, 1A2, 1A3 | 2, 4, 6, 8 | data input |
| 2A0, 2A1, 2A2, 2A3 | 17, 15, 13, 11 | data input |
| 1Y0, 1Y1, 1Y2, 1Y3 | 18, 16, 14, 12 | bus output |
| 2Y0, 2Y1, 2Y2, 2Y3 | 3, 5, 7, 9 | bus output |
| GND | 10 | ground (0 V) |
| $V_{CC}$ | 20 | supply voltage |

# ANEXO II

**Programa Para los Esclavos**

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//              Esclavo 1 con dirección 10=0x0A                       //
//                                                                    //
//                                                                    //
////////////////////////////////////////////////////////////////////////
#include <16f877A.h>
#device adc=10
//*******************configuracion del controlador******************
#fuses HS, NOWDT, NOPROTECT, PUT, NOBROWNOUT,NOLVP
#use delay(clock=20M)
#use rs232(baud=9600, xmit=PIN_B1, rcv=PIN_B0)
#use standard_io(A)
#use standard_io(B)
#use standard_io(D)
#use standard_io(E)
#byte TRISA = 0x85
#byte TRISB = 0x86
#byte TRISC = 0x87
#byte TRISD = 0x88
#byte TRISE = 0x89
#byte TMR0 = 0x01
#byte STATUS= 0x03
#byte PORTA = 0x05
#byte PORTB = 0x06
#byte PORTC = 0x07
#byte PORTD = 0x08
#byte PORTE = 0x09
#byte INTCON= 0x0B
int8 dato1, dato2, dato3;
int16 velocidad;
//***************configuracio modbus*****************************
#define MODBUS_TYPE MODBUS_TYPE_SLAVE
#define MODBUS_SERIAL_TYPE MODBUS_RTU
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64
#define MODBUS_SERIAL_BAUD 9600
#ifndef USE_WITH_PC
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_EXT
#define MODBUS_SERIAL_TX_PIN PIN_B1
#define MODBUS_SERIAL_RX_PIN PIN_B0
#define MODBUS_SERIAL_ENABLE_PIN   PIN_E0
#else
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
#endif
#include "modbuslib.c"
#define MODBUS_ADDRESS 0x0A
int8 swap_bits(int8 c)
{
return((c&1)?128:0)|((c&2)?64:0)|((c&4)?32:0)|((c&8)?16:0)|((c&16)?8:0)|(
(c&32)?4:0)|((c&64)?2:0)|((c&128)?1:0);}
void read_analogico1(void);
#int_TIMER1
void TIMER1_fsr(void)
{
  velocidad=get_timer0();     //lectura contador TMR0
  dato1=velocidad>>8;         //byte alto
```

```
  dato2=velocidad;              //byte bajo
  set_timer0(0);                //reinicia cuenta
  set_timer1(-59464);           //recarga a 0.5 s 3036
}
void main()
{
int8 coils = 0b00000000;
int8 inputs = 0b00000000;
int16 hold_regs[] = {0x0000,0x0000,0x0000,0x0000,0x0000,
                     0x0000,0x0000,0x0000,0x0000,0x0000};
int16 input_regs[] = {0x0000,0x0000,0x0000,0x0000,0x0000,
                      0x0000,0x0000,0x0000,0x0000,0x0000};
int16 event_count = 0;
   setup_timer_0(rtcc_ext_l_to_h|RTCC_DIV_2);
   setup_timer_1(T1_internal|T1_DIV_BY_8);
   set_timer0(0);                //borrado contado
   set_timer1(-59464);           //carga a 1 s
   enable_interrupts(int_timer1);
   enable_interrupts(global);
   delay_ms(10);
 setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
 setup_adc(ADC_CLOCK_INTERNAL);
 TRISD=0XFF;
 bit_clear(TRISC,0);
 bit_clear(TRISC,1);
 bit_clear(TRISC,2);
 bit_clear(TRISC,3);
 bit_clear(TRISC,4);
 bit_clear(TRISC,5);
 bit_clear(TRISC,6);
 bit_clear(TRISC,7);
 set_tris_e(0x00);
//*****************************************************
 output_high(PIN_E0);
 output_high(PIN_E1);
 PORTC=0XFF;
 delay_ms(50);
 output_low(PIN_E0);
 output_low(PIN_E1);
 PORTC=0X00;
 modbus_init();
while(TRUE)
{
  output_high(PIN_E2);
  output_high(PIN_E1);
if (modbus_rx.func==4)
{
      output_LOW(PIN_E1);
      delay_ms(10);
      read_analogico1();
}
//////////////////////////Registros Modbus/////////////////////////////
      input_regs[0]=make16(dato1,dato2);
      input_regs[1]=make16(00,dato3);
      input_regs[2]=make16(00,00);
      input_regs[3]=make16(00,00);
```

```
      input_regs[4]=make16(00,00);
      input_regs[5]=make16(00,00);
      input_regs[6]=make16(00,00);
      input_regs[7]=make16(00,00);
      input_regs[8]=make16(00,00);
      input_regs[9]=make16(00,00);
////////////////////////////////////////////////////////////////////////////
while(!modbus_kbhit());
   delay_us(50);
if((modbus_rx.address == MODBUS_ADDRESS)|| modbus_rx.address == 0)
{
switch(modbus_rx.func)
{
case FUNC_READ_COILS:
case FUNC_READ_DISCRETE_INPUT:
int a;
        a=PORTD;
        inputs=a;
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 8 ||
modbus_rx.data[3]+modbus_rx.data[1] > 8)
  modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRE);
else
{
int8 data;
if(modbus_rx.func == FUNC_READ_COILS)
                  data = coils>>(modbus_rx.data[1]);
else
                  data = inputs>>(modbus_rx.data[1]);
                  data = data & (0xFF>>(8-modbus_rx.data[3]));
if(modbus_rx.func == FUNC_READ_COILS)
          modbus_read_discrete_input_rsp(MODBUS_ADDRESS, 0x01,&data);
else
          modbus_read_discrete_input_rsp(MODBUS_ADDRESS, 0x01, &data);
          event_count++;

}
break;
case FUNC_READ_HOLDING_REGISTERS:
case FUNC_READ_INPUT_REGISTERS:
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 10 ||
modbus_rx.data[3]+modbus_rx.data[1] > 10) ;

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
if(modbus_rx.func == FUNC_READ_HOLDING_REGISTERS)

modbus_read_holding_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2)
,hold_regs+modbus_rx.data[1]);
else

modbus_read_input_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2)
,input_regs+modbus_rx.data[1]);
          event_count++;
}
break;
case FUNC_WRITE_SINGLE_COIL:
```

```
if(modbus_rx.data[0] || modbus_rx.data[3] || modbus_rx.data[1] > 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else if(modbus_rx.data[2] != 0xFF && modbus_rx.data[2] != 0x00)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_VALUE);
else
{
if(modbus_rx.data[2] == 0xFF)
{
                  bit_set(coils,modbus_rx.data[1]);
                     bit_set(PORTC,modbus_rx.data[1]);
}
else
{
                     bit_clear(coils,modbus_rx.data[1]);
                    bit_clear(PORTC,modbus_rx.data[1]);
}

modbus_write_single_coil_rsp(MODBUS_ADDRESS,modbus_rx.data[1],((int16)(mo
dbus_rx.data[2]))<<8);
                  event_count++;
}
break;
case FUNC_WRITE_SINGLE_REGISTER:
if(modbus_rx.data[0] || modbus_rx.data[1] >= 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
                  hold_regs[modbus_rx.data[1]] =
make16(modbus_rx.data[3],modbus_rx.data[2]);

modbus_write_single_register_rsp(MODBUS_ADDRESS,make16(modbus_rx.data[1,m
odbus_rx.data[0]),make16(modbus_rx.data[3],modbus_rx.data[2]));
//*****************aqu se empieza para la escritura modbus***********
               OUTPUT_BIT(PIN_E2,1);
//*****************fin de la escritura modbus*********************
}
break;
case FUNC_WRITE_MULTIPLE_COILS:
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 8 ||
modbus_rx.data[3]+modbus_rx.data[1] > 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
int i,j;
            modbus_rx.data[5] = swap_bits(modbus_rx.data[5]);
for(i=modbus_rx.data[1],j=0;i < modbus_rx.data[1]+modbus_rx.data[3];
++i,++j)
{
if(bit_test(modbus_rx.data[5],j))
{
                        bit_set(coils,7-i);
                        bit_set(PORTC,7-i);
```

```
}
else
{
                        bit_clear(coils,7-i);
                        bit_clear(PORTC,7-i);
}
}
                    modbus_write_multiple_coils_rsp(MODBUS_ADDRESS,
make16(modbus_rx.data[0],modbus_rx.data[1]),make16(modbus_rx.data[2],modb
us_rx.data[3]));
                    event_count++;
}
break;
case FUNC_WRITE_MULTIPLE_REGISTERS:
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 8 ||
modbus_rx.data[3]+modbus_rx.data[1] > 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
int i,j;
for(i=0,j=5; i < modbus_rx.data[4]/2; ++i,j+=2)
                      hold_regs[i] =
make16(modbus_rx.data[j+1],modbus_rx.data[j]);
                    modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
make16(modbus_rx.data[0],modbus_rx.data[1]),
make16(modbus_rx.data[2],modbus_rx.data[3]));
          event_count++;
}
break;
default:
  modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
}
}
}
}
void read_analogico1(void)
{
   set_adc_channel(0);
   delay_us(20);
   dato3=read_adc();
}
```

```
////////////////////////////////////////////////////////////////////////
//                                                                    //
//              Esclavo 1 con dirección 10=0x0B                       //
//                                                                    //
//                                                                    //
////////////////////////////////////////////////////////////////////////
#include <16f877A.h>
#device adc=10
//*******************configuracion del controlador******************
#fuses HS, NOWDT, NOPROTECT, PUT, NOBROWNOUT,NOLVP
#use delay(clock=20M)
#use rs232(baud=9600, xmit=PIN_B1, rcv=PIN_B0)
#use standard_io(A)
#use standard_io(B)
#use standard_io(D)
#use standard_io(E)
#byte TRISA = 0x85
#byte TRISB = 0x86
#byte TRISC = 0x87
#byte TRISD = 0x88
#byte TRISE = 0x89
#byte TMR0 = 0x01
#byte STATUS= 0x03
#byte PORTA = 0x05
#byte PORTB = 0x06
#byte PORTC = 0x07
#byte PORTD = 0x08
#byte PORTE = 0x09
#byte INTCON= 0x0B
int8 dato1, dato2, dato3;
int16 velocidad;
//***************configuracio modbus*****************************
#define MODBUS_TYPE MODBUS_TYPE_SLAVE
#define MODBUS_SERIAL_TYPE MODBUS_RTU
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64
#define MODBUS_SERIAL_BAUD 9600
#ifndef USE_WITH_PC
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_EXT
#define MODBUS_SERIAL_TX_PIN PIN_B1
#define MODBUS_SERIAL_RX_PIN PIN_B0
#define MODBUS_SERIAL_ENABLE_PIN   PIN_E0
#else
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
#endif
#include "modbuslib.c"
#define MODBUS_ADDRESS 0x0B
int8 swap_bits(int8 c)
{
return((c&1)?128:0)|((c&2)?64:0)|((c&4)?32:0)|((c&8)?16:0)|((c&16)?8:0)|(
(c&32)?4:0)|((c&64)?2:0)|((c&128)?1:0);}
void read_analogico1(void);
void read_analogico2(void);
void main()
{
int8 coils = 0b00000000;
int8 inputs = 0b00000000;
```

```
int16 hold_regs[] = {0x0000,0x0000,0x0000,0x0000,0x0000,
                     0x0000,0x0000,0x0000,0x0000,0x0000};
int16 input_regs[] = {0x0000,0x0000,0x0000,0x0000,0x0000,
                      0x0000,0x0000,0x0000,0x0000,0x0000};
int16 event_count = 0;
setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
 setup_adc(ADC_CLOCK_INTERNAL);
 TRISD=0XFF;
 bit_clear(TRISC,0);
 bit_clear(TRISC,1);
 bit_clear(TRISC,2);
 bit_clear(TRISC,3);
 bit_clear(TRISC,4);
 bit_clear(TRISC,5);
 bit_clear(TRISC,6);
 bit_clear(TRISC,7);
 set_tris_e(0x00);
//*************************************************
 output_high(PIN_E0);
 output_high(PIN_E1);
 PORTC=0XFF;
 delay_ms(50);
 output_low(PIN_E0);
 output_low(PIN_E1);
 PORTC=0X00;
 modbus_init();
while(TRUE)
{
  output_high(PIN_E2);
  output_high(PIN_E1);
if (modbus_rx.func==4)
{
      output_LOW(PIN_E1);
      delay_ms(10);
      read_analogico1();
      delay_ms(10);
      read_analogico2();

}
////////////////////////Registros Modbus//////////////////////////
      input_regs[0]=make16(00,dato1);
      input_regs[1]=make16(00,dato2);
      input_regs[2]=make16(00,00);
      input_regs[3]=make16(00,00);
      input_regs[4]=make16(00,00);
      input_regs[5]=make16(00,00);
      input_regs[6]=make16(00,00);
      input_regs[7]=make16(00,00);
      input_regs[8]=make16(00,00);
      input_regs[9]=make16(00,00);
//////////////////////////////////////////////////////////////////
while(!modbus_kbhit());
   delay_us(50);
if((modbus_rx.address == MODBUS_ADDRESS)|| modbus_rx.address == 0)
{
switch(modbus_rx.func)
```

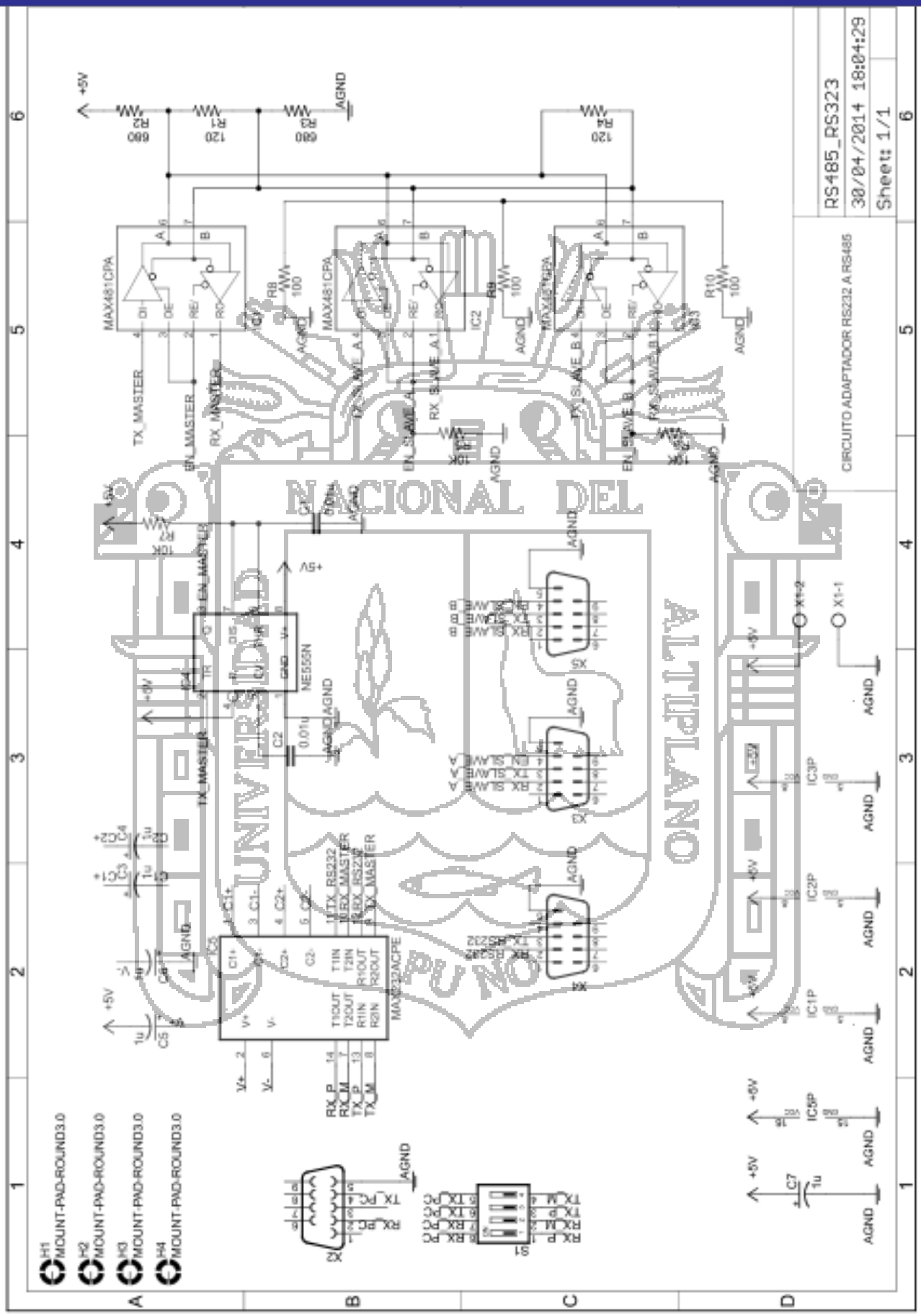
```
{
case FUNC_READ_COILS:
case FUNC_READ_DISCRETE_INPUT:
int a;
        a=PORTD;
        inputs=a;
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 8 ||
modbus_rx.data[3]+modbus_rx.data[1] > 8)
  modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRE);
else
{
int8 data;
if(modbus_rx.func == FUNC_READ_COILS)
                data = coils>>(modbus_rx.data[1]);
else
               data = inputs>>(modbus_rx.data[1]);
               data = data & (0xFF>>(8-modbus_rx.data[3]));
if(modbus_rx.func == FUNC_READ_COILS)
          modbus_read_discrete_input_rsp(MODBUS_ADDRESS, 0x01,&data);
else
          modbus_read_discrete_input_rsp(MODBUS_ADDRESS, 0x01, &data);
          event_count++;
}
break;
case FUNC_READ_HOLDING_REGISTERS:
case FUNC_READ_INPUT_REGISTERS:
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 10 ||
modbus_rx.data[3]+modbus_rx.data[1] > 10) ;

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
if(modbus_rx.func == FUNC_READ_HOLDING_REGISTERS)

modbus_read_holding_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2)
,hold_regs+modbus_rx.data[1]);
else

modbus_read_input_registers_rsp(MODBUS_ADDRESS,(modbus_rx.data[3]*2)
,input_regs+modbus_rx.data[1]);
          event_count++;
}
break;
case FUNC_WRITE_SINGLE_COIL:
if(modbus_rx.data[0] || modbus_rx.data[3] || modbus_rx.data[1] > 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else if(modbus_rx.data[2] != 0xFF && modbus_rx.data[2] != 0x00)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_VALUE);
else
{
if(modbus_rx.data[2] == 0xFF)
{
                  bit_set(coils,modbus_rx.data[1]);
                     bit_set(PORTC,modbus_rx.data[1]);
```

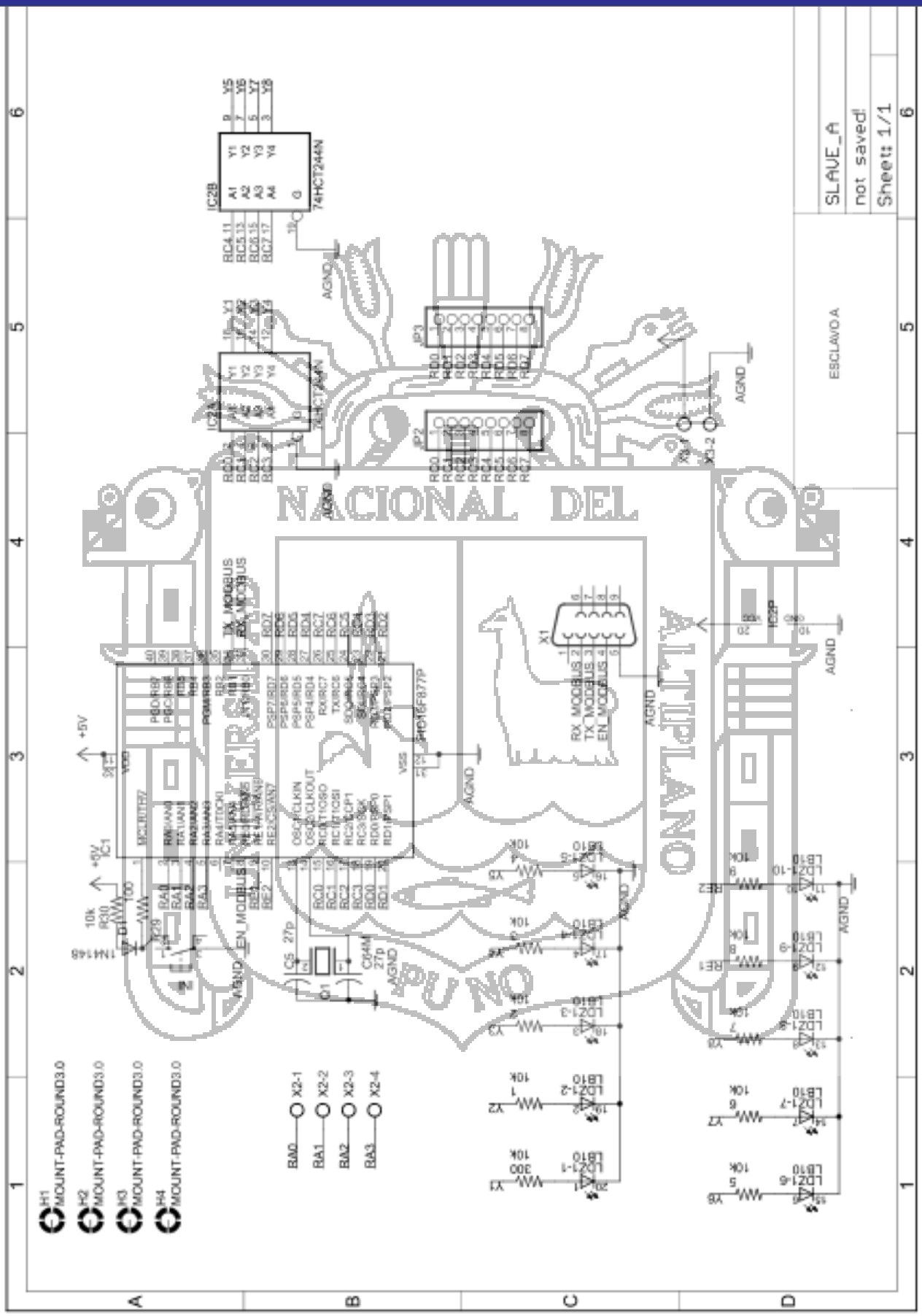
```
}
else
{
                        bit_clear(coils,modbus_rx.data[1]);
                       bit_clear(PORTC,modbus_rx.data[1]);
}

modbus_write_single_coil_rsp(MODBUS_ADDRESS,modbus_rx.data[1],((int16)(mo
dbus_rx.data[2]))<<8);
                    event_count++;
}
break;
case FUNC_WRITE_SINGLE_REGISTER:
if(modbus_rx.data[0] || modbus_rx.data[1] >= 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
                    hold_regs[modbus_rx.data[1]] =
make16(modbus_rx.data[3],modbus_rx.data[2]);

modbus_write_single_register_rsp(MODBUS_ADDRESS,make16(modbus_rx.data[1,m
odbus_rx.data[0]),make16(modbus_rx.data[3],modbus_rx.data[2]));
//*******************aqu se empieza para la escritura modbus***********
                  OUTPUT_BIT(PIN_E2,1);
//**********************fin de la escritura modbus**********************
}
break;
case FUNC_WRITE_MULTIPLE_COILS:
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 8 ||
modbus_rx.data[3]+modbus_rx.data[1] > 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
int i,j;
              modbus_rx.data[5] = swap_bits(modbus_rx.data[5]);
for(i=modbus_rx.data[1],j=0;i < modbus_rx.data[1]+modbus_rx.data[3];
++i,++j)
{
if(bit_test(modbus_rx.data[5],j))
{
                        bit_set(coils,7-i);
                        bit_set(PORTC,7-i);
}
else
{
                        bit_clear(coils,7-i);
                        bit_clear(PORTC,7-i);
}
}
                    modbus_write_multiple_coils_rsp(MODBUS_ADDRESS,
make16(modbus_rx.data[0],modbus_rx.data[1]),make16(modbus_rx.data[2],modb
us_rx.data[3]));
                    event_count++;
}
```

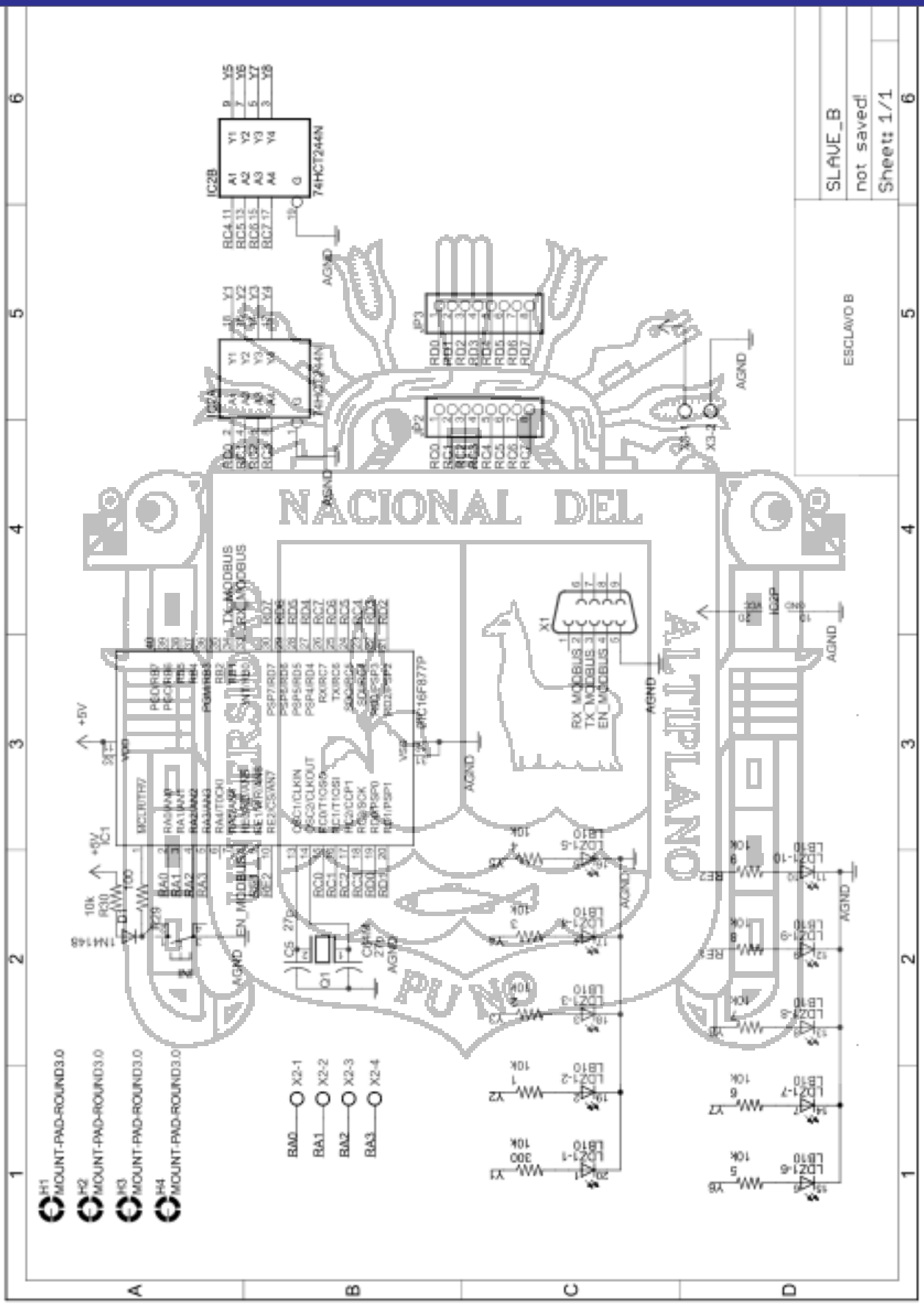
```
break;
case FUNC_WRITE_MULTIPLE_REGISTERS:
if(modbus_rx.data[0] || modbus_rx.data[2] || modbus_rx.data[1] >= 8 ||
modbus_rx.data[3]+modbus_rx.data[1] > 8)

modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
else
{
int i,j;
for(i=0,j=5; i < modbus_rx.data[4]/2; ++i,j+=2)
                    hold_regs[i] =
make16(modbus_rx.data[j+1],modbus_rx.data[j]);
                   modbus_write_multiple_registers_rsp(MODBUS_ADDRESS,
make16(modbus_rx.data[0],modbus_rx.data[1]),
make16(modbus_rx.data[2],modbus_rx.data[3]));
          event_count++;
}
break;
default:
  modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
}
}
}
}
void read_analogico1(void)
{
   set_adc_channel(0);
   delay_us(20);
   dato1=read_adc();
}
 void read_analogico2(void)
{
   set_adc_channel(0);
   delay_us(20);
   dato2=read_adc();
}
```

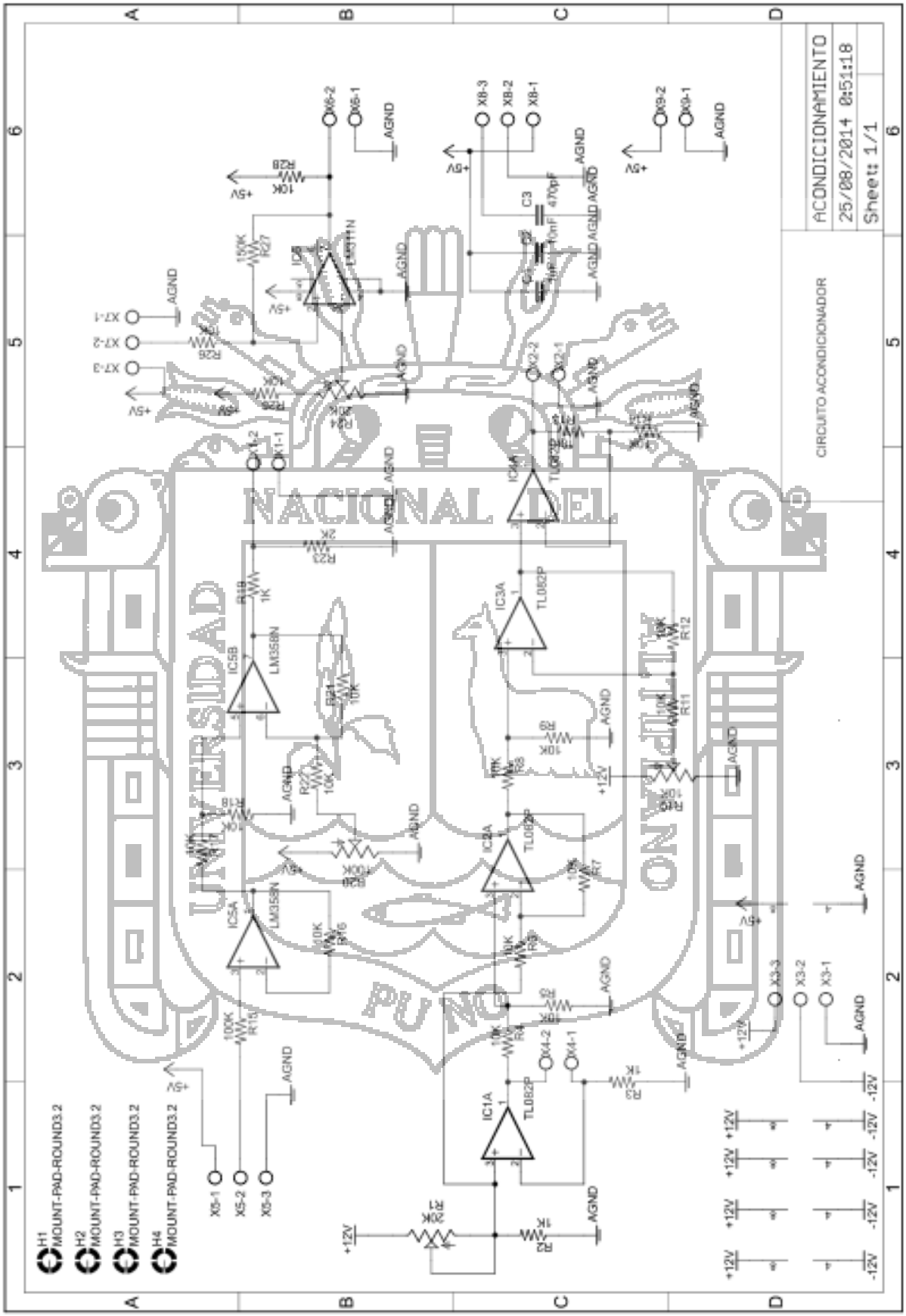
# ANEXO III

**Tarjetas Modbus**




SLAVE_A
not saved!
Sheet: 1/1
ESCLAVO A
IC2B
74HCT244N
IC1
PIC16F877P
X1
RX_MODBUS
TX_MODBUS
EN_MODBUS
AGND
+5V
+5V
IC2P
JP3
JP2
X3-2
X2-1
X2-2
X2-3
X2-4
RA0
RA1
RA2
RA3
H1
MOUNT-PAD-ROUND3.0
H2
MOUNT-PAD-ROUND3.0
H3
MOUNT-PAD-ROUND3.0
H4
MOUNT-PAD-ROUND3.0


SLAVE_B
not saved!
Sheet: 1/1
ESCLAVO B
IC2B
74HCT244N
IC2A
74HCT244N
IC1
PIC16F877P
JP1
JP2
X1
RX_MODBUS
TX_MODBUS
EN_MODBUS
AGND
+5V
MOUNT-PAD-ROUND3.0
H1
H2
H3
H4
X2-1
X2-2
X2-3
X2-4
BA0
BA1
BA2
BA3


ACONDICIONAMIENTO
25/08/2014 8:51:18
Sheet: 1/1
CIRCUITO ACONDICIONADOR
TL082P
LM358N
AGND
+5V
+12V
-12V
MOUNT-PAD-ROUND3.2

# ANEXO IV

**Detección de Errores CRC**

- Detección de errores CRC Función 04 error de dirección

Petición maestro

**Deteccion de errores - CRC**

[0a][04][00][0a][00][0a][51][74]

| N. bytes | Trama Modbus | crc modbus |
|---|---|---|
| 6 | 0A04000A000A | 5174 |

Valor Hex. del polinomio 0xA001 = 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

| byte N. | Hex | Cargar CRC | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | |
|---|---|---|---|---|
| 1 | 0A | 0000000000001010 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 | |
| | | xor 2 tramas de arriba | 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 | |
| | | xor 1 | 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 | |
| | | xor 2 | 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 | |
| | | xor 3 | 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 | |
| | | xor 4 | 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 | |
| | | xor 5 | 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 | |
| | | xor 6 | 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 | |
| | | xor 7 | 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1 0 | |
| | | xor 8 | 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1 | 3F47 |
| 2 | 04 | 0000000000000100 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 | |
| | | xor 2 tramas de arriba | 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 | |
| | | xor 1 | 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 | |
| | | xor 2 | 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 | |
| | | xor 3 | 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 | |
| | | xor 4 | 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 | |
| | | xor 5 | 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 | |
| | | xor 6 | 1 0 0 0 1 1 0 0 0 0 0 1 1 1 0 1 | |
| | | xor 7 | 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 | |
| | | xor 8 | 1 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 | 06D3 |
| 3 | 00 | 0000000000000000 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | |
| | | xor 2 tramas de arriba | 1 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 | |
| | | xor 1 | 0 1 1 0 1 0 0 1 1 0 0 0 0 0 1 1 | |
| | | xor 2 | 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 | |
| | | xor 3 | 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 | |
| | | xor 4 | 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 | |
| | | xor 5 | 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 | |
| | | xor 6 | 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 | |
| | | xor 7 | 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 | |
| | | xor 8 | 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 | 5302 |
| 4 | 0A | 0000000000001010 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 | |
| | | xor 2 tramas de arriba | 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 | |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 2 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | |
| | | xor 3 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | |
| | | xor 4 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 8 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | **C23A** |
| 5 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 8 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | **BB91** |
| 6 | **0A** | 0000000000001010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | | |
| | | xor 2 tramas de arriba | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | **5174** |

Respuesta esclavo

**Detección de errores - CRC**

[0a][84][02][b3][03]

| **N. bytes** | Trama Modbus | **crc modbus** |
|---|---|---|
| **3** | **0a8402** | **B303** |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Valor Hex. del polinomio 0xA001 = | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |

| byte N. | Hex | Cargar CRC | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **0a** | 0000000000001010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | **3F47** |
| 2 | **84** | 0000000010000100 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| | | xor 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | | xor 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| | | xor 4 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| | | xor 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| | | xor 6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | xor 8 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | **0773** |
| 3 | **02** | 0000000000000010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** |
| | | xor 2 tramas de arriba | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | | xor 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | xor 2 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | xor 3 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | xor 4 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | xor 5 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | | xor 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| | | xor 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | | xor 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | **B303** |

- Detección de errores CRC Función 04 error de datos

Petición maestro

**Deteccion de errores - CRC**

[0a][04][00][00][00][14][f1][7e]

| **N. bytes** | Trama Modbus | **crc modbus** |
|---|---|---|
| **6** | **0A0400000014** | **F17E** |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Valor Hex. del polinomio 0xA001 = | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |

| byte N. | Hex | Cargar CRC | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **0A** | 0000000000001010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | **3F47** |
| 2 | **04** | 0000000000000100 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 4 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | **06D3** |
| 3 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| | | xor 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | | xor 3 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | |
| | | xor 5 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | |
| | | xor 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | |
| | | xor 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | **5302** |
| 4 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 8 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | **423D** |
| 5 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | |
| | | xor 7 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | **BD31** |
| 6 | **14** | 0000000000010100 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | |
| | | xor 2 tramas de arriba | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 2 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |
| | | xor 4 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | **F17E** |

Respuesta esclavo

**Deteccion de errores - CRC**

| **N. bytes** | Trama Modbus | **crc modbus** |
|---|---|---|
| **3** | **0A8402** | **B303** |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Valor Hex. del polinomio 0xA001 = | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |

| byte N. | Hex | Cargar CRC | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **0A** | 0000000000001010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | **3F47** |
| 2 | **84** | 0000000010000100 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **0** |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| | | xor 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | | xor 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| | | xor 4 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| | | xor 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| | | xor 6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | xor 8 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | **0773** |
| 3 | **02** | 0000000000000010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** |
| | | xor 2 tramas de arriba | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | | xor 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | xor 2 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 3 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | |
| | | xor 5 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | |
| | | xor 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | |
| | | xor 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | **B303** |

- Detección de errores CRC Función 02

Petición maestro

**Deteccion de errores - CRC**

[0a][02][00][00][00][08][78][b7]

| **N. bytes** | Trama Modbus | **crc modbus** |
|---|---|---|
| **6** | **0A0200000008** | **78B7** |

Valor Hex. del polinomio 0xA001 =

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |

| byte N. | Hex | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Cargar CRC | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | |
| 1 | **0A** | 0000000000001010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | | |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | | |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | |
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | **3F47** | |
| 2 | **02** | 0000000000000010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | **86D1** |
| 3 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| | | xor 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | xor 3 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | xor 4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| | | xor 5 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| | | xor 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| | | xor 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| | | xor 8 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | | | | **50A2** |
| 4 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| | | xor 2 tramas de arriba | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| | | xor 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| | | xor 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| | | xor 3 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | xor 4 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| | | xor 5 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | | xor 6 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | | xor 7 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | xor 8 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | | | | | | | | | | | | | | | | | | **A23C** |
| 5 | **00** | 0000000000000000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| | | xor 2 tramas de arriba | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| | | xor 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| | | xor 2 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| | | xor 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | xor 4 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| | | xor 5 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| | | xor 6 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| | | xor 7 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| | | xor 8 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| | | | | | | | | | | | | | | | | | | **BDB9** |
| 6 | **08** | 0000000000001000 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **0** |
| | | xor 2 tramas de arriba | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| | | xor 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| | | xor 2 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| | | xor 3 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | | xor 7 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | xor 8 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | | | | | | | | | | | | | | | | | **78B7** |

Respuesta esclavo

**Detección de errores - CRC**

[0a][02][01][c3][e3][fd]

| **N. bytes** | Trama Modbus | **crc modbus** |
|---|---|---|
| **4** | **0A0201C3** | **E3FD** |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Valor Hex. del polinomio 0xA001 = | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| byte N. | Hex | Cargar CRC | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0A | 0000000000001010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | | xor 2 tramas de arriba | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 3F47 |
| 2 | 02 | 0000000000000010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 86D1 |
| 3 | 01 | 0000000000000001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 2 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 9162 |
| 4 | C3 | 0000000011000011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | |
| | | xor 2 tramas de arriba | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 2 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 3 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 4 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 5 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 6 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 7 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | |
| | | xor 8 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | E3FD |

- Detección de errores CRC Función 05

Petición maestro

**Detección de errores - CRC**

[0a][01][00][00][00][08][3c][b7]

| N. bytes | Trama Modbus | crc modbus |
|---|---|---|
| 6 | 0A0100000008 | 3CB7 |

Valor Hex. del polinomio 0xA001 = 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

| byte N. | Hex | Cargar CRC | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | |
|---|---|---|---|---|
| 1 | 0A | 0000000000001010 | 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 | |
| | | xor 2 tramas de arriba | 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 | |
| | | xor 1 | 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 | |
| | | xor 2 | 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 | |
| | | xor 3 | 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 | |
| | | xor 4 | 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 | |
| | | xor 5 | 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 | |
| | | xor 6 | 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 | |
| | | xor 7 | 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1 0 | |
| | | xor 8 | 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1 | 3F47 |
| 2 | 01 | 0000000000000001 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | |
| | | xor 2 tramas de arriba | 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 | |
| | | xor 1 | 0 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 | |
| | | xor 2 | 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 | |
| | | xor 3 | 0 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 | |
| | | xor 4 | 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 | |
| | | xor 5 | 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 | |
| | | xor 6 | 1 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 | |
| | | xor 7 | 1 1 1 0 0 0 0 1 1 0 0 0 1 1 1 1 | |
| | | xor 8 | 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 | C6D0 |
| 3 | 00 | 0000000000000000 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | |
| | | xor 2 tramas de arriba | 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 | |
| | | xor 1 | 0 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 | |
| | | xor 2 | 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 | |
| | | xor 3 | 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 | |
| | | xor 4 | 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0 | |
| | | xor 5 | 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 | |
| | | xor 6 | 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 | |
| | | xor 7 | 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 | |
| | | xor 8 | 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 | 5052 |
| 4 | 00 | 0000000000000000 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | |
| | | xor 2 tramas de arriba | 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 | |

| | | | | |
|---|---|---|---|---|
| | | xor 1 | 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 | |
| | | xor 2 | 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 | |
| | | xor 3 | 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 | |
| | | xor 4 | 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 | |
| | | xor 5 | 1 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 | |
| | | xor 6 | 1 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 | |
| | | xor 7 | 0 1 1 1 1 0 0 0 1 0 1 0 0 1 0 0 | |
| | | xor 8 | 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 0 | 523C |
| 5 | 00 | 0000000000000000 | **0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0** | |
| | | xor 2 tramas de arriba | 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 0 | |
| | | xor 1 | 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 | |
| | | xor 2 | 1 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 | |
| | | xor 3 | 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 1 | |
| | | xor 4 | 1 1 0 1 1 0 1 1 1 1 0 0 0 1 0 0 | |
| | | xor 5 | 0 1 1 0 1 1 0 1 1 1 1 0 0 0 1 0 | |
| | | xor 6 | 0 0 1 1 0 1 1 0 1 1 1 1 0 0 0 1 | |
| | | xor 7 | 1 0 1 1 1 0 1 1 0 1 1 1 1 0 0 1 | |
| | | xor 8 | 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 | BDFD |
| 6 | 08 | 0000000000001000 | **0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0** | |
| | | xor 2 tramas de arriba | 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 | |
| | | xor 1 | 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 | |
| | | xor 2 | 1 1 0 0 1 1 1 1 0 1 1 0 1 1 0 0 | |
| | | xor 3 | 0 1 1 0 0 1 1 1 1 0 1 1 0 1 1 0 | |
| | | xor 4 | 0 0 1 1 0 0 1 1 1 1 0 1 1 0 1 1 | |
| | | xor 5 | 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 0 | |
| | | xor 6 | 0 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 | |
| | | xor 7 | 0 0 1 0 1 1 1 0 0 1 1 1 1 0 1 1 | |
| | | xor 8 | 1 0 1 1 0 1 1 1 0 0 1 1 1 1 0 0 | 3CB7 |

Respuesta esclavo

**Detección de errores - CRC**

[0a][02][01][01][62][66]

| N. bytes | Trama Modbus | crc modbus |
|---|---|---|
| 4 | 0A020101 | 626C |

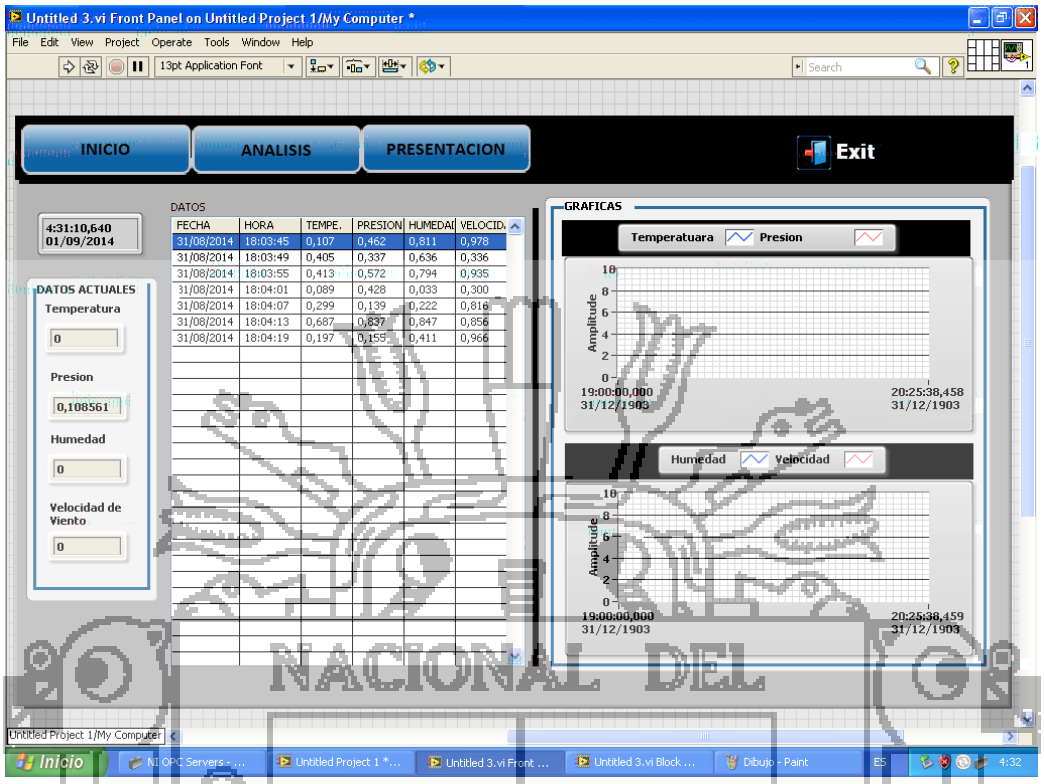
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Valor Hex. del polinomio 0xA001 = | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

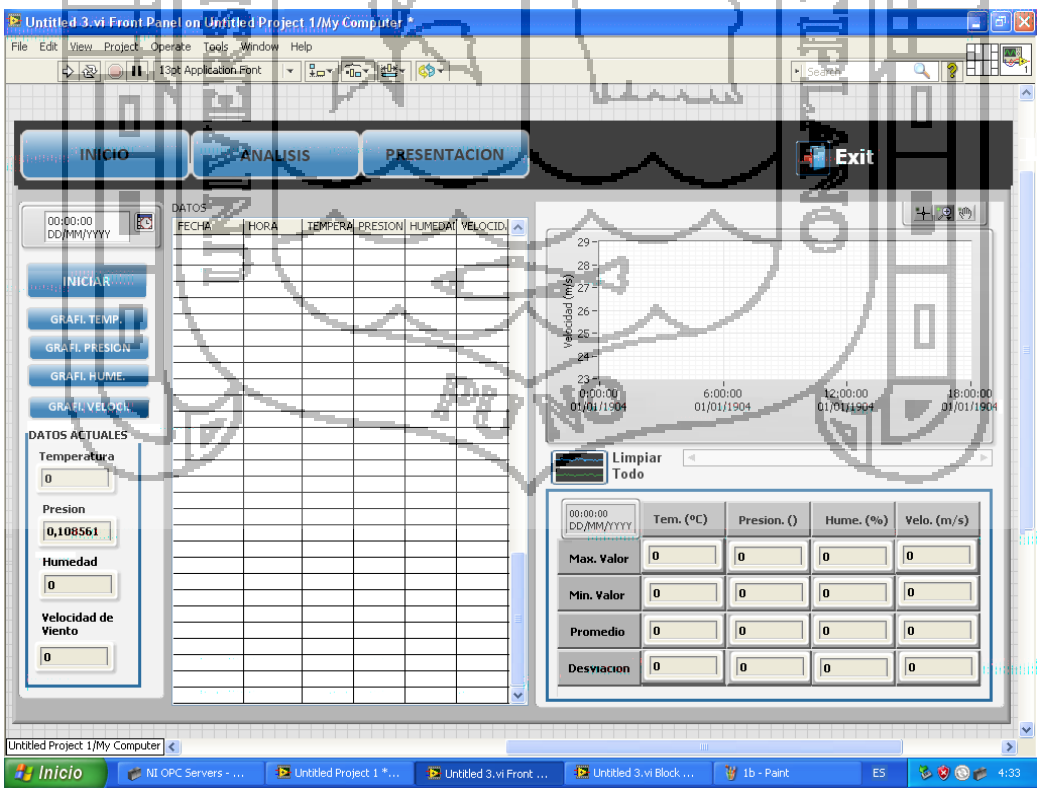
| byte N. | Hex | Cargar CRC | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
|---|---|---|---|
| 1 | 0A | 0000000000001010 | **0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0** |
| | | xor 2 tramas de arriba | 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 |
| | | xor 1 | 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 |
| | | xor 2 | 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 |

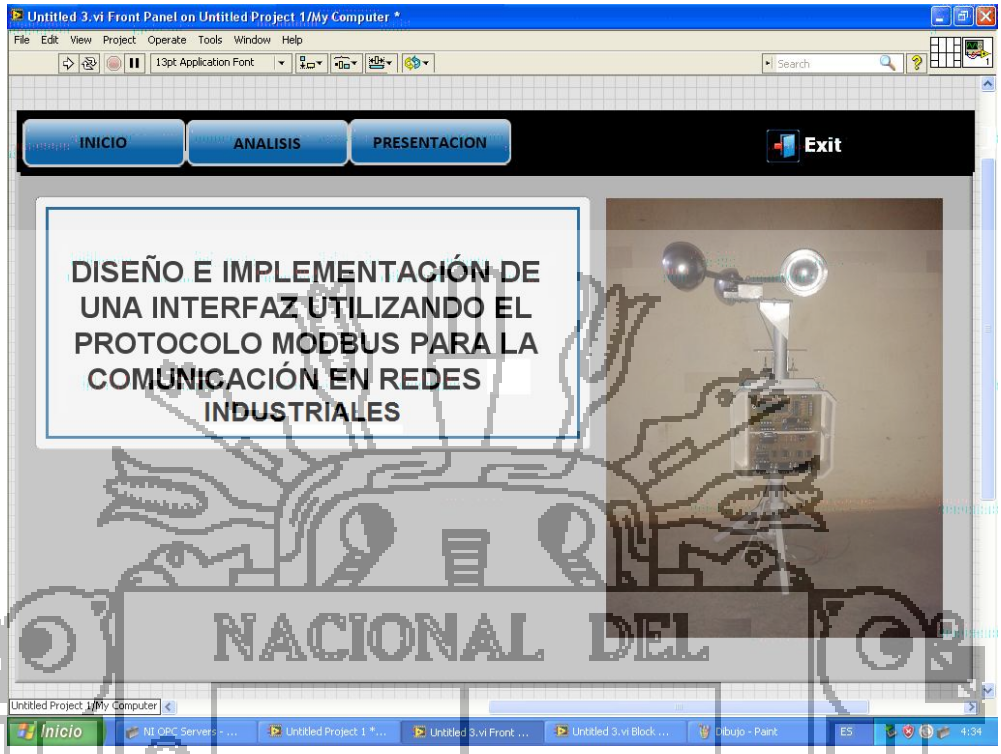
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | xor 3 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 6 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | **3F47** |
| 2 | **02** | 0000000000000010 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | |
| | | xor 2 tramas de arriba | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | |
| | | xor 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | |
| | | xor 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 5 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 6 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | |
| | | xor 7 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| | | xor 8 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | **86D1** |
| 3 | **01** | 0000000000000001 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | |
| | | xor 2 tramas de arriba | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 2 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | | xor 3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | |
| | | xor 4 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | |
| | | xor 5 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | |
| | | xor 6 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| | | xor 7 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| | | xor 8 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | **9162** |
| 4 | **01** | 0000000000000001 | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | |
| | | xor 2 tramas de arriba | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| | | xor 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | |
| | | xor 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | |
| | | xor 4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | |
| | | xor 5 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | |
| | | xor 6 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | |
| | | xor 7 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | | xor 8 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | **626C** |

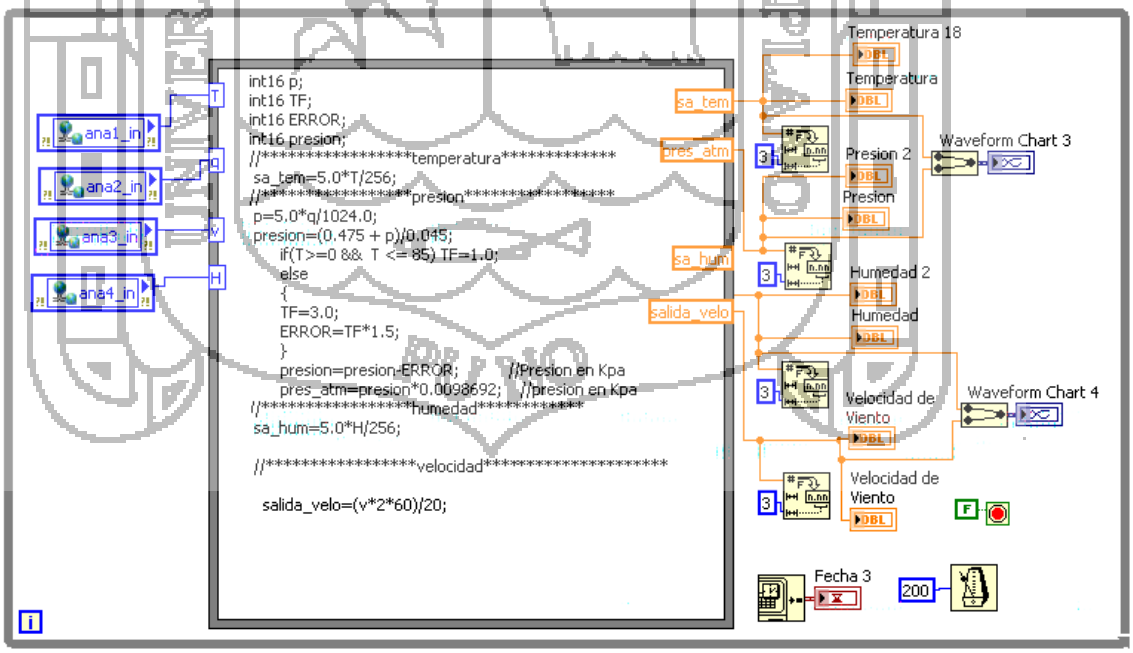
# ANEXO V

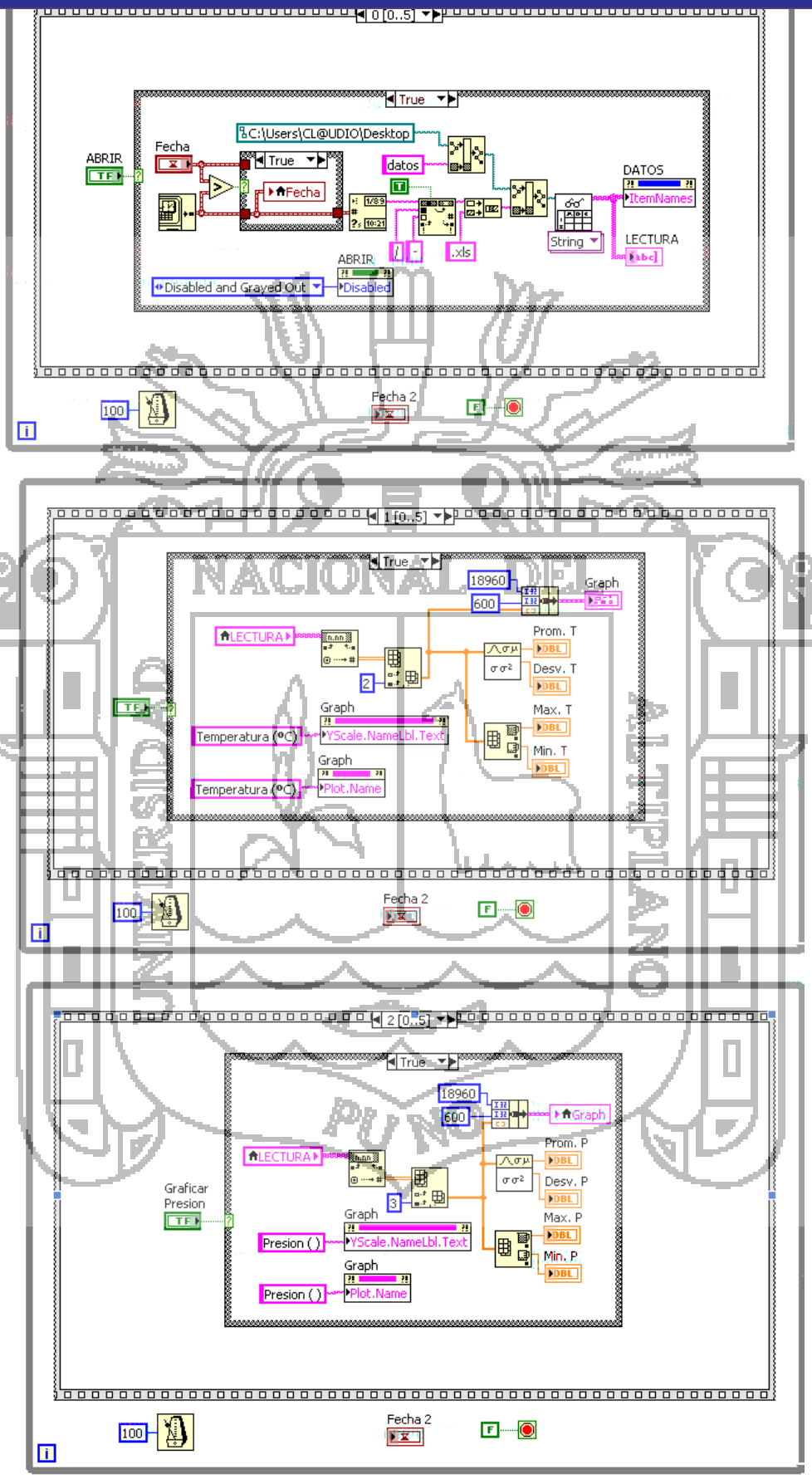
**Programa Aplicación**

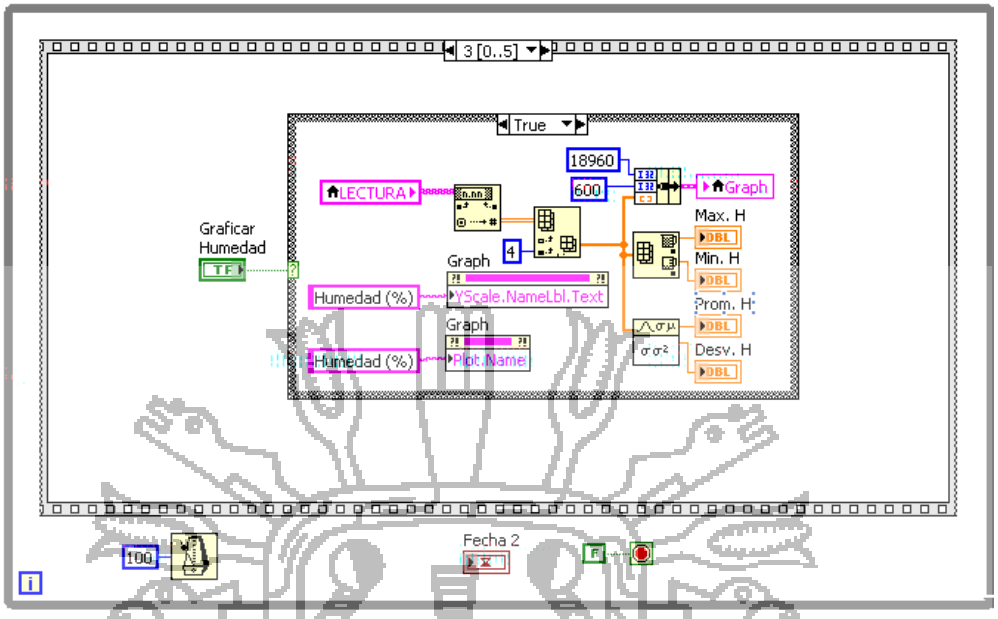

Untitled 3.vi Front Panel on Untitled Project 1/My Computer *
File Edit View Project Operate Tools Window Help
INICIO
ANALISIS
PRESENTACION
Exit
4:31:10,640
01/09/2014
DATOS
FECHA HORA TEMPE. PRESION HUMEDAD VELOCID.
31/08/2014 18:03:45 0,107 0,462 0,811 0,978
31/08/2014 18:03:49 0,405 0,337 0,636 0,336
31/08/2014 18:03:55 0,413 0,572 0,794 0,935
31/08/2014 18:04:01 0,089 0,428 0,033 0,300
31/08/2014 18:04:07 0,299 0,139 0,222 0,816
31/08/2014 18:04:13 0,687 0,837 0,847 0,856
31/08/2014 18:04:19 0,197 0,155 0,411 0,966
DATOS ACTUALES
Temperatura
0
Presion
0,108561
Humedad
0
Velocidad de Viento
0
GRAFICAS
Temperatuara
Presion
Amplitude
19:00:00,000 31/12/1903
20:25:38,458 31/12/1903
Humedad
Velocidad
Amplitude
19:00:00,000 31/12/1903
20:25:38,459 31/12/1903
Untitled Project 1/My Computer
Inicio
NI OPC Servers - ...
Untitled Project 1 *...
Untitled 3.vi Front ...
Untitled 3.vi Block ...
Dibujo - Paint
ES
4:32

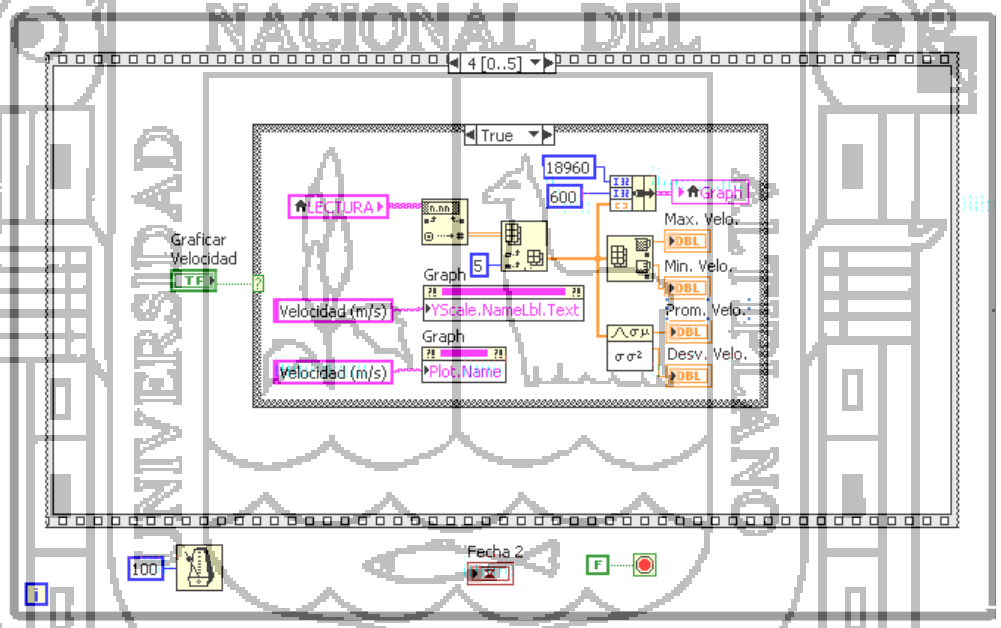

Untitled 3.vi Front Panel on Untitled Project 1/My Computer *
File Edit View Project Operate Tools Window Help
INICIO
ANALISIS
PRESENTACION
Exit
00:00:00
DD/MM/YYYY
INICIAR
GRAFI. TEMP.
GRAFI. PRESION
GRAFI. HUME.
GRAFI. VELOCI.
DATOS
FECHA HORA TEMPERA PRESION HUMEDAD VELOCID.
DATOS ACTUALES
Temperatura
0
Presion
0,108561
Humedad
0
Velocidad de Viento
0
Velocidad (m/s)
0:00:00 01/01/1904
6:00:00 01/01/1904
12:00:00 01/01/1904
18:00:00 01/01/1904
Limpiar Todo
00:00:00
DD/MM/YYYY
Tem. (ºC)
Presion. ()
Hume. (%)
Velo. (m/s)
Max. Valor
Min. Valor
Promedio
Desviacion
Untitled Project 1/My Computer
Inicio
NI OPC Servers - ...
Untitled Project 1 *...
Untitled 3.vi Front ...
Untitled 3.vi Block ...
1b - Paint
ES
4:33

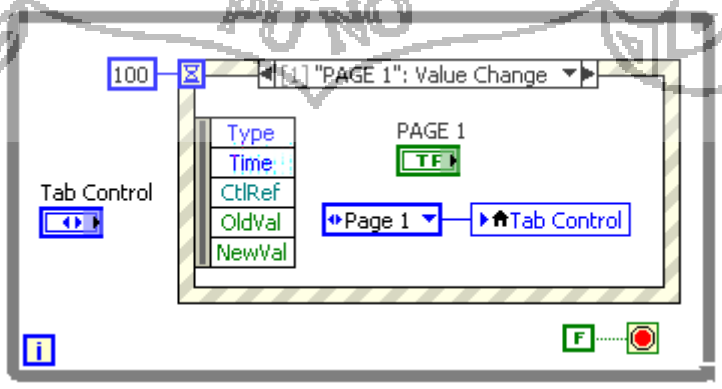

Untitled 3.vi Front Panel on Untitled Project 1/My Computer *
File Edit View Project Operate Tools Window Help
13pt Application Font
Search
INICIO
ANALISIS
PRESENTACION
Exit
DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ UTILIZANDO EL PROTOCOLO MODBUS PARA LA COMUNICACIÓN EN REDES INDUSTRIALES
Untitled Project 1/My Computer
Inicio
NI OPC Servers - ...
Untitled Project 1 *...
Untitled 3.vi Front ...
Untitled 3.vi Block ...
Dibujo - Paint
ES
4:34


ana1_in
ana2_in
ana3_in
ana4_in
T
q
V
H
int16 p;
int16 TF;
int16 ERROR;
int16 presion;
//**********temperatura**********
sa_tem=5.0*T/256;
//**********presion**********
p=5.0*q/1024.0;
presion=(0.475 + p)/0.045;
if(T>=0 && T <= 85) TF=1.0;
else
{
TF=3.0;
ERROR=TF*1.5;
}
presion=presion-ERROR; //Presion en Kpa
pres_atm=presion*0.0098692; //presion en Kpa
//**********humedad**********
sa_hum=5.0*H/256;
//**********velocidad**********
salida_velo=(v*2*60)/20;
sa_tem
pres_atm
sa_hum
salida_velo
Temperatura 18
Temperatura
Presion 2
Presion
Waveform Chart 3
Humedad 2
Humedad
Velocidad de Viento
Waveform Chart 4
Velocidad de Viento
Fecha 3
200


0 [0..5]
True
C:\Users\CL@UDIO\Desktop
Fecha
ABRIR
True
datos
Fecha
DATOS
ItemNames
String
LECTURA
.xls
ABRIR
Disabled and Grayed Out
Disabled
100
Fecha 2
1 [0..5]
True
18960
600
Graph
LECTURA
Prom. T
Desv. T
2
Graph
Max. T
Temperatura (°C)
YScale.NameLbl.Text
Min. T
Graph
Temperatura (°C)
Plot.Name
100
Fecha 2
2 [0..5]
True
18960
600
Graph
LECTURA
Prom. P
Desv. P
Graficar Presion
3
Graph
Max. P
Presion ( )
YScale.NameLbl.Text
Min. P
Graph
Presion ( )
Plot.Name
100
Fecha 2


3 [0..5]
True
LECTURA
18960
600
Graph
Max. H
Min. H
Prom. H
Desv. H
Graficar Humedad
4
Graph
Humedad (%)
YScale.NameLbl.Text
Graph
Humedad (%)
Plot.Name
100
Fecha 2


4 [0..5]
True
LECTURA
18960
600
Graph
Max. Velo.
Min. Velo.
Prom. Velo.
Desv. Velo.
Graficar Velocidad
5
Graph
Velocidad (m/s)
YScale.NameLbl.Text
Graph
Velocidad (m/s)
Plot.Name
100
Fecha 2


100
[1] "PAGE 1": Value Change
Type
Time
CtlRef
OldVal
NewVal
PAGE 1
Tab Control
Page 1
Tab Control