UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO
FACULTAD DE INGENIERIAS
CARRERA DE INGENIERÍA ELÉCTRICA

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO EN CONTROL INDUSTRIAL

TEMA:

"Diseño y construcción de un módulo de adquisición de datos para la supervisión y control de una mini planta de procesos con interfaz USB para LabView"

AUTORES:

MANOSALVAS BENALCÁZAR ÁNGEL EFRAÍN
QUINGA JARRIN JAVIER ALEJANDRO

DIRIGIDA POR:

ING. RAMIRO ROBAYO.

QUITO, JUNIO DEL 2012

DECLARACIÓN DE AUTORÍA

Nosotros, Angel Efraín Manosalvas Benalcázar y Javier Alejandro Quinga Jarrín alumnos de la Universidad Politécnica Salesiana, Facultad de Ingenierías, carrera de Ingeniería Eléctrica, libre y voluntariamente declaramos que la presente tesis a sido realizada en su totalidad por nosotros, por tal razón asumimos la responsabilidad de la autoría.

Quito, 24 de Mayo del 2012

---------- ----------

Angel Efraín Manosalvas Benalcázar Javier Alejandro Quinga Jarrín.

Certifico que el presente trabajo de tesis ha sido realizado en forma total por los señores Manosalvas Benalcázar Angel Efraín y Quinga Jarrin Javier Alejandro.

_____

Ing. Ramiro Robayo.

DIRECTOR DE TESIS

AGRADECIMIENTOS

Agradezco a mi familia por al apoyo incondicional que ha sabido brindarme en todo este proceso, a Javier Q. por la ayuda y comprensión en momentos complicados y de discordancia en temas que se tratan en este documento y agradezco también a nuestro tutor de tesis que ha sabido guiarnos de la mejor manera en momentos de incertidumbre.

Angel Manosalvas Benalcázar.

Este proyecto es el resultado del esfuerzo conjunto que hemos realizado con Ángel Manosalvas con quien hemos formado un muy buen grupo trabajo, por eso recibe mis agradecimientos ya que fue muy importante el trabajo en equipo, de manera especial agradezco a mis padres que han sabido apoyarme en todo sentido para que pueda lograr mis objetivos, finalmente agradezco a la oportunidad que brinda la electrónica de poder crear cosas y a la creatividad que cada persona tiene y nosotros hemos sabido aprovechar.
También te agradezco a tí chiquitita.

Javier Quinga Jarrín.

PLANTEAMIENTO DEL PROBLEMA

En el laboratorio de Microbotica de la Universidad Politécnica Salesiana el software de programación grafica LabView es muy utilizado por los estudiantes de Ingeniería Eléctrica como herramienta para la adquisición de datos, en el mercado existen módulos de adquisición de datos, pero su elevado costo hace difícil que el estudiante tenga acceso a una de dichas tarjetas, tomando en cuenta que para los estudiante sería de mucha utilidad tener una de estas tarjetas para el buen aprendizaje de LabView enfocado al control industrial.

Este antecedente nos ha animado a desarrollar una mini planta de procesos industriales que nos facilite el trabajo en el laboratorio y que además posea todas la herramientas necesarias que le permitan al estudiante realizar simulaciones de procesos industriales y que su trabajo en el laboratorio sea óptimo con prestaciones similares a las que nos da una de estas tarjetas y facilite realizar un trabajo con sensores industriales.

JUSTIFICACIÓN

Por el elevado costo de las tarjetas de adquisición de datos que existen en el mercado y por la rigidez que estas presentan en su hardware se ve la necesidad de explorar otras alternativas más económicas, útiles y funcionales para el estudiante de ingeniería, una de estas alternativas es la construcción de una mini planta didáctica de procesos industriales la cual tendrá características básicas de un proceso industrial real y permitirá la adquisición de datos digitales, analógicos y salidas del mismo tipo en tiempo real.

ALCANCE

Actualmente en el mercado existen varias opciones al momento de elegir que hardware o software a utilizar para automatizar un proceso, en este ámbito el uso de Controladores Lógicos Programables (PLCs) y sistemas para la Supervisión, Control y Adquisición de Datos (SCADAs), es muy generalizado, debido a su simplicidad de manejo y su capacidad de expansión, sin embargo muchas soluciones resultan muy costosas para nuestro medio, por lo que es necesario buscar otras alternativas, que aunque involucran más tiempo de desarrollo permiten ofrecer una solución satisfactoria a numerosos problemas de control e instrumentación.

Una de esas alternativas es la utilización de tarjetas o módulos para la adquisición de datos y el control de variables de un proceso, que en la actualidad son cada vez más utilizadas debido a que ofrecen soluciones a la medida de los requerimientos de la aplicación. Comercialmente existen muchas tarjetas con diferentes características aunque también cabe la posibilidad de desarrollarlas como es el caso del módulo propuesto.

En esta sección describiremos las características técnicas del modulo llamado mini planta de procesos, el mismo que constará de:

- Un motor de corriente continua, tendrá inversión de giro, control de velocidad y para su monitoreo se utilizará un encoder.
- Un motor de pasos, tendrá su circuito para control y un encoder también para poder monitorearlo.
- Un servomotor, para poder realizar control de posición.
- Una luminaria tipo dicroico, la cual poseerá control de iluminación ON/OFF y proporcional, este a su vez servirá para calentar los diferentes tipos de sensores de temperatura que podrán conectarse al modulo.

- Dentro de los sensores de temperatura se utilizara circuitos de acondicionamiento de señales para termopares tipo K, y sensores resistivos (RTD) PT-100 a tres y cuatro hilos, y sensores de temperatura lineales como el LM35.
- Un circuito de acondicionamiento de señal para un sensor de presión tipo celda de carga.
- 4 salidas tipo transistor.

El modulo tendrá la capacidad de expandirse, es decir tendrá la capacidad y el espacio para colocar tarjetas adicionales descritas a continuación:

- Tarjeta de entradas digitales, esta tarjeta será de 8 entradas digitales optoacopladas de 10 - 24 voltios.
- Una tarjeta de 8 salidas de relé.
- Tarjeta de señales analógicas:
    - Entradas analógicas: lazo de corriente 4 - 20mA, voltaje de 0 - 5v y 0 - 10v .
    - Salidas analógicas: lazo de corriente 4 - 20mA, voltaje de 0 - 5v y 0 - 10v.

Todo lo mencionado anteriormente estará conectado a la tarjeta maestra que se encargara de la gestión de todos los datos obtenidos y a su vez será la interfaz con VISA de LabView vía USB, el módulo tendrá su alimentación propia de 110VAC.

La tarjeta tendrá además comunicación mediante el protocolo MODBUS – RTU el cual será habilitado mediante jumpers.

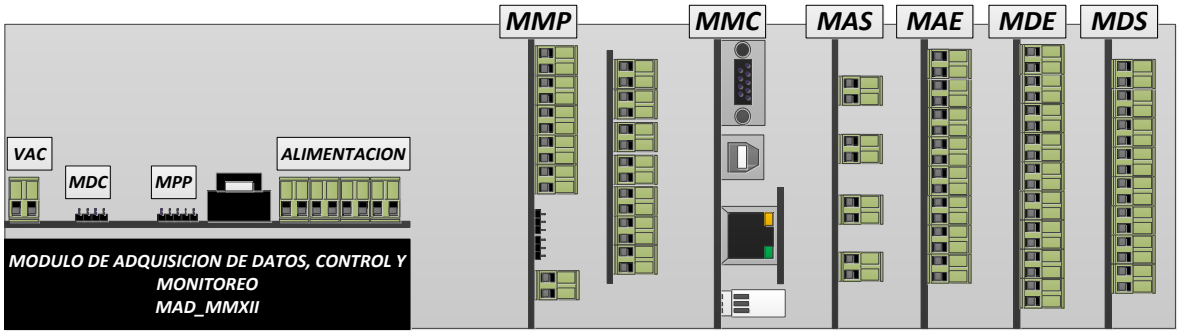
OBJETIVO GENERAL.

- Diseñar y construir de un módulo de adquisición de datos para la supervisión y control de una mini planta de procesos con interfaz USB para LabView 2010.

OBJETIVOS ESPECÍFICOS

- Ejecutar trabajos de investigación acerca de los elementos que intervienen en un sistema de control industrial, características de las señales típicas de control, tipos de microcontroladores existentes en el mercado, circuitos integrados especiales, protocolos de comunicación que se tomarán en cuenta para el diseño y la construcción del módulo de adquisición de datos.
- Diseñar y construir las tarjetas electrónicas que servirán para hacer la interface con LabView, el acondicionamiento de señales y demás funciones de control, ayudándonos de micro controladores de Microchip.
- Establecer la comunicación USB (Universal Serial Bus) entre el microcontrolador PIC18f2550 y un computador utilizando la herramienta VISA de Labview, valiéndonos de las funciones que nos ofrece el micro controlador PIC18F2550 programado en el software PCWHD Compiler.
- Realizar el protocolo de pruebas respectivo del módulo de adquisición de datos una vez acabada la construcción del mismo.
- Realizar el análisis de costo / beneficio para constatar si la construcción del equipo es rentable en comparación con un módulo de similares características disponible en el mercado

HIPÓTESIS:



Tarjeta Master

Esta tarjeta es la encargada del procesamiento de datos, control y monitoreo de la mini planta industrial así como también de la gestión de datos de los módulos de expansión, la interfaz con LabVIEW y de la comunicación MODBUS.

Tarjeta de Entradas Digitales

Esta tarjeta es la encargada de adquisición de señales de entradas digitales con niveles de tensión de 10 – 24 VDC y del envío de los datos adquiridos a la tarjeta master.

Tarjeta de Salidas Digitales.

Esta tarjeta es la encargada de la gestión de las salidas digitales con los datos recibidos de la tarjeta master, esta constara de 8 salidas de tipo relé.

Tarjeta de Entradas Analógicas.

Esta tarjeta es la encargada de adquisición de señales de entradas analógicas de tipo lazo de corriente (4 – 20 mA), de niveles de tensión (0 – 5 VDC y 0 – 10 VDC) y del envío de los datos adquiridos a la tarjeta master.

Tarjeta de Salidas Analógicas

Esta tarjeta es la encargada de la gestionar de datos de las salidas analógicas entregadas por la tarjeta master para generar los lazos de corriente (4 – 10 mA) y niveles de tensión (0 – 5 VDC y 0 – 10 VDC).

CAPITULACIÓN

1. MARCO TEORICO

ACONDICIONADORES DE SEÑAL

- Tipos de Sensores.
- Elementos Finales de Control.
- Acciones Básicas de Control.
- Sistemas de Control Realimentados.
- Estabilidad en un Sistema de Control.
- Tipos de señales de medición.
- Clasificación de las señales de medición.
- Amplificadores operacionales.
- Circuitos de acondicionamiento de señal.

2. DISEÑO DEL HARDWARE.

- Altium Designer, breve introducción.
- Diseño de la Fuente de alimentación.
- Diseño de circuitos acondicionadores de señal.
- Diseño de la Tarjeta Master para adquisición de datos.
- Diseño de circuitos actuadores.
- Diseño de tarjetas de expansión, I/O digitales y Analógicas.
- Protocolo de pruebas preliminares de las tarjetas diseñadas.

3. DESARROLLO DEL SOFTWARE Y FIRMWARE

- LabView y NI-VISA
- Measurement & Automation Introducción
- El puerto USB
- El software PCWHD Compiler, y la programación en C para el Microcontrolador.
- Comunicación entre tarjetas Master-Slave via RS-485.

- Firmware para las tarjetas de adquisición de datos. (I/O digitales y Analógicas).
- Configuración del puerto USB del Microcontrolador PIC18F2550 en PCWHD Compiler de la tarjeta Master.
- Creación de drivers e instalación del Hardware.
- Establececiendo la comunicación entre el Hardaware y Labview vía USB.

4. PROCOCOLO DE PRUEBAS

- Prueba de sensores de temperatura y presión del modulo Master.
- Prueba actuadores del Modulo Master.
- Pruebas de los módulos de expansión.
- Pruebas de comunicación USB NI-VISA con la tarjeta de adquisición de datos.
- Pruebas en conjunto sensores, actuadores, comunicación, control y monitoreo.

RESUMEN

El proyecto presentado es un módulo de adquisición de datos el cual consta de algunas partes como son: tarjetas de entradas y salidas digitales, tarjetas de entradas y salida análogas, tarjeta master de procesos, tarjeta master de comunicaciones, fuente de alimentación.

Este módulo de adquisición podrá ser conectado a diferentes sistemas SCADA mediante servidores OPC Servers de diferentes marcas y tiene conexión directa con USB – VISA de National Instruments ya que posee la configuración adecuada para actuar como Dispositivo USB RAW para Labview.

El módulo será conectado a sistemas SCADA y a otros dispositivos que trabajen con Protocolo Modbus, siendo así el módulo un esclavo al cual se le podrá asignar direcciones, estas estarán en un rango de 0 a 15 donde la dirección 0 está reservada para funciones especiales de distribución de mensajes generales.

ÍNDICE GENERAL.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I

MARCO TEÓRICO

Introducción.

En el presente capitulo se tratarán temas tales como el análisis del acondicionamiento de señales, circuitos integrados y microprocesadores utilizados para el diseño y construcción del módulo, se realizará una breve reseña del software utilizado para realizar la adquisición de datos, nos hemos ayudado del programa PIC C compiler o CCS para el desarrollo del firmware por lo que se realizara también una breve descripción de este software de desarrollo y de las principales directivas en lenguaje C.

1.1. ACONDICIONADORES DE SEÑAL

1.1.1. Introducción.

La función de estos dispositivos es la de referenciar los cambios eléctricos a una misma escala de corriente o voltaje, además, provee aislamiento eléctrico y filtraje de la señal con el objeto de proteger el sistema de ruidos originados en el campo.

Una vez acondicionada la señal, la misma se convierte en un valor digital equivalente en el bloque de conversión de datos, generalmente, esta función es llevada a cabo por un circuito de conversión analógico/digital. El ordenador almacena esta información, la cual es utilizada para su análisis y para la toma de decisiones. Simultáneamente, se muestra la información al usuario del sistema en tiempo real.

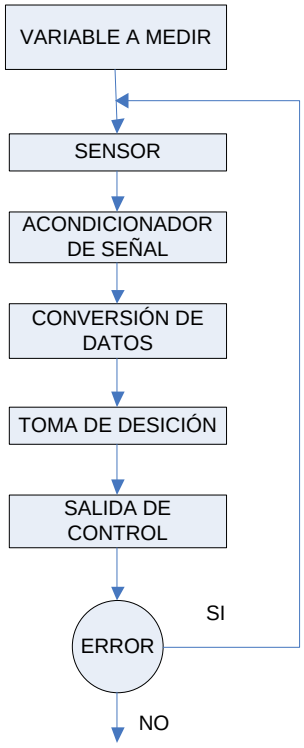


Figura 1.1. Flujo de información en un SCADA

1.1.2. Sensores y su clasificación.

Convierten las variaciones del fenómeno físico en variables eléctricas. Las variables eléctricas más utilizadas son: voltaje, corriente, resistencia o capacitancia. Sin embargo, esta variedad de tipos de señales eléctricas deben ser procesadas para ser entendidas por el ordenador. Para ello se utilizan los acondicionadores de señal.

Así como los seres humanos necesitamos de los sentidos para percibir todo lo que ocurre en nuestro entorno, de manera similar, los sistemas automatizados precisan de los sensores para adquirir información acerca de:

- El cambio de ciertas magnitudes físicas del sistema.
- El estado de los componentes que forman parte del sistema.
- El movimiento y la posición de las piezas fabricadas en el sistema.

Los dispositivos y mecanismos encargados de convertir las magnitudes físicas en señales eléctricas se denominan sensores, los sensores, en función del tipo de señal que transmiten, se pueden clasificar en:

1.1.2.1. Sensores on – off.

Suministran una señal eléctrica claramente diferenciada (verdadera o falsa). Los finales de carrera son un claro ejemplo de este tipo de sensores.

1.1.2.2. Sensores numéricos.

Este tipo de sensores transmiten valores numéricos en forma de combinaciones binarias.

1.1.2.3. Sensores analógicos.

Estos sensores suministran una señal eléctrica variable que es fiel reflejo de la variación de la magnitud física medida. La temperatura es un ejemplo de una magnitud física que necesita de este tipo de sensor.

1.1.3. Elementos Finales de Control y su clasificación.

1.1.3.1. Sensores.

Son los elementos finales de control que, en respuesta a una señal de mando que recibe, actúa sobre la variable o elemento final del proceso. Un actuador transforma la energía de salida del automatismo en otra útil y necesaria para el entorno industrial de trabajo.

Los actuadores pueden ser clasificados en:

- Actuadores eléctricos.
- Actuadores neumáticos.
- Actuadores hidráulicos.

Los más utilizados en la industria son: Cilindros, motores de corriente alterna y motores de corriente continua.

1.1.3.2. Termocuplas.

Una Termocupla se forma al unir dos metales diferentes, como resultado de esta unión aparece entre los extremos libres de los metales una diferencia de potencial que depende de la Temperatura. Este fenómeno se conoce como Efecto Seebeck, quien en 1821 estudió su comportamiento. Este efecto permite calibrar la Termocupla para usarla como un Termómetro.

Figura 1.2. Termocupla típica.

1.1.3.3. RTD (RTD – Resistance Temperature Detector).

Los detectores o sensores de temperatura resistivos están basados en la variación de la resistencia de un conductor con la temperatura. Al calentarse un metal habrá una mayor agitación térmica, dispersándose más los electrones y reduciéndose su velocidad media, aumentando la resistencia. A mayor temperatura, mayor agitación, y mayor resistencia.

La variación de la resistencia puede ser expresada de manera polinómica como sigue a continuación. Por lo general, la variación es bastante lineal en márgenes amplios de temperatura.

Figura 1.3. RTD de tres hilos.

$$\boldsymbol{R = R_o * (1 + \alpha * \Delta T)}$$

Dónde:

$R_o = Resistencia\ referencial\ de\ la\ Termocupla\ a\ temperatura\ T_o$.

$\Delta T =$ Variación de la temperatura respecto la referencia T_o.

$\alpha =$ Coeficiente de temperatura de conductor a 0^oC.

1.1.3.4. Celdas de carga.

Una celda de carga es un transductor que es utilizado para convertir una fuerza en una señal eléctrica. Esta conversión es indirecta y se realiza en dos etapas. Mediante un dispositivo mecánico, la fuerza que se desea medir deforma una galga extensiométrica. La galga extensiométrica convierte el (desplazamiento) o deformación en señales eléctricas. Una celda de carga por lo general se compone de cuatro galga extensiométricas conectadas en una configuración tipo puente de Wheatstone. Sin embargo es posible adquirir celdas de carga con solo uno o dos galga extensiométricas. La señal eléctrica de salida es típicamente del orden de unos pocos mili volts y debe ser amplificada mediante un amplificador de instrumentación antes de que pueda ser utilizada. La salida del transductor se conecta en un algoritmo para calcular la fuerza aplicada al transductor.

Figura 1.4. 5 modelos diferentes de Celas de Carga.

1.1.4. Amplificadores operacionales.

1.1.4.1. Un poco de historia.

En 1965, la compañía Fairchild Semiconductor introdujo en el mercado el uA709, el primer amplificador operacional monolítico ampliamente usado. Aunque disfrutó de un gran éxito, esta primero generación de amplificadores operacionales tenía

muchas desventajas. Este hecho condujo a fabricar un amplificador operacional mejorado, el uA741. Debido a que es muy barato y sencillo de usar, el uA741 ha tenido un enorme éxito. Otros diseños del 741 han aparecido a partir de entonces en el mercado. Por ejemplo, Motorola produce el MC1741, National Semiconductor el LM741 y Texas Instruments el SN72741. Todos estos amplificadores operacionales son equivalentes al uA741, ya que tienen las mismas especificaciones en sus hojas de características. Para simplificar el nombre, la mayoría de la gente ha evitado los prefijos y a este amplificador operacional de gran uso se le llama simplemente 741

1.1.4.2. Circuitos de acondicionamiento de señal.

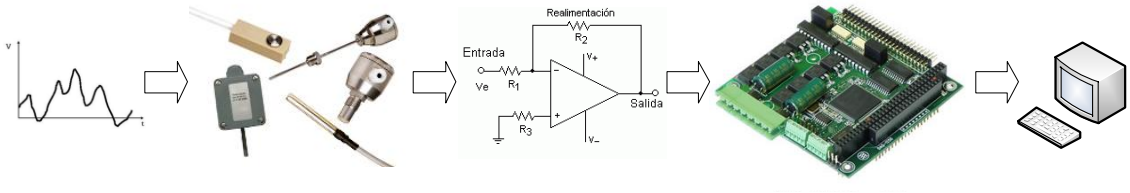


Figura 1.5. Flujo de un sistema de acondicionamiento de señales

Difícilmente un diseñador conecta un transductor directamente y la parte de procesamiento o de despliegue de un sistema, ya que la señal que nos envía nuestro transductor por lo general es muy débil o contiene ruido y componentes que no deseamos, por eso realizamos etapas de acondicionamiento de señales.
Por lo general las señales que recibimos de un transductor deben ser amplificadas a gran escala y no pueden pasar mucha corriente para este fin, por eso se utilizan los amplificadores operacionales, ya que tienen las siguientes características:

- Resistencia de entrada alta (orden de cientos de mega ohms)
- Resistencia de salida baja (debajo de 1Ohmio)
- Buen rango de frecuencias de operación
- Baja sensibilidad a las variaciones de la fuente de alimentación
- Gran estabilidad al cambio de temperatura en el ambiente.

1.2. Adquisición de Datos.

La adquisición de datos o adquisición de señales, consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador u otras electrónicas (sistema digital). Consiste, en tomar un conjunto de señales físicas, convertirlas en tensiones eléctricas y digitalizarlas de manera que se puedan procesar en una computadora. Se requiere una etapa de acondicionamiento, que adecua la señal a niveles compatibles con el elemento que hace la transformación a señal digital. El elemento que hace dicha transformación es el módulo de digitalización o tarjeta de Adquisición de Datos (DAQ).

Figura 1.6. Diagrama de flujos de una conversión análoga a digital

El proceso de adquisición de datos del mundo físico conlleva los siguientes pasos fundamentales:

1. Adquisición de la variable física, se trata de la adquisición de la señal análoga a través de diferentes tipos de conexiones físicas al dispositivo de adquisición de datos.
2. Utilización de un sensor/transductor adecuado para la variable que se desea medir, el cual permite detectar y convertir la variable física a una señal analógica de voltaje o corriente eléctrica.
3. Acondicionador de la señal de voltaje o corriente, si se requiere. Si la señal que proviene del sensor es débil, se requiere un amplificador de voltaje y algún método para filtrar los ruidos eléctricos.
4. El Convertidor A/D de esta señal analógica al lenguaje propio del computador: lenguaje digital. Este proceso se conoce técnicamente como conversión ANÁLOGO/DIGITAL (A/D).

5. Adquisición propiamente dicha de los datos que, en forma digital, podrán ser almacenados en la memoria del micro y llevados luego a pantalla o a otro periférico del computador.

1.3. Altium Designer, breve introducción.

Una solución unificada de productos electrónicos de desarrollo. Altium Designer es el primer y único sistema unificado de desarrollo de productos electrónicos que permite a los ingenieros para tener un diseño desde el concepto hasta su finalización en una única aplicación. Altium Designer reúne a hardware, software y desarrollo de hardware programable dentro de un entorno unificado que permite a todos los aspectos de un producto electrónico que ser concebido y gestionado dentro de un único sistema. Esto, combinado con las capacidades de diseño moderno de gestión de datos permite Altium Designer para trascender las fronteras tradicionales de herramientas, abriendo nuevas posibilidades de diseño que permitirá que los productos electrónicos más inteligentes que se creen más rápido que nunca. Altium Designer le permite aprovechar el potencial que existe dentro de la tecnología actual de la electrónica, dándole la innovación del diseño en un entorno unificado para los dos de hoy y las necesidades de desarrollo de mañana.

1.4. Descripción de protocolos de comunicación que se han utilizado en el módulo de control y adquisición de datos.

1.4.1. El Estándar RS – 232.

1.4.2.1. Historia

En los años 60, cada fabricante usaba una interfaz diferente para comunicar un DTE (Data Terminal Equipment) y un DCE (Data Communications Equipment). Cables, conectores y niveles de voltaje eran diferentes e incompatibles, por lo tanto, la interconexión entre equipos de diferentes fabricantes requería el uso de convertidores de los niveles de voltaje y la fabricación de cables y conectores especiales.

En 1969, el EIA junto con Bell Laboratories y otros fabricantes establecieron un estándar para la interfaz entre DTE's y DCE´s. El objetivo de este estándar era simplificar la interconexión de equipos fabricados por diferentes firmas.

Este estándar llegó a ser el RS-232-C (Recommended Standard number 232, revision C from the Electronic Industry Association). Un estándar similar fue desarrollado en Europa por el CCITT (Comite Consultatif Internatinale de Telegraphie et Telephonie) conocido como V.24 (descripción funcional) y V.28 (especificaciones eléctricas). El RS-232-C fue adoptado por la mayor parte de fabricantes de terminales y equipamiento.

En 1980 la creciente industria de las PC encontró el estándar RS-232-C barato y apropiado para conectar periféricos a la PC. El RS-232-C llego a ser rápidamente un estándar para conectar a la PC: impresoras, cintas de backup, terminales y otras PC’s.

Como el estándar solamente soporta velocidades de transmisión hasta 20 kbps y distancias hasta 16 metros, se adoptaron nuevos estándares por la EIA. El RS449

(descripción mecánica) y RS423 (descripción eléctrica) son compatibles con el RS-232-C y se puede operar a velocidades de hasta 10 Mbps y alcanzar distancias de hasta 1200 metros. Sin embargo, la adopción de un nuevo estándar es un proceso largo y costoso. El RS-232-C está muy expandido y por lo tanto le queda bastante vida.

1.4.2.2. Descripción del estándar

El estándar RS-232-C describe una interfaz entre un DTE y un DCE que emplea un intercambio en serie de datos binarios. En él se definen características eléctricas, mecánicas, funcionales de la interfaz y modos de conexión comunes. Las características eléctricas incluyen parámetros tales como niveles de voltaje e impedancia del cable. La sección mecánica describe los pines. La descripción funcional define las funciones de las señales eléctricas que se usan.

1.4.2.3. Características eléctricas

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

Señales de datos	"0"	"1"	
Emisor (necesario)	de 5 a 15	de -5 a -15	Voltios
Receptor (esperado)	de 3 a 25	de -3 a -25	Voltios
Señales de control	"Off"	"On"	
Emisor (necesario)	de -5 a -15	de 5 a 15	Voltios
Receptor (esperado)	de -3 a -25	de 3 a 25	Voltios

Tabla 1.1. Tabla de niveles de tensión del estándar.

Puede verse que los voltajes del emisor y el receptor son diferentes. Esta definición de los niveles de voltaje compensa las perdidas 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 esta 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.

1.4.2.4. 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.

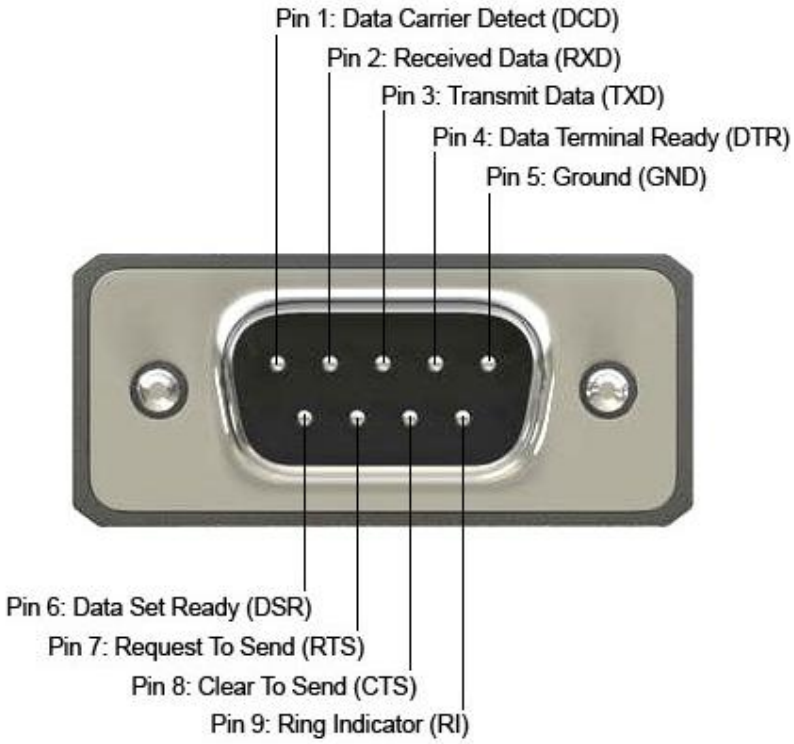


Figura 1.7. Conector DB – 9 usado en el estándar RS – 232.

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 1.2. Señales eléctricas en el conector DB - 9.

1.4.2.5. Métodos de transmisión en serie

Existen dos métodos de transmisión en serie que corrigen errores de bit:

El primero es la comunicación síncrona. El emisor y el receptor son sincronizados usando una señal de reloj que indica el tiempo entre cada bit. Controlando esta señal, el receptor puede determinar si se ha perdido o se ha añadido un bit. Un aspecto a tener en cuenta en este tipo de comunicación es que si alguno de los extremos de la comunicación pierde la señal de reloj, la comunicación finaliza.

El método alternativo, conocido como comunicación asíncrona, es añadir marcadores dentro del flujo de bits para seguir la pista a cada bit de datos. Si se introduce un bit de comienzo que indica el comienzo de un bloque de bits, la posición de cada bit puede ser determinada temporizando los bits en periodos regulares. Enviando bits de comienzo al principio de cada bloque de bits, los dos extremos no tienen que estar sincronizados por una señal de reloj. Al utilizar bloques de pequeño tamaño no hay tiempo para que el temporizador se desincronice. El único factor importante es que receptor y emisor tengan configurada la misma velocidad en el puerto. Los datos se dividen en bloques de 5 a 8 bits llamados palabras. El bit menos significativo de la palabra se envía primero y el más significativo al último. En la comunicación el emisor codifica cada palabra añadiendo al principio de esta un bit de comienzo y uno o dos bits de parada al final. Algunas veces se añade un bit de paridad entre el último bit de la palabra y el bit de parada para comprobar la integridad de los datos.

1.4.2.6. Circuito Integrado MAX232

Descripción

El MAX232 es un circuito integrado de Maxim que convierte las señales de un puerto serie RS-232 a señales compatibles con los niveles TTL de circuitos lógicos. El MAX232 sirve como interfaz de transmisión y recepción para las señales RX, TX, CTS y RTS.

El circuito integrado tiene salidas para manejar niveles de voltaje del RS-232 (aprox. ± 7.5 V) que las produce a partir de un voltaje de alimentación de + 5V utilizando multiplicadores de voltaje internamente en el MAX232 con la adición de condensadores externos.

Esto es de mucha utilidad para la implementación de puertos serie RS-232 en dispositivos que tengan una alimentación simple de +5 V.

Las entradas de recepción de RS-232 (las cuales pueden llegar a ± 25 V), se convierten al nivel estándar de 5 V de la lógica TTL. Éstos receptores tienen un umbral típico de 1.3 V, y una histéresis de 0.5 V.

La versión MAX232A es compatible con la original MAX232, y tiene la mejora de trabajar con mayores velocidades de transferencia de información (mayor tasa de baudios), lo que reduce el tamaño de los condensadores externos utilizados por el multiplicador de voltaje, – 0.1 µF en lugar del 1.0 µF usado en el dispositivo original.

Una versión más nueva de este circuito integrado, el MAX3232 también es compatible con el original, pero opera en un rango más amplio, de 3 a 5.5 V.

Descripción de los pines

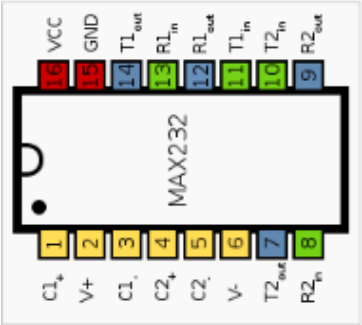


Figura 1.8. Circuito integrado MAX232

- C1+.- Conexión positiva del condensador C1 del doblador de voltaje de +5V a +10V.
- C1-.- Conexión negativa del condensador C1 del doblador de voltaje de +5V a +10V.
- C2+.- Conexión positiva del condensador C2 del inversor de voltaje de +10V a -10V.
- C2-.- Conexión negativa del condensador C2 del inversor de voltaje de +10V a -10V.
- V-.- Conexión de salida del voltaje de -10V.
- V+.- Conexión de salida del voltaje de +10V.
- T1in, T2in, R1out, R2out.- Conexiones a niveles de voltaje de TTL o CMOS.
- T1out, T2out, R1in, R2in.- Conexiones a niveles de voltaje del protocolo RS-232.
- VCC.- Alimentación positiva del MAX232
- GND.- Alimentación negativa del MAX232

Circuito básico de funcionamiento.

El circuito integrado lleva internamente 2 conversores de nivel de TTL a RS-232 y otros 2 de RS-232 a TTL con lo que en total podremos manejar 4 señales del puerto serie del PC.

Por lo general las mas usadas son; TXD, RXD, RTS y CTS. Las dos últimas son las usadas para el protocolo handshaking pero no es imprescindible su uso.

Para que el MAX232 funcione correctamente debemos poner unos condensadores externos, todo esto lo podemos ver en la siguiente figura en la que solo se han cableado las líneas TXD y RXD que son las más usualmente usadas para casi cualquier aplicación:

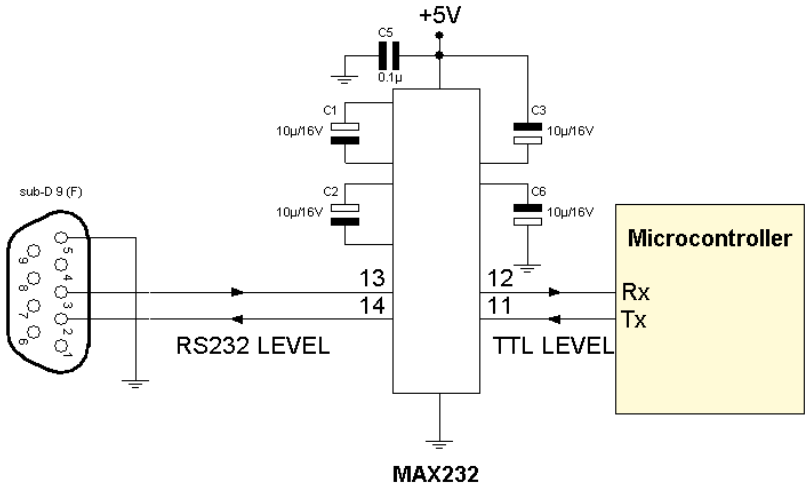


Figura 1.9. Esquema básico de conexión de un circuito integrado MAX232 a un microprocesador.

En los circuitos que emplean el MAX232 todos los condensadores deben ser de 1 microfaradio para llegar hasta 120 Kbps o de 100 nF para llegar hasta 64 Kbps. Para el MAX232A los condensadores han de ser de 100 nF y se consiguen hasta 200 Kbps.

1.4.2. El Estándar RS – 485.

1.4.2.1. La Norma TIA/EIA - 485

Cuando se necesita transmitir a largas distancias o con más altas velocidades que RS-232, RS-485 es la solución. Utilizando enlaces con RS-485 no hay limitación á conectar tan solo dos dispositivos.

Dependiendo de la distancia, velocidad de transmisión y los circuitos integrados que utilicemos, se pueden conectar hasta 32 nodos con un simple par de cables.

Este estándar nos ofrece ventajas como:

- Bajo costo.- Los Circuitos Integrados para trasmitir y recibir son baratos y solo requieren una fuente de +5V para poder generar una diferencia mínima de 1.5v entre las salidas diferenciales. En contraste con RS-232 que en algunos casos requiere de fuentes dobles para alimentar algunos circuitos integrados.
- Capacidad de interconexión.- RS-485 es una interface multi-enlace con la capacidad de poder tener múltiples transmisores y receptores. Con una alta impedancia receptora, los enlaces con RS-485 pueden llegar a tener a lo máximo hasta 256 nodos.
- Longitud de Enlace.- En un enlace RS-485 puede tener hasta 4000 pies de longitud, comparado con RS-232 que tiene unos límites típicos de 50 a 100 pies.
- Rapidez.- La razón de bits puede ser tan alta como 10 Mega bits/ segundo.

1.4.2.2. Balanceo y Desbalanceo de Líneas

La razón por la que RS-485 puede transmitir a largas distancias, es porque utiliza el balanceo de líneas. Cada señal tiene dedicados un par de cables, sobre uno de ellos se encontrará un voltaje y en el otro se estará su complemento, de esta forma, el receptor responde a la diferencia entre voltajes.

La ventaja de las líneas balanceadas es su inmunidad al ruido.

1.4.2.3. Requerimientos de voltaje

Las interfaces típicas RS-485 utilizan una fuente de +5 Volts, pero lo niveles lógicos de los transmisores y receptores no operan a niveles estándares de +5V o voltajes lógicos CMOS. Para una salida válida, la diferencia entre las salidas A y B debe ser al menos +1.5V. Si la interface está perfectamente balanceada, las salidas estarán desfasadas igualmente a un medio de la fuente de Voltaje.

En el receptor RS-485, la diferencia de voltaje entre las entradas A y B necesita ser 0.2V. Si A es al menos 0.2V más positiva que B, el receptor ve un 1 lógico y si B es al menos 0.2v más positivo que A, el receptor ve un 0 lógico. Si la diferencia

entre A y B es menor a 0.2v, el nivel lógico es indefinido. Si esto ocurre habría un error en la transmisión y recepción de la información.

La diferencia entre los requerimientos del Transmisor y el Receptor pueden tener un margen de ruido de 1.3V. La señal diferencial puede atenuarse o tener picos de largo como de 1.3v, y aun así el receptor vera el nivel lógico correcto. El margen de ruido es menor que el de un enlace RS-232, no hay que olvidar que RS-485 maneja señales diferenciales y que cancela la mayoría del ruido a través de su enlace.

El total de corriente utilizada por un enlace RS-485 puede variar debido a las impedancias de los componentes, incluyendo los Transmisores, Receptores, cables y la terminación de los componentes. Una baja impedancia a la salida del Transmisor y una baja impedancia en los cables, facilita los cambios de nivel y asegura que el receptor vea la señal, no importa cuan larga sea la línea de transmisión. Una alta impedancia en el receptor decrementa la corriente en el enlace e incrementa la vida de la fuentes de voltaje.

1.4.2.4. Comunicación Half Duplex

El término Half Duplex en un sistema de comunicación se refiere, a que solamente en un tiempo determinado, el sistema puede transmitir o recibir información, sin embargo no lo puede hacer al mismo tiempo. En muchos enlaces del tipo RS-485 se comparte el BUS.

Existe una línea de control, la cual habilita a los controladores en un solo sentido. Por lo tanto, se debe tener cuidado de no transmitir y recibir al mismo tiempo, ya que se podría crear una superposición de información. La siguiente. Figura muestra el esquema de una comunicación RS-485 en Modo Half Duplex.

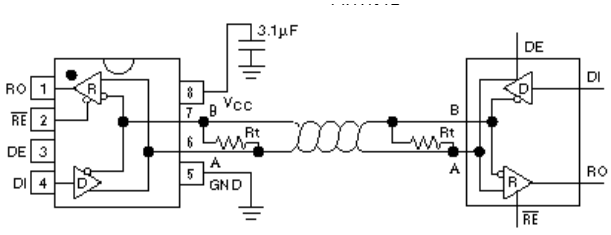


Figura 1.10. Cableado de red Half Duplex de una red RS-485.

1.4.2.5. Comunicación Full Duplex

Para este trabajo se utilizará la comunicación RS-485 en modo Full Duplex, ya que al contar con varios microcontroladores esclavos, se necesita que cada uno de ellos este reportando los datos obtenidos de cada proceso, sin embargo, como no se sabe cuando se necesitará dicha información, se requieren de dos canales, uno independiente del otro, para poder transmitir y recibir al mismo tiempo la información.

El término Full Duplex se refiere a que un sistema puede transmitir y recibir información al simultáneamente. Bajo este concepto la interface RS-485 está diseñada para sistemas multipunto, esto significa que los enlaces pueden llegar a tener más de un transmisor y receptor, ya que cada dirección o sea Transmisión y Recepción tienen su propia ruta. La siguiente figura muestra lo anteriormente dicho.

En la siguiente figura se muestra como es posible utilizar la comunicación Full Duplex con múltiples nodos trasmisores y receptores.

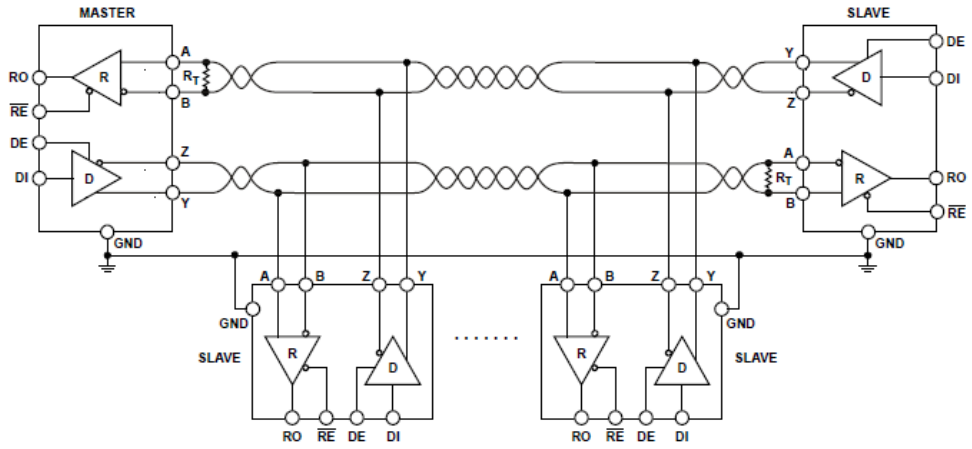


Figura 1.11. Cableado de red Full Duplex de una red RS-485.

En este arreglo del tipo maestro / esclavo, se pondrá como ejemplo que el nodo 1 es el maestro, por lo tanto tiene el control de la red y el asigna el permiso para transmitir. Un par de cables están conectados del nodo trasmisor Maestro a todos los controladores receptores esclavos. En el otro sentido, un par de cables conectan a todos los esclavos al receptor del Maestro.

Todos los esclavos deben leer lo que el maestro envía, pero solo uno va a poder responder y lo hace a través de los cables opuestos.

1.4.2.6. El Circuito Integrado SN75176

Descripción

Para lograr la comunicación con el ordenador se elabora una interface del tipo RS-485, para su elaboración, se utilizan dos circuitos integrados con la matricula SN75176 de Texas Instruments, uno es para la recepción de datos y otro para la transmisión.

Estos dispositivos se encargan de hacer la conversión entre los niveles TTL del microcontrolador y las señales del tipo diferencial que se utilizan el bus

RS-485. Vale la pena decir que en el controlador de transmisión se agregó una línea de habilitación, esto se debe a que todas las salidas de los microcontroladores están conectadas a la línea de recepción del ordenador, así cada uno está siempre deshabilitado para enviar datos y solo se habilitará en el momento en que deba hacer una transmisión, evitando así conflictos o choques de información en la línea o bus de datos, a continuación la sig. Figura hace una breve descripción de este circuito integrado.

Descripción de pines

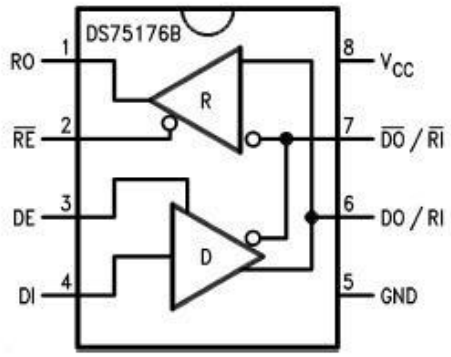


Figura 1.12. Circuito integrado MAX232

- En las termínales VCC y GND se encuentra la alimentación del circuito, que este caso es de +5V.

- La terminal R0 y DI recibe un nivel lógico TTL si y solo si la línea RE se habilita y como se puede observar es con un ‘0’ lógico
- Las terminales D0 y -D0 reciben también el nombre de A y B y son sobre estas líneas las que forman el Bus de Transmisión y Recepción.

Como se puede observar, cada chip consta de un transmisor y un receptor, si las terminales RE (Pin 2) y DE (Pin 3) se unen entre si con un solo Bit se puede controlar el flujo de la información.

1.4.3. Protocolo SPI (Serial Peripherical Interface)

1.4.3.1. Introducción.

SPI es un bus de tres líneas, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas tres líneas porta la información entre los diferentes dispositivos conectados al bus. Cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo, por lo que este tipo de comunicación serial es full dúplex. Dos de estas líneas trasfieren los datos (una en cada dirección) y la tercer línea es la del reloj. Algunos dispositivos solo pueden ser transmisores y otros solo receptores, generalmente un dispositivo que tramite datos también puede recibir.

Un ejemplo podría ser una memoria EEPROM, el cual es un dispositivo que puede transmitir y recibir información.

Los dispositivos conectados al bus son definidos como maestros y esclavos. Un maestro es aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y control.

Un esclavo es un dispositivo controlado por el maestro. Cada esclavo es controlado sobre el bus a través de una línea selectora llamada Chip Select o Select Slave, por lo tanto es esclavo es activado solo cuando esta línea es seleccionada. Generalmente una línea de selección es dedicada para cada esclavo.

En un tiempo determinado T1, solo podrá existir un maestro sobre el bus. Cualquier dispositivo esclavo que no esté seleccionado, debe deshabilitarse (ponerlo en alta impedancia) a través de la línea selectora (chip select).

1.4.3.2. Especificaciones del bus

Todas las líneas del bus transmiten la información sobre una sola dirección.
La señal sobre la línea de reloj (SCLK) es generada por el maestro y sincroniza la transferencia de datos, la línea MOSI (Master Out Slave In) transporta los datos del maestro hacia el esclavo y la línea MISO (Master In Slave Out) transporta los datos del esclavo hacia el maestro.

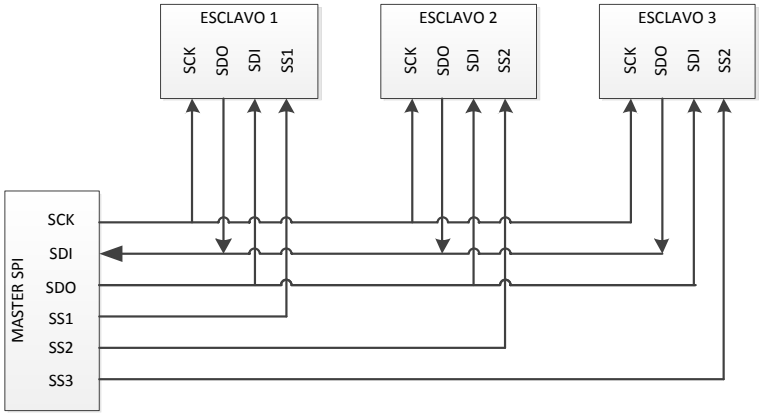


Figura 1.13. Diagrama de conexiones en una red SPI

Cada esclavo es seleccionado por un nivel lógico bajo (‘0’) a través de la línea (CS = Chip Select o SS Slave Select). Los datos sobre este bus pueden ser transmitidos a una razón de casi cero bits /segundo hasta 1 Mbits/ segundo. Los datos son transferidos en bloques de 8 bits, en donde el bits más significativo (MSB) se transmite primero.

1.4.3.3. Ventajas Bus SPI

- Comunicación Full Duplex
- Mayor velocidad de transmisión que con I²C o SMBus
- Protocolo flexible en que se puede tener un control absoluto sobre los bits transmitidos
- No está limitado a la transferencia de bloques de 8 bits

- Elección del tamaño de la trama de bits, de su significado y propósito
- Su implementación en hardware es extremadamente simple
- Consume menos energía que I²C o que SMBus debido que posee menos circuitos (incluyendo las resistencias pull-up) y estos son más simples
- No es necesario arbitraje o mecanismo de respuesta ante fallos
- Los dispositivos clientes usan el reloj que envía el servidor, no necesitan por tanto su propio reloj
- No es obligatorio implementar un transceptor (emisor y receptor), un dispositivo conectado puede configurarse para que solo envíe, sólo reciba o ambas cosas a la vez
- Usa mucho menos terminales en cada chip/conector que una interfaz paralelo equivalente
- Como mucho una única señal específica para cada cliente (señal SS), las demás señales pueden ser compartidas

1.4.3.4. Desventajas Bus SPI

- Consume más pines de cada chip que I²C, incluso en la variante de 3 hilos
- El direccionamiento se hace mediante líneas específicas (señalización fuera de banda) a diferencia de lo que ocurre en I²C que se selecciona cada chip mediante una dirección de 7 bits que se envía por las mismas líneas del bus
- No hay control de flujo por hardware
- No hay señal de asentimiento. El servidor podría estar enviando información sin que estuviese conectado ningún cliente y no se daría cuenta de nada
- No permite fácilmente tener varios servidores conectados al bus
- Sólo funciona en las distancias cortas a diferencia de, por ejemplo, RS-232, RS-485, o Bus CAN

1.4.4. Protocolo I^2C

1.4.4.1. Introducción.

I2C (pronunciado I cuadrado C) es un estándar que facilita la comunicación entre distintos dispositivos (microcontroladores, memorias, monitores de computadoras, y muchos otros dispositivos que tengan "inteligencia").

Solamente requiere dos líneas de señal: Una de datos y una de clock (además de la masa común). Fue diseñado por Philips y la velocidad a la que se lo suele usar es 100 Kbits por segundo, pero hay algunos dispositivos que pueden transmitir más rápido.

El bus I2C es serie (todos los datos van por una misma señal "uno atrás del otro") y sincrónico (una de las señales se usa para sincronizar y marcar el tiempo).

La señal de datos se llama SDA, la de clock se llama SCL o SCK. Como son señales de colector abierto, es necesario agregar resistencias de pullup de entre 2K y 50K (típicamente se usa 10K).

El protocolo se basa en distintas señales, que por lo general son enviadas por el dispositivo maestro. En estado de reposo ambas líneas permanecen en un estado lógico "1".

La señal de start y la señal de stop, que se hacen para indicar el comienzo/finalización de un pedido a un dispositivo esclavo, son los únicos tipos de señal en la que se permite que SDA cambie cuando SCK está en alto:

1.4.4.2. La comunicación en más detalle

Cuando el dispositivo maestro quiere comunicarse con un esclavo, produce una secuencia de inicio en el bus. La secuencia de inicio es una de las dos secuencias especiales que se han definido en el bus I2C; la otra es la secuencia de parada. Las secuencias de inicio y la de parada son especiales porque son los dos únicos casos en que se permite que la línea de datos (SDA) cambie cuando la línea de reloj (SCL) está alta. Cuando se están transmitiendo datos, la línea SDA debe permanecer estable, y jamás cambiar, mientras la línea SCL está alta. Las

secuencias de inicio y de parada señalan el comienzo y el final de una transacción con los dispositivos esclavos.

Los datos se transfieren en secuencias de 8 bits. Estos bits se colocan en la línea SDA comenzando por el bit de más peso (o más significativo). Una vez puesto un bit en SDA, se lleva la línea SCL a alto. Debemos recordar que el chip no puede llevar la línea a un estado alto, en realidad, lo que hace es "soltarla", y el que la pone en nivel lógico alto es el resistor de polarización. Por cada 8 bits que se transfieren, el dispositivo que recibe el dato envía de regreso un bit de reconocimiento, de modo que en realidad por cada byte de dato se producen 9 pulsos sobre la línea SCL (es decir, 9 pulsos de reloj por cada 8 bits de dato). Si el dispositivo que recibe envía un bit de reconocimiento bajo, indica que ha recibido el dato y que está listo para aceptar otro byte. Si retorna un alto, lo que indica es que no puede recibir más datos y el dispositivo maestro debería terminar la transferencia enviando una secuencia de parada.

1.4.4.3. Funcionamiento

Como dijimos, las líneas SDA y SCL transportan información entre los dispositivos conectados al bus.

Cada dispositivo es reconocido por su código (dirección) y puede operar como transmisor o receptor de datos. Además, cada dispositivo puede ser considerado como Master o Slave.

El Master es el dispositivo que inicia la transferencia en el bus y genera la señal de Clock mientras que el Slave (esclavo) es el dispositivo direccionado.

Las líneas SDA (serial Data) y SCL (serial Clock) son bidireccionales, conectadas al positivo de la alimentación a través de las resistencias de pullup. Cuando el bus está libre, ambas líneas están en nivel alto. La transmisión bidireccional serie (8-bits) de datos puede realizarse a 100Kbits/s en el modo estándar o 400 Kbits/s en el modo rápido.

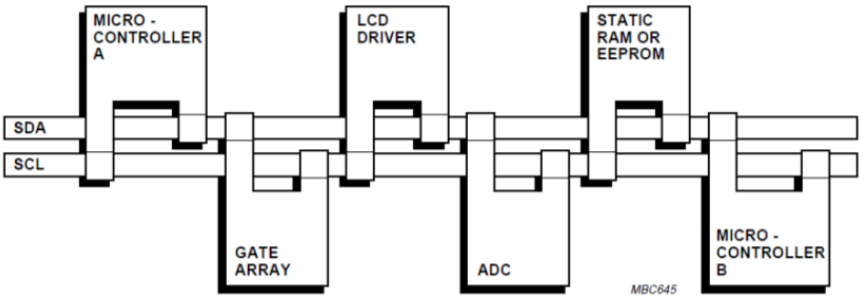


Figura 1.14. Bus I2C Con distintos tipos de dispositivos

1.4.4.4. Descripción de las señales

- SCL (System Clock) es la línea de los pulsos de reloj que sincronizan el sistema.
- SDA (System Data) es la línea por la que se mueven los datos entre los dispositivos.
- GND (Masa) común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pullup") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

Las dos líneas del bus están en un nivel lógico alto cuando están inactivas. En principio, el número de dispositivos que se puede conectar al bus no tiene límites, aunque hay que observar que la capacidad máxima sumada de todos los dispositivos no supere los 400 pF. El valor de los resistores de polarización no es muy crítico, y puede ir desde 1K8 (1.800 ohms) a 47K (47.000 ohms). Un valor menor de resistencia incrementa el consumo de los integrados pero disminuye la sensibilidad al ruido y mejora el tiempo de los flancos de subida y bajada de las señales. Los valores más comunes en uso son entre 1K8 y 10K.

1.4.4.5. Condiciones de START y STOP.

Antes de que se establezca un intercambio de datos entre el circuito Master y los Esclavos, el Master debe informar el comienzo de la comunicación (condición de Start): la línea SDA cae a cero mientras SCL permanece en nivel alto. A partir de este momento comienza la transferencia de datos. Una vez finalizada la comunicación se debe informar de esta situación (condición de Stop). La línea SDA pasa a nivel alto mientras SCL permanece en estado alto.

1.4.4.6. Transferencia de datos.

El Maestro genera la condición de Start Y cada palabra puesta en el bus SDA debe tener 8 bits, la primera palabra transferida contiene la dirección del Esclavo seleccionado luego el Master lee el estado de la línea SDA, si vale 0 (impuesto por el esclavo), el proceso de transferencia continúa. Si vale 1, indica que el circuito direccionado no valida la comunicación, entonces, el Maestro genera un bit de stop para liberar el bus I2C.

Este acuse de recibo se denomina SCK (acknowledge) y es una parte importante del protocolo I2C.

Al final de la transmisión, el Maestro genera la condición de Stop y libera el bus I2C, las líneas SDA y SCL pasan a estado alto.

1.4.4.7. Direccionamiento de dispositivos en el bus I2C

Lo más común en los dispositivos para el bus I2C es que utilicen direcciones de 7 bits, aunque existen dispositivos de 10 bits. Este último caso es raro.

Una dirección de 7 bits implica que se pueden poner hasta 128 dispositivos sobre un bus I2C, ya que un número de 7 bits puede ir desde 0 a 127. Cuando se envían las direcciones de 7 bit, de cualquier modo la transmisión es de 8 bits. El bit extra se utiliza para informarle al dispositivo esclavo si el dispositivo maestro va a escribir o va a leer datos desde él. Si el bit de lectura/escritura (R/W) es cero, el dispositivo maestro está escribiendo en el esclavo. Si el bit es 1 el maestro está leyendo desde el esclavo. La dirección de 7 bit se coloca en los 7 bist más significativos del byte y el bit de lectura/escritura es el bit menos significativo.

El hecho de colocar la dirección de 7 bits en los 7 bits más significativos del byte produce confusiones entre quienes comienzan a trabajar con este bus. Si, por ejemplo, se desea escribir en la dirección 21 (hexadecimal), en realidad se debe enviar un 42, que es un 21 desplazado un bit hacia arriba. También se pueden tomar las direcciones del bus I2C como direcciones de 8 bit, en las que las pares son de sólo escritura y las impares son de sólo lectura. Para dar un ejemplo, el integrado de brújula magnética CMPS03 es fijado en fábrica en la dirección 0xC0 ($C0). La dirección 0xC0 se utiliza para escribir en él y la dirección 0xC1 es para leer de él.

1.4.4.8. Ventajas BUS I2C

- Usa solo 2 pines para establecer la comunicación entre dispositivos (SDA y SCL) a diferencia de SPI que usa mínimo tres pines.
- Su implementación en hardware es extremadamente simple
- Los dispositivos *clientes* usan el reloj que envía el *servidor*, no necesitan por tanto su propio reloj
- No es obligatorio implementar un transceptor (emisor y receptor), un dispositivo conectado puede configurarse para que solo envíe, sólo reciba o ambas cosas a la vez.

1.4.4.9. Desventajas BUS I2C

- El control de flujo de la información se lo realiza con funcione propias del protocolo I2C, carece de un mecanismo que realice esta función mediante Hardware.
- Sólo funciona en las distancias cortas a diferencia de, por ejemplo, RS-232, RS-485, o Bus CAN

1.4.5. Protocolo TCP/IP

1.4.5.1. Introducción

Internet no es un nuevo tipo de red física, sino un conjunto de tecnologías que permiten interconectar redes muy distintas entre sí. Internet no es dependiente de

la máquina ni del sistema operativo utilizado. De esta manera, podemos transmitir información entre un servidor Unix y un ordenador que utilice Windows XP. O entre plataformas completamente distintas como Macintosh, Alpha o Intel. Es más: entre una máquina y otra generalmente existirán redes distintas: redes Ethernet, redes Token Ring e incluso enlaces vía satélite. Como vemos, está claro que no podemos utilizar ningún protocolo que dependa de una arquitectura en particular. Lo que estamos buscando es un método de interconexión general que sea válido para cualquier plataforma, sistema operativo y tipo de red. La familia de protocolos que se eligieron para permitir que Internet sea una Red de redes es TCP/IP. Nótese aquí que hablamos de familia de protocolos ya que son muchos los protocolos que la integran, aunque en ocasiones para simplificar hablemos sencillamente del protocolo TCP/IP.

El protocolo TCP/IP tiene que estar a un nivel superior del tipo de red empleado y funcionar de forma transparente en cualquier tipo de red. Y a un nivel inferior de los programas de aplicación (páginas WEB, correo electrónico…) particulares de cada sistema operativo. Todo esto nos sugiere el siguiente modelo de referencia:

Capa de aplicación (HTTP, SMTP, FTP, TELNET...)
Capa de transporte (UDP, TCP)
Capa de red (IP)
Capa de acceso a la red (Ethernet, Token Ring...)
Capa física (cable coaxial, par trenzado...)

Tabla 1.3. Capas OSI que usa el protocolo TCP/IP

El nivel más bajo es la *capa física*. Aquí nos referimos al medio físico por el cual se transmite la información. Generalmente será un cable aunque no se descarta cualquier otro medio de transmisión como ondas o enlaces vía satélite.

La *capa de acceso a la red* determina la manera en que las estaciones (ordenadores) envían y reciben la información a través del soporte físico

proporcionado por la capa anterior. Es decir, una vez que tenemos un cable, ¿cómo se transmite la información por ese cable? ¿Cuándo puede una estación transmitir? ¿Tiene que esperar algún turno o transmite sin más? ¿Cómo sabe una estación que un mensaje es para ella? Pues bien, son todas estas cuestiones las que resuelve esta capa.

Las dos capas anteriores quedan a un nivel inferior del protocolo TCP/IP, es decir, no forman parte de este protocolo. La *capa de red* define la forma en que un mensaje se transmite a través de distintos tipos de redes hasta llegar a su destino. El principal protocolo de esta capa es el IP aunque también se encuentran a este nivel los protocolos ARP, ICMP e IGMP. Esta capa proporciona el direccionamiento IP y determina la ruta óptima a través de los encaminadores (routers) que debe seguir un paquete desde el origen al destino.

La *capa de transporte* (protocolos TCP y UDP) ya no se preocupa de la ruta que siguen los mensajes hasta llegar a su destino. Sencillamente, considera que la comunicación extremo a extremo está establecida y la utiliza. Además añade la noción de puertos, como veremos más adelante.

Una vez que tenemos establecida la comunicación desde el origen al destino nos queda lo más importante, ¿qué podemos transmitir? La *capa de aplicación* nos proporciona los distintos servicios de Internet: correo electrónico, páginas Web, FTP, TELNET, etc.

1.4.5.2. Capa de red

La familia de protocolos TCP/IP fue diseñada para permitir la interconexión entre distintas redes. El mejor ejemplo de interconexión de redes es Internet: se trata de un conjunto de redes unidas mediante encaminadores o routers.

A lo largo de este Curso aprenderemos a construir redes privadas que funcionen siguiendo el mismo esquema de Internet. En una red TCP/IP es posible tener, por ejemplo, servidores web y servidores de correo para uso interno. Obsérvese que

todos los servicios de Internet se pueden configurar en pequeñas redes internas TCP/IP.

A continuación veremos un ejemplo de interconexión de 3 redes. Cada host (ordenador) tiene una dirección física que viene determinada por su adaptador de red. Estas direcciones se corresponden con la capa de acceso al medio y se utilizan para comunicar dos ordenadores que pertenecen a la misma red. Para identificar globalmente un ordenador dentro de un conjunto de redes TCP/IP se utilizan las direcciones IP (capa de red). Observando una dirección IP sabremos si pertenece a nuestra propia red o a una distinta (todas las direcciones IP de la misma red comienzan con los mismos números, según veremos más adelante).

Host	Dirección física	Dirección IP	Red
A	00-60-52-0B-B7-7D	192.168.0.10	Red 1
R1	00-E0-4C-AB-9A-FF	192.168.0.1	
	A3-BB-05-17-29-D0	10.10.0.1	Red 2
B	00-E0-4C-33-79-AF	10.10.0.7	
R2	B2-42-52-12-37-BE	10.10.0.2	
	00-E0-89-AB-12-92	200.3.107.1	Red 3
C	A3-BB-08-10-DA-DB	200.3.107.73	
D	B2-AB-31-07-12-93	200.3.107.200	

Tabla 1.4. Tabla de direcciones en el ejemplo de la figura 1.17

El concepto de red está relacionado con las direcciones IP que se configuren en cada ordenador, no con el cableado. Es decir, si tenemos varias redes dentro del mismo cableado solamente los ordenadores que permanezcan a una misma red podrán comunicarse entre sí. Para que los ordenadores de una red puedan comunicarse con los de otra red es necesario que existan routers que interconecten las redes. Un router o encaminador no es más que un ordenador con varias direcciones IP, una para cada red, que permita el tráfico de paquetes entre sus redes.

La capa de red se encarga de fragmentar cada mensaje en paquetes de datos llamados datagramas IP y de enviarlos de forma independiente a través de la red de redes. Cada datagrama IP incluye un campo con la dirección IP de destino. Esta información se utiliza para enrutar los datagramas a través de las redes necesarias que los hagan llegar hasta su destino.

Nota: Cada vez que visitamos una página web o recibimos un correo electrónico es habitual atravesar un número de redes comprendido entre 10 y 20, dependiendo de la distancia de los hosts. El tiempo que tarda un datagrama en atravesar 20 redes (20 routers) suele ser inferior a 600 milisegundos.

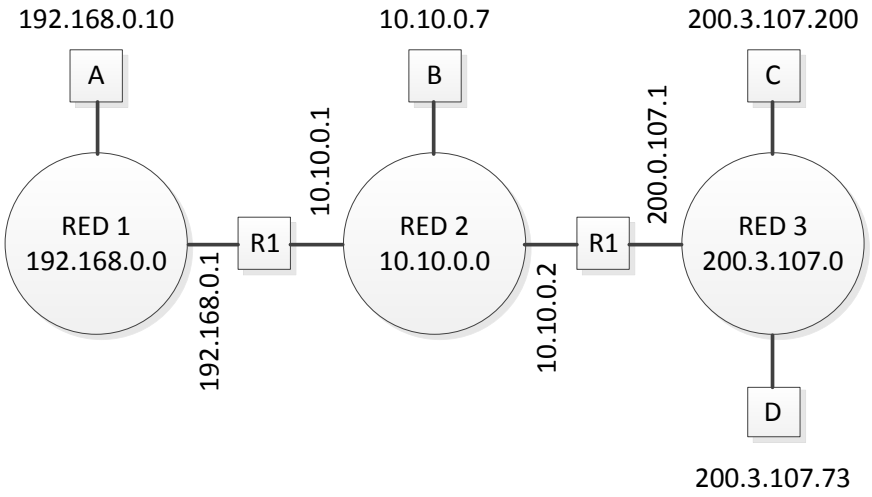


Figura 1.15. Diagrama de interconexión entre redes.

En el ejemplo anterior, supongamos que el ordenador 200.3.107.200 (D) envía un mensaje al ordenador con 200.3.107.73 (C). Como ambas direcciones comienzan con los mismos números, D sabrá que ese ordenador se encuentra dentro de su propia red y el mensaje se entregará de forma directa. Sin embargo, si el ordenador 200.3.107.200 (D) tuviese que comunicarse con 10.10.0.7 (B), D advertiría que el ordenador destino no pertenece a su propia red y enviaría el mensaje al router R2 (es el ordenador que le da salida a otras redes). El router entregaría el mensaje de forma directa porque B se encuentra dentro de una de sus redes (la Red 2).

1.4.5.3. Dirección IP.

La dirección IP es el identificador de cada host dentro de su red de redes. Cada host conectado a una red tiene una dirección IP asignada, la cual debe ser distinta a todas las demás direcciones que estén vigentes en ese momento en el conjunto de redes visibles por el host. En el caso de Internet, no puede haber dos ordenadores con 2 direcciones IP (públicas) iguales. Pero sí podríamos tener dos ordenadores con la misma dirección IP siempre y cuando pertenezcan a redes independientes entre sí (sin ningún camino posible que las comunique).
Las direcciones IP se clasifican en:
Direcciones IP públicas.- Son visibles en todo Internet. Un ordenador con una IP pública es accesible (visible) desde cualquier otro ordenador conectado a Internet. Para conectarse a Internet es necesario tener una dirección IP pública.

Direcciones IP privadas (reservadas).- Son visibles únicamente por otros hosts de su propia red o de otras redes privadas interconectadas por routers. Se utilizan en las empresas para los puestos de trabajo. Los ordenadores con direcciones IP privadas pueden salir a Internet por medio de un router (o proxy) que tenga una IP pública. Sin embargo, desde Internet no se puede acceder a ordenadores con direcciones IP privadas. A su vez, las direcciones IP pueden ser:

Direcciones IP estáticas (fijas).- Un host que se conecte a la red con dirección IP estática siempre lo hará con una misma IP. Las direcciones IP públicas estáticas son las que utilizan los servidores de Internet con objeto de que estén siempre localizables por los usuarios de Internet. Estas direcciones hay que contratarlas.

Direcciones IP dinámicas.- Un host que se conecte a la red mediante dirección IP dinámica, cada vez lo hará con una dirección IP distinta. Las direcciones IP públicas dinámicas son las que se utilizan en las conexiones a Internet mediante un módem. Los proveedores de Internet utilizan direcciones IP dinámicas debido a que tienen más clientes que direcciones IP (es muy improbable que todos se conecten a la vez).

Las direcciones IP están formadas por 4 bytes (32 bits). Se suelen representar de la forma a.b.c.d donde cada una de estas letras es un número comprendido entre el 0 y el 255.

Las direcciones IP también se pueden representar en hexadecimal, desde la 00.00.00.00 hasta la FF.FF.FF.FF o en binario, desde la 00000000.00000000.00000000.00000000 hasta la 11111111.11111111.11111111.11111111.

Las tres direcciones siguientes representan a la misma máquina (podemos utilizar la calculadora científica de Windows para realizar las conversiones).

- (decimal) 128.10.2.30
- (hexadecimal) 80.0A.02.1E
- (binario) 10000000.00001010.00000010.00011110

¿Cuántas direcciones IP existen? Si calculamos 2 elevado a 32 obtenemos más de 4000 millones de direcciones distintas. Sin embargo, no todas las direcciones son válidas para asignarlas a hosts. Las direcciones IP no se encuentran aisladas en Internet, sino que pertenecen siempre a alguna red. Todas las máquinas conectadas a una misma red se caracterizan en que los primeros bits de sus direcciones son iguales. De esta forma, las direcciones se dividen conceptualmente en dos partes: el identificador de red y el identificador de host.

Dependiendo del número de hosts que se necesiten para cada red, las direcciones de Internet se han dividido en las clases primarias A, B y C. La clase D está formada por direcciones que identifican no a un host, sino a un grupo de ellos. Las direcciones de clase E no se pueden utilizar (están reservadas).

	0	1	2	3	4	8	16	24	31
Clase A	0	red				host			
Clase B	1	0	red				host		
Clase C	1	1	0	red				host	
Clase D	1	1	1	0	grupo de multicast (multidifusión)				
Clase E	1	1	1	1	(direcciones reservadas: no se pueden utilizar)				

Tabla 1.5. Clases de direcciones de red

Clase	Formato (r=red, h=host)	Número de redes	Número de hosts por red	Rango de direcciones de redes	Máscara de subred
A	r.h.h.h	128	16.777.214	0.0.0.0 - 127.0.0.0	255.0.0.0
B	r.r.h.h	16.384	65.534	128.0.0.0 - 191.255.0.0	255.255.0.0
C	r.r.r.h	2.097.152	254	192.0.0.0 - 223.255.255.0	255.255.255.0
D	grupo	-	-	224.0.0.0 - 239.255.255.255	-
E	no válidas	-	-	240.0.0.0 - 255.255.255.255	-

Tabla 1.6. Formato de las clases de direcciones de red.

1.4.5.4. Máscara de subred

Una máscara de subred es aquella dirección que enmascarando nuestra dirección IP, nos indica si otra dirección IP pertenece a nuestra subred o no.

La siguiente tabla muestra las máscaras de subred correspondientes a cada clase:

Clase	Máscara de subred
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

Tabla 1.7. Máscara de subred de las clases de direcciones IP.

Los unos indican los bits de la dirección correspondientes a la red y los ceros, los correspondientes al host. Según la máscara anterior, el primer byte (8 bits) es la red y los tres siguientes (24 bits), el host. Por ejemplo, la dirección de clase A 35.120.73.5 pertenece a la red 35.0.0.0.

Supongamos una subred con máscara 255.255.0.0, en la que tenemos un ordenador con dirección 148.120.33.110. Si expresamos esta dirección y la de la máscara de subred en binario, tenemos:

```
148.120.33.110   10010100.01111000.00100001.01101110   (dirección de una máquina)
255.255.0.0      11111111.11111111.00000000.00000000   (dirección de su máscara de red)
148.120.0.0      10010100.01111000.00000000.00000000   (dirección de su subred)
                 <----------RED--------> <---------HOST-------->
```

Al hacer el producto binario de las dos primeras direcciones (donde hay dos 1 en las mismas posiciones ponemos un 1 y en caso contrario, un 0) obtenemos la tercera.

Si hacemos lo mismo con otro ordenador, por ejemplo el 148.120.33.89, obtenemos la misma dirección de subred. Esto significa que ambas máquinas se encuentran en la misma subred (la subred 148.120.0.0).

148.120.33.89	10010100.01111000.00100001.01011001	(dirección de una máquina)
255.255.0.0	11111111.11111111.00000000.00000000	(dirección de su máscara de red)
148.120.0.0	10010100.01111000.00000000.00000000	(dirección de su subred)

En cambio, si tomamos la 148.115.89.3, observamos que no pertenece a la misma subred que las anteriores.

148.115.89.3	10010100.01110011.01011001.00000011	(dirección de una máquina)
255.255.0.0	11111111.11111111.00000000.00000000	(dirección de su máscara de red)
148.115.0.0	10010100.01110011.00000000.00000000	(dirección de su subred)

1.4.5.5. Protocolo IP

IP es el principal protocolo de la capa de red. Este protocolo define la unidad básica de transferencia de datos entre el origen y el destino, atravesando toda la red de redes. Además, el software IP es el encargado de elegir la ruta más adecuada por la que los datos serán enviados. Se trata de un sistema de entrega de paquetes (llamados datagramas IP) que tiene las siguientes características:

- Es no orientado a conexión debido a que cada uno de los paquetes puede seguir rutas distintas entre el origen y el destino. Entonces pueden llegar duplicados o desordenados.
- Es no fiable porque los paquetes pueden perderse, dañarse o llegar retrasados.

1.4.6. Protocolo Modbus.

Es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de comunicaciones estándar de facto en la industria es el que goza de mayor disponibilidad para la conexión de

dispositivos electrónicos industriales. Las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones son:

1. Es público
2. Su implementación es fácil y requiere poco desarrollo
3. Maneja bloques de datos sin suponer restricciones

Modbus permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a un ordenador. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión adquisición de datos (SCADA). Existen versiones del protocolo Modbus para puerto serie y Ethernet (Modbus/TCP).

Existen dos variantes, con diferentes representaciones numéricas de los datos y detalles del protocolo ligeramente desiguales. Modbus RTU es una representación binaria compacta de los datos. Modbus ASCII es una representación legible del protocolo pero menos eficiente. Ambas implementaciones del protocolo son serie. El formato RTU finaliza la trama con un suma de control de redundancia cíclica (CRC), mientras que el formato ASCII utiliza una suma de control de redundancia longitudinal (LRC). La versión Modbus/TCP es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP.

Modbus Plus (Modbus+ o MB+), es una versión extendida del protocolo y privativa de Modicon. Dada la naturaleza de la red precisa un coprocesador dedicado para el control de la misma. Con una velocidad de 1 Mbit/s en un par trenzado sus especificaciones son muy semejantes al estándar EIA/RS-485 aunque no guarda compatibilidad con este.

Cada dispositivo de la red Modbus posee una dirección única. Cualquier dispositivo puede enviar órdenes Modbus, aunque lo habitual es permitirlo sólo a un dispositivo maestro. Cada comando Modbus contiene la dirección del dispositivo destinatario de la orden. Todos los dispositivos reciben la trama pero sólo el destinatario la ejecuta (salvo un modo especial denominado "Broadcast"). Cada uno de los mensajes incluye información redundante que asegura su

integridad en la recepción. Los comandos básicos Modbus permiten controlar un dispositivo RTU para modificar el valor de alguno de sus registros o bien solicitar el contenido de dichos registros.

1.4.6.1. Estableciendo la comunicación MODBUS.

Medio Físico

El medio físico de conexión puede ser un bus semidúplex (half duplex) (RS-485 o fibra óptica) o dúplex (full duplex) (RS-422, BC 0-20mA o fibra óptica).

La comunicación es asíncrona y las velocidades de transmisión previstas van desde los 75 baudios a 19.200 baudios. La máxima distancia entre estaciones depende del nivel físico, pudiendo alcanzar hasta 1200 m sin repetidores.

Acceso al Medio

La estructura lógica es del tipo maestro-esclavo, con acceso al medio controlado por el maestro. El número máximo de estaciones previsto es de 63 esclavos más una estación maestra.

Los intercambios de mensajes pueden ser de dos tipos:

- Intercambios punto a punto, que comportan siempre dos mensajes: una demanda del maestro y una respuesta del esclavo (puede ser simplemente un reconocimiento («acknowledge»).
- Mensajes difundidos. Estos consisten en una comunicación unidireccional del maestro a todos los esclavos. Este tipo de mensajes no tiene respuesta por parte de los esclavos y se suelen emplear para mandar datos comunes de configuración, reset, etc.

Protocolo

La codificación de datos dentro de la trama puede hacerse en modo ASCII o puramente binario, según el estándar RTU (Remote Transmission Unit). En cualquiera de los dos casos, cada mensaje obedece a una trama que contiene cuatro campos principales, según se muestra en la figura 1. La única diferencia

estriba en que la trama ASCII incluye un carácter de encabezamiento («:»=3AH) y los caracteres CR y LF al final del mensaje.

Pueden existir también diferencias en la forma de calcular el CRC, puesto que el formato RTU emplea una fórmula poli nómica en vez de la simple suma en módulo 16.

Con independencia de estos pequeños detalles, a continuación se da una breve descripción de cada uno de los campos del mensaje:

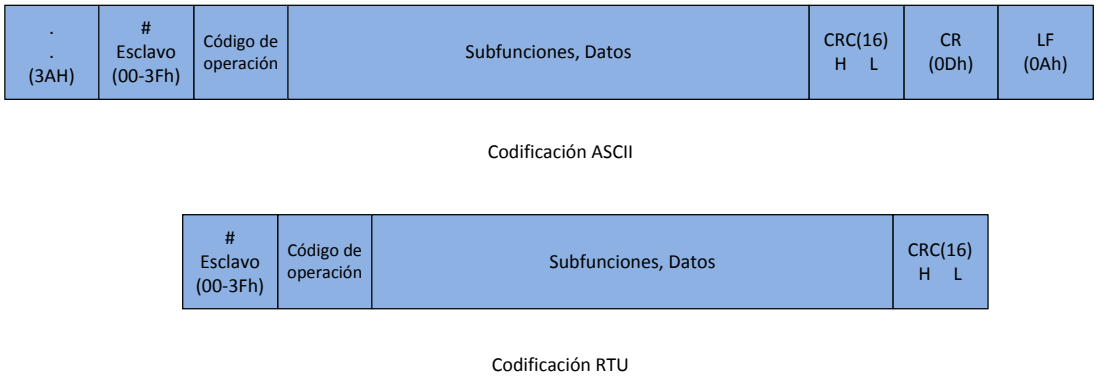


Figura 1.16. Codificación de la trama ASCII y RTU del protocolo Modbus

Número de esclavo (1 byte):

Permite direccionar un máximo de 63 esclavos con direcciones que van del 01H hasta 3FH. El número 00H se reserva para los mensajes difundidos.

Código de operación o función (1 byte):

Cada función permite transmitir datos u órdenes al esclavo. Existen dos tipos básicos de órdenes:

- Ordenes de lectura/escritura de datos en los registros o en la memoria del esclavo.
- Ordenes de control del esclavo y el propio sistema de comunicaciones (RUN/STOP, carga y descarga de programas, verificación de contadores de intercambio, etc.)

La tabla muestra la lista de funciones disponibles en el protocolo MODBUS con sus correspondientes códigos de operación.

FUNCIÓN	CÓDIGO hex	TAREA
00	00h	Control de estaciones esclavas
01	01h	Lectura de n bits de salida o internos
02	02h	Lectura de n bits de entradas
03	03h	Lectura de n palabras de salida o internos
04	04h	Lectura de n palabras de entradas
05	05h	Escritura de un bit
06	06h	Escritura de una palabra
07	07h	Lectura rápida de 8 bits
08	08h	Control de contadores de diagnóstico número 1 a 8
09	09h	No utilizado
10	0Ah	No utilizado
11	0Bh	Control del contador de diagnóstico numero 9
12	0Ch	No utilizado
13	0Dh	No utilizado
14	0Eh	No utilizado
15	0Fh	Escritura de n bits
16	10h	Escritura de n palabras

Tabla 1.8. Funciones protocolo Modbus

Campo de subfunciones/datos (n bytes):

Este campo suele contener, en primer lugar, los parámetros necesarios para ejecutar la función indicada por el byte anterior. Estos parámetros podrán ser códigos de subfunciones en el caso de órdenes de control (función 00H) o direcciones del primer bit o byte, número de bits o palabras a leer o escribir, valor del bit o palabra en caso de escritura, etc.

Palabra de control de errores (2 bytes):

En código ASCII, esta palabra es simplemente la suma de comprobación (‘*checksum*’) del mensaje en módulo 16 expresado en ASCII. En el caso de codificación RTU el CRC se calcula con una fórmula poli nómica según el algoritmo mostrado en la figura.

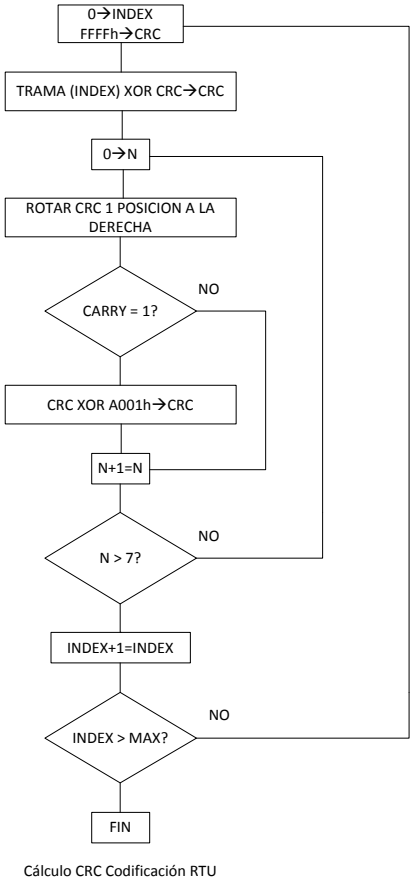


Figura 1.17. Diagrama de flujos del cálculo CRC Codificación RTU.

Descripción de las funciones del protocolo

Función 0:

Esta función permite ejecutar órdenes de control, tales como marcha, paro, carga y lectura de programas de usuario del autómata. Para codificar cada una de las citadas órdenes se emplean los cuatro primeros bytes del campo de datos. La trama resultante es la representada en la figura y la interpretación de los códigos de subfunción se especifica en la tabla.

En caso de las órdenes de marcha y paro, el campo de «información» de la trama representada en la figura está vacío y, por tanto, el mensaje se compone simplemente de 6 bytes de función más 2 bytes de CRC. La respuesta del esclavo a estas órdenes es un mensaje idéntico al enviado por el maestro. Cabe señalar, además, que después de un paro el autómata sólo acepta ejecutar subfunciones de la función 00H.

# Esclavo (00-3Fh)	00h	Código Subfunción SF0 SF1	Datos Subfunción D0 D1	Información	CRC(16) H L

Figura 1.18. Trama genérica de las subfunciones de control de esclavos (función 00h)

Código Sub función SF0	SF1	Datos Sub función D0	D1	Tarea
00h	00h	00h	00h	Paro del esclavo sin inicializar
00h	01h	00h	00h	Marcha del esclavo sin inicializar
00h	02h	00h	00h	Marcha e inicialización del esclavo
00h	03h	00h	XXh	Lectura de la secuencia XX de programa de usuario en el esclavo
00h	04h	YYh	XXh	Carga de una secuencia de programa de usuario en el esclavo. Petición: YY= Secuencia a cargar XX= Próxima Secuencia Respuesta: XX= Código de error, YY= 00

Tabla 1.9. Lista de sub funciones dentro de la función 0.

Funciones 1 y 2:

Lectura de bits del autómata. La trama es la indicada en la figura. La forma de direccionamiento de los bits es a base de dar la dirección de la palabra que los contiene y luego la posición del bit. Obsérvese también que la respuesta es dada siempre en octetos completos.

Petición del maestro

# Esclavo (00-3Fh)	01h Ó 02h	Dirección primer Bit PP PB	Número de Bits NN NN	CRC(16) H L

PPP: Dirección de la palabra (hex).
B: dirección del Bit dentro de la palabra 0 a F h

Respuesta del esclavo

# Esclavo (00-3Fh)	01h Ó 02h	# de Octetos leídos NN NN	Primer Octeto B7.....B0	Otros Octetos hasta maximo 256	CRC(16) H L

Petición y respuesta de las funciones 01h y 02h (Lectura de Bits)

Figura 1.19. Tramas de las funciones 1 y 2

Funciones 3 y 4:

Lectura de palabras del autómata. La trama es la indicada en la figura. Obsérvese que la petición indica el número de palabras a leer, mientras que en la respuesta se indica el número de octetos leídos.

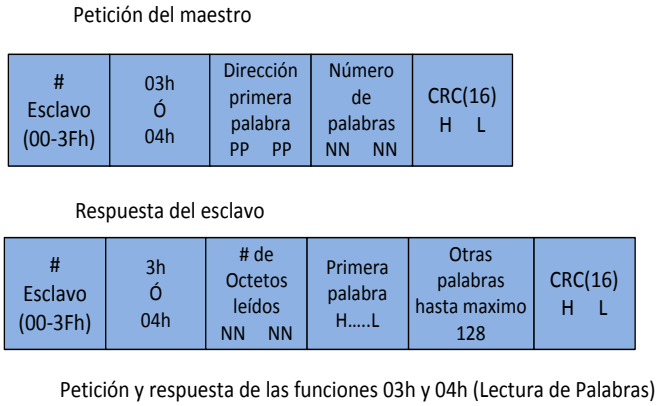


Figura 1.20. Tramas de las funciones 3 y 4

Función 5:

Escritura de un bit. La trama es la indiada en la figura. El direccionamiento del bit se efectúa tal como se ha indicado para las funciones 1 y 2.

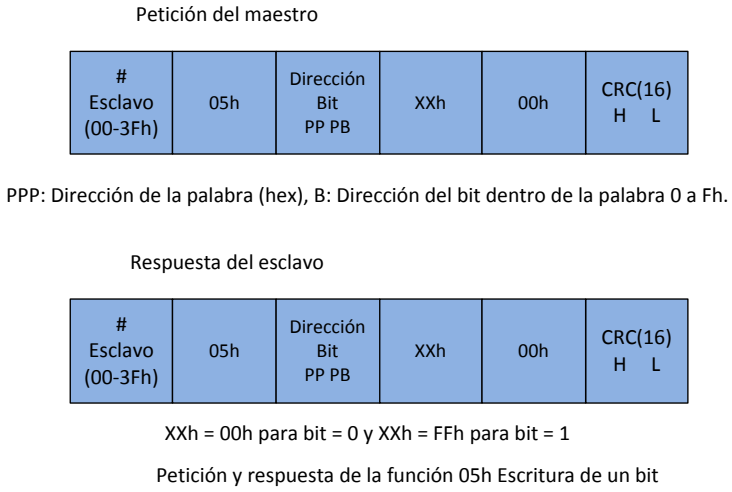


Figura 1.21. Trama de la función 5

Función 6:

Escritura de una palabra, la trama es la indicada en la figura.

Petición del maestro

# Esclavo (00-3Fh)	06h	Dirección Bit PP PP	Valor Palabra DD DD	CRC(16) H L

Respuesta del esclavo

# Esclavo (00-3Fh)	06h	Dirección Bit PP PP	Valor Palabra DD DD	CRC(16) H L

Petición y respuesta de la función 06h Escritura de una palabra.

Figura 1.22. Trama de la función 6.

Función 15:

Escritura de bits del autómata. La trama es la indicada en la figura. La forma de direccionamiento es análoga a la indicada para las funciones 1 y 2.

Petición del maestro

# Esclavo (00-3Fh)	0Fh	Dirección Primer Bit PP PB	Número de bits NN NN	Número de Octetos M	Valor de los bits 8xM valores	CRC(16) H L

Respuesta del esclavo

# Esclavo (00-3Fh)	0Fh	Dirección Primer Bit PP PB	Número de bits NN NN	CRC(16) H L

Petición y respuesta de la función 0Fh Escritura de bits.

Figura 1.23. Trama de la función 15

Función 16:

Escritura de palabras, la trama es la indicada en la figura.

Petición del maestro

# Esclavo (00-3Fh)	10h	Dirección Primera palabra PP PP	Número de palabras NN NN	Número de Octetos M	Valor de las palabras HL, HL, ...	CRC(16) H L

Respuesta del esclavo

# Esclavo (00-3Fh)	10h	Dirección Primera palabra PP PP	Número de palabras NN NN	CRC(16) H L

Petición y respuesta de la función 10h Escritura de palabras.

Figura 1.24. Trama de la función 16

Mensajes de error:

Puede ocurrir que un mensaje se interrumpa antes de terminar. Cada esclavo interpreta que el menaje ha terminado si transcurre un tiempo de silencio equivalente a 3,5 caracteres. Después de este tiempo el esclavo considera que el carácter siguiente es el campo de dirección de esclavo de un nuevo mensaje.

Cuando un esclavo recibe una trama incompleta o errónea desde el punto de vista lógico, envía un mensaje de error como respuesta, excepto en el caso de mensajes de difusión. La trama del mensaje de error es la indicada en la figura.

Respuesta del esclavo

# Esclavo (00-3Fh)	Código Función	Código Error	CRC(16) H L

Trama del mensaje de error

Figura 1.25. Trama del mensaje de error

Código Función = Código función recibido + 80H

Código Error = 01 Código de Función erróneo:

02 Dirección incorrecta

03 Datos incorrectos

06 Autómata ocupado

Si la estación maestra no recibe respuesta de un esclavo durante un tiempo superior a un límite establecido, declara el esclavo fuera de servicio, a pesar de que al cabo de un cierto número de ciclos hace nuevos intentos de conexión.

Nivel de Aplicación

Como se ha dicho a nivel general de buses de campo, el nivel de aplicación de MODBUS no está cubierto por un software estándar, sino que cada fabricante suele suministrar programas para controlar su propia red. No obstante, el nivel de concreción en la definición de las funciones permite al usuario la confección de software propio para gestionar cualquier red, incluso con productos de distintos fabricantes.

Modbus TCP/IP

MODBUS® TCP/IP es una variante o extensión del protocolo Modbus que permite utilizarlo sobre la capa de transporte TCP/IP. De este modo, Modbus-TCP se puede utilizar en Internet, de hecho, este fue uno de los objetivos que motivó su desarrollo (la especificación del protocolo se ha remitido a la IETF=Internet Engineering Task Force).

En la práctica, un dispositivo instalado en Europa podría ser direccionado desde EEUU o cualquier otra parte del mundo.

Las ventajas para los instaladores o empresas de automatización son innumerables:

- Realizar reparaciones o mantenimiento remoto desde la oficina utilizando un PC, reduciendo así los costes y mejorando el servicio al cliente.
- El ingeniero de mantenimiento puede entrar al sistema de control de la planta desde su casa, evitando desplazamientos.
- Permite realizar la gestión de sistemas distribuidos geográficamente mediante el empleo de las tecnologías de Internet/Intranet actualmente disponibles.
- MODBUS® TCP/IP se ha convertido en un estándar industrial *de facto* debido a su simplicidad, bajo coste, necesidades mínimas en cuanto a

componentes de hardware, y sobre todo a que se trata de un protocolo abierto.

En la actualidad hay cientos de dispositivos MODBUS® TCP/IP disponibles en el mercado.

Se emplea para intercambiar información entre dispositivos, así como monitorizarlos y gestionarlos. También se emplea para la gestión de entradas/salida distribuidas, siendo el protocolo más popular entre los fabricantes de este tipo de componentes.

La combinación de una red física versátil y escalable como Ethernet con el estándar universal de interredes TCP/IP y una representación de datos independiente de fabricante, como MODBUS®, proporciona una red abierta y accesible para el intercambio de datos de proceso.

El protocolo Modbus TCP

Modbus/TCP simplemente encapsula una trama Modbus en un segmento TCP. TCP proporciona un servicio orientado a conexión fiable, lo que significa que toda consulta espera una respuesta.

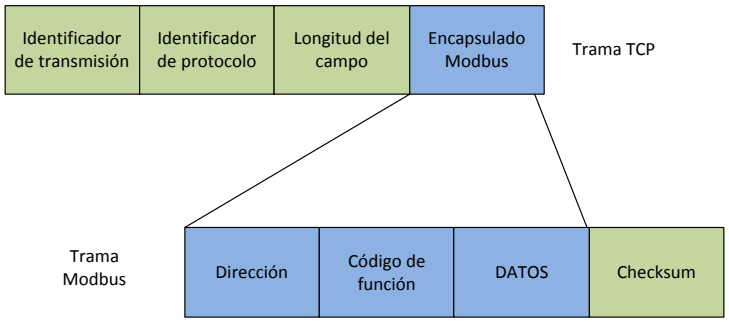


Figura 1.26. Encapsulado de la trama de Modbus en TCP

Esta técnica de consulta/respuesta encaja perfectamente con la naturaleza Maestro/Esclavo de Modbus, añadido a la ventaja del determinismo que las redes Ethernet conmutadas ofrecen a los usuarios en la industria. El empleo del

protocolo abierto Modbus con TCP proporciona una solución para la gestión desde unos pocos a decenas de miles de nodos.

Prestaciones de un sistema MODBUS TCP/IP

Las prestaciones dependen básicamente de la red y el hardware. Si se usa MODBUS® TCP/IP sobre Internet, las prestaciones serán las correspondientes a tiempos de respuesta en Internet, que no siempre serán las deseables para un sistema de control. Sin embargo pueden ser suficientes para la comunicación destinada a depuración y mantenimiento, evitando así desplazamientos al lugar de la instalación.

Si disponemos de una Intranet de altas prestaciones con conmutadores Ethernet de alta velocidad, la situación es totalmente diferente.

En teoría, MODBUS® TCP/IP, transporta datos hasta 250/(250+70+70) o alrededor de un 60% de eficiencia cuando se trasfieren registros en bloque, y puesto que 10 Base T proporciona unos 1.25 Mbps de datos, la velocidad de transferencia de información útil será:

1.25M / 2 * 60% = 360000 registros por Segundo En 100BaseT la velocidad es 10 veces mayor.

Esto suponiendo que se están empleando dispositivos que pueden dar servicio en la red Ehernet aprovechando todo el ancho de banda disponible.

En los ensayos prácticos realizados por by Schneider Automation utilizando un PLC Ethernet Momentum TM con entradas/salidas Ethernet, demostró que se podían escanear hasta 4000 bloques I/O por segundo, cada uno con hasta 16 I/O analógicas de 12-bits o 32 I/O digitales (se pueden actualizar 4 bases por milisegundo). Aunque estos resultados están por debajo del límite teórico calculado anteriormente, pero debemos recordar que el dispositivo se probó con una CPU de baja velocidad (80186 a 50MHz con 3 MIPS). Además, el abaratamiento de los ordenadores personales y el desarrollo de redes Ethernet cada vez más rápidas, permite elevar las velocidades de funcionamiento, a diferencia de otros buses que están inherentemente limitados una sola velocidad.

Cómo podemos comunicar dispositivos MODBUS existentes sobre MODBUS TCP/IP?

Puesto que MODBUS® TCP/IP es simplemente un protocolo MODBUS® encapsulado en TCP, es muy sencillo comunicar dispositivos MODBUS® existentes sobre MODBUS® TCP/IP. Para ello se requiere una pasarela que convierta el protocolo MODBUS a MODBUS TCP/IP.

1.5. Los Microcontroladores Pic

1.5.1. Descripción.

Entre los distintos fabricantes de microcontroladores nos decantamos por Microchip Technology. Los microcontroladores PIC de Microchip están muy extendidos actualmente en el mercado gracias a su gran variedad y bajo coste. Otra razón de su éxito es su utilización, ya que una vez se ha aprendido a utilizar uno, conociendo su arquitectura y juego de instrucciones, es muy fácil emplear otro modelo diferente.

Para clasificar los distintos tipos de PIC se utiliza una nomenclatura que atiende a sus características, tanto físicas, indicando el número de pines que posee el encapsulado del PIC, como eléctricas o de procesamiento, ya que podemos indicar el tamaño de la palabra de memoria que es capaz de manejar el procesador UAL (Unidad Aritmético Lógica), o lo que es lo mismo, el ancho o número de bits de la palabra en la memoria de datos (registros en la terminología usada por Microchip).

La variedad de PICs disponibles en el mercado es bastante amplia, desde PIC pequeños de 6-pin y 8-bits con sólo 16 bytes de memoria de datos con los que únicamente podemos fabricar una I/O digital básica, hasta dispositivos de 100-pin y 16-bit, con 30 kilobytes de memoria de datos, diseñados para ofrecer posibilidades prácticamente infinitas para el procesamiento digital de la señal.

Los PIC menos prestigiosos o con menos prestaciones, aquellos que operan con palabras de datos de 8 bits, están divididos en 3 familias que se diferencian básicamente por su estructura:

Baseline (12-bit instuctions)

Estos dispositivos tienen muchas limitaciones comparándolos con el resto de las opciones disponibles en el mercado, ya que, por ejemplo, no presentan mecanismos de detección de eventos por interrupciones por lo que su manejo es más complejo; sin embargo poseen un diseño muy simple y son bastante sencillos a la hora de trabajar con ellos en ensamblador. Algunos de estos dispositivos que podemos encontrar hoy en día serían los incluidos en la serie 10F de 6-pines, o el 12F509 de 8-pines e incluso el dispositivo 16F506 de 14-pines.

Midrange (14-bit instructions)

Aparecen cuando el diseño básico de la arquitectura convencional comenzó a desarrollarse, añadiendo el soporte necesario para poder trabajar con interrupciones, más memoria, temporizadores, algunos periféricos, incluso PWM, controles de motores, choppers para sensores en ambientes ruidosos y algunas otras aplicaciones. También se añadió el soporte para la transferencia de datos en serie, interfaces I2C y SPI y controladores con pantalla LCD. Algunos ejemplos de esta clase de dispositivos los encontramos en las series 12F ó 16F, como los dispositivos: 12F629 de 8-pines, 16F629 de 20-pines o el 16F917 de 40-pines.

High-end (16-bits instructions)

Se conoce con este nombre a los dispositivos de la serie 18F, que superan muchas de las limitaciones que presentan los dispositivos de las dos familias anteriores. Provistos de más memoria (más de 128k de memoria de programa y aproximadamente 4k para memoria de datos) y de periféricos mucho más modernos incluyendo comunicación USB o Ethernet. La arquitectura que presentan los dispositivos de la familia 18F está desarrollada para soportar la programación con lenguaje C. Ejemplos de estos dispositivos serían los de 18-pines, como el 18F1220, de 28-pines: 18F2455 y de incluso con 80-pines, dispositivo 18F8520.

Es importante recordar que los dispositivos de la familia 18F tienen 16-bit para definir las instrucciones de programa que operan con datos de anchura 8 bits, por lo que serán considerados dispositivos de 8-bits. Existen otros microcontroladores o dispositivos de 16-bit, como los pertenecientes a la familia PIC24F, pero no hablaremos de ellos en el capítulo.

1.5.1.1. La familia 16FXXXX

Vamos a describir la arquitectura en general de la familia de microcontroladores 16F que son de 8 bits (bus de datos) que tienen las siguientes características generales que los distinguen de otras familias:

- Arquitectura Harvard.
- Tecnología RISC.
- Tecnología CMOS.

Las características que veremos a continuación son compartidas en la mayoría de los puntos por los microcontroladores que de la familia 16F en especia por los microcontroladores que hemos utilizado en el módulo de control y adquisición de datos como son los PIC's 16F887, 16F876A Y 16F819 por lo que se generalizara la explicación para dichos microcontroladores.

Estas características se conjugan para lograr un dispositivo altamente eficiente en el uso de la memoria de datos y programa y por lo tanto en la velocidad de ejecución.

1.5.1.2. La familia 18FXXXX

El PIC utilizado, está dentro de la familia PIC18 de microcontroladores, que ofrece un alto rendimiento computacional a un precio económico, con una ostentosa mejora con respecto a las anteriores, en cuanto a la memoria de programa.

La familia PIC18Fxx incluye una serie de características que hacen que se pueda reducir el consumo de energía durante las operaciones. Estas pueden ser:

- Modos RUN alternativos: Bloqueando el controlador desde el Timer1 o el oscilador interno, el consumo puede ser reducido hasta un 90%

- Modos libres múltiples: El núcleo de procesador puede estar parado mientras los periféricos están activos.
- Consumo bajo en módulos de Switch: el requisito de consumo para ambos Timer1 y el Timer del guardián está minimizado.
- Memoria resistente: Las celdas de las memorias de programa y de datos han sido mejoradas para aguantar entre 100.000 y 1.000.000 de escrituras y borrados, pudiendo llegar a durar hasta más de 40 años.
- Conjunto de instrucciones extendido: Se han creado 8 nuevas instrucciones para manejar direccionamientos indexados para poder posibilitar la utilización de lenguajes de programación como “C”.
- USART direccionable mejorado: este módulo de comunicación serie es capaz de operar a nivel RS - 232 y suministra el soporte para el protocolo LIN de bus.

1.6. Circuitos integrados especiales.

A continuación se describe el funcionamiento y modos de comunicación de los circuitos integrados especiales utilizado en el módulo de control y adquisición de datos como son los siguientes:

- MCP4822
- MCP3202
- MCP3551
- MCP9800
- MCP3421
- MCP1702
- AD822

1.6.1. MCP4822

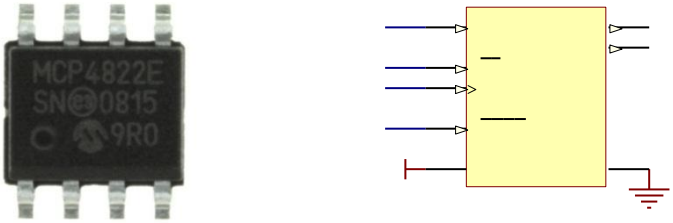


Figura 1.27. Empaquetado smd y esquemático del CI MCP4822

1.6.1.1. Descripción

Estos circuitos integrados funcionan en un rango de 2.7V-5.5V, con una resolución de conversión digital análoga de 12-bits, salida opcional 2x-buffer y de interfaz periférica serial (SPI ™).

El circuito integrado proporciona una alta precisión y el rendimiento con bajo ruido para aplicaciones industriales donde se requiere de calibración o compensación de las señales (como la temperatura, presión y humedad).

Los dispositivos utilizan una arquitectura de cadena resistiva, bajo coeficiente de relación entre la temperatura y la métrica rápido tiempo de estabilización. La familia MCP482X incluye registros de doble búfer de lectura que permita actualizaciones simultáneas con el pin LDAC.MCP3202

1.6.2. MCP3202

1.6.2.1. Descripción

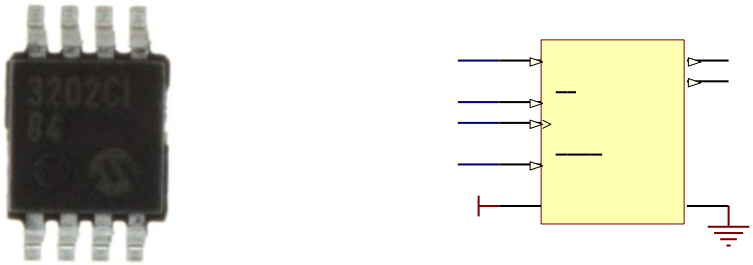


Figura 1.28. Empaquetado smd y esquemático del CI MCP3202

Este circuito integrado es un conversor análogo digita con una resolución de 12 bit. El IC es completamente programable para dar una sola lectura con un resultado seudodiferencial o dos entradas de una sola terminal. La comunicación con el dispositivo se realiza mediante una sencilla interfaz en serie compatible con el protocolo SPI. El dispositivo es capaz de realizar tasas de conversión de hasta 100 ksps a 5V y 50 ksps a 2.7V, Los rango de voltaje a los que puede trabajar el IC van desde 2.7V a 5V. El MCP3202 se ofrece en 8-pin SOIC, PDIP, SOIC y 150 mil paquetes de SOIC.

1.6.3. MCP3551

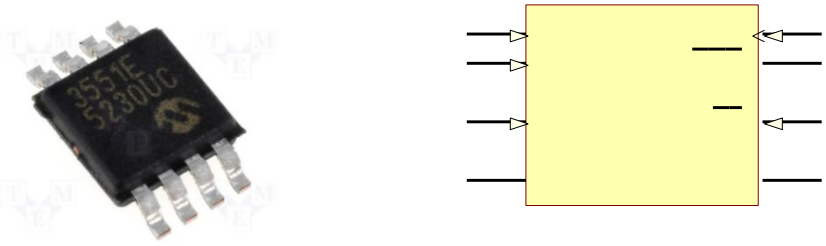


Figura 1.29. Empaquetado smd y esquemático del CI MCP3202

1.6.3.1. Descripción

Este CI tiene un rango de alimentación de 2.7V a 5.5V, en un conversor análogo digital con una resolución de 22-bit Delta-Sigma. Los dispositivos ofrecen salida de ruido tan bajo como 2,5 µVRMS, con un error total no ajustado de 10 ppm. Los dispositivos MCP3551 / 3 proporciona una alta precisión y el rendimiento de bajo ruido para aplicaciones en las mediciones de los sensores (tales como presión, temperatura y humedad). Esto lo hace ideal para el uso en aplicaciones prácticas por su precisión al momento de la conversión y pro los pocos componentes exteriores que son necesarios.

Esta línea de productos tiene entradas analógicas completamente diferenciales, por lo que es compatible con una amplia variedad de sensores, control industrial o aplicaciones de control de procesos.

Los dispositivos MCP3551 / 3 operar desde -40 ° C a +125 ° C y están disponibles en el ahorro de espacio de 8 pines en encapsulados de tipo DIP y SOIC.

1.6.3.2. Conversión Delta Sigma

La conversión Delta-Sigma (ΔΣ) es un método para codificación de alta resolución o señales analógicas en señales de baja resolución digitales. La conversión se realiza mediante realimentación de error, cuando la diferencia entre las dos señales se mide y se utiliza para mejorar la conversión.

La señal de baja resolución normalmente cambia más rápidamente que la señal de alta resolución y puede ser filtrado para recuperar la señal de alta resolución con poca o ninguna pérdida de fidelidad.

Esta técnica ha encontrado un uso creciente en los modernos componentes electrónicos tales como analógico-digital (ADC) y digital-analógicos convertidores (DACS), los sintetizadores de frecuencia, fuentes de alimentación conmutadas fuentes de alimentación y los controladores de motor.

1.6.4. MCP3421

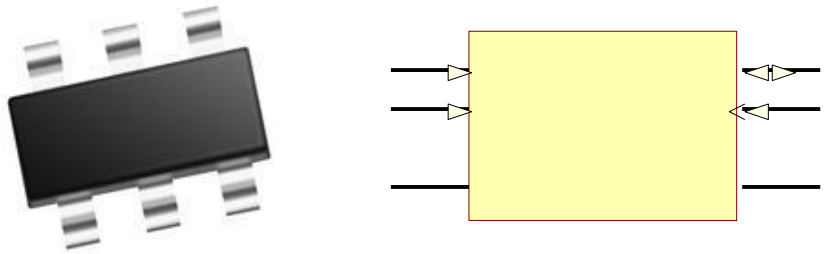


Figura 1.30. Empaquetado smd y esquemático del CI MCP3202

1.6.4.1. Descripción

El MCP3421 es un solo canal de bajo ruido, alta precisión convertidor ΔΣ análogo digital con entradas diferenciales y hasta 18 bits de resolución en un pequeño encapsulado SOT-23-6. La tensión de la precisión en la placa de referencia 2.048V permite un rango de entrada de ± 2.048V diferencial (tensión Δ = 4.096V). El dispositivo utiliza dos con interfaz I2C y funciona con una sola fuente de alimentación de 2.7V a 5.5V.

El dispositivo MCP3421 realiza la conversión a un ritmo de 3.75, 15, 60 o 240 muestras por segundo (MPS) en función de la configuración de los bits de control sobre el bus I2C. Este dispositivo tiene un amplificador interno de ganancia programable (PGA). El usuario puede seleccionar la ganancia en x1, x2, x4, x8. Esto permite que el dispositivo MCP3421 realice las conversiones con mayor precisión. El dispositivo tiene dos modos de trabajo: (a) el modo continuo y (b) el modo One-Shot. En el modo One-Shot, el dispositivo entra en un modo de baja corriente de espera automáticamente después de una conversión. Esto reduce el consumo de corriente en gran medida durante los períodos de inactividad.

El dispositivo MCP3421 se puede utilizar para varias aplicaciones de alta precisión de conversión analógica donde la simplicidad de diseño, bajo consumo y pequeño tamaño son las consideraciones principales.

1.6.5. MCP4822

Figura 1.31. Empaquetado smd y esquemático del CI MCP3202

1.6.5.1. Descripción

El MCP1702 viene de una familia CMOS de baja caída (LDO), los reguladores positivos de tensión que puede entregar hasta 250 mA de corriente mientras que consume sólo 2,0 mA de corriente de reposo (típico). La operación de entrada de rango de voltaje se especifica hasta 13,2 V, lo que es ideal para los iones de litio (una, dos o tres células).

El MCP1702 es capaz de entregar 250 mA con un diferencial de tensión de entrada a salida de 650 mV. La baja caída de voltaje extiende la vida útil de la batería. También permite que las corrientes de alta en paquetes pequeños cuando se opera con un mínimo de VIN - VOUT diferenciales.

El MCP1702 tiene un rendimiento de muy baja tolerancia a la regulación de voltaje de ± 0,5% (típico) y la regulación de muy buena línea de ± 0,2% / V. La salida LDO

es estable cuando se utiliza sólo 1 uF de capacitancia de salida. El LDO MCP1702 también incorpora protección contra cortocircuitos y apagado térmico para garantizar la máxima fiabilidad.

1.6.5.2. Circuito típico de aplicación.

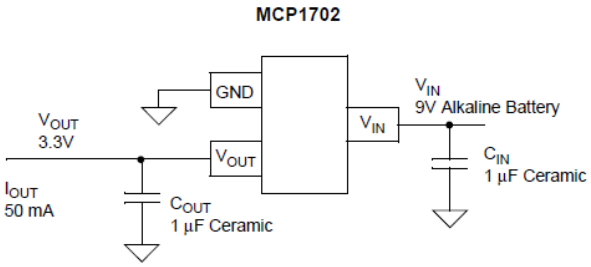


Figura 1.32. Circuito típico de aplicación del MCP1702

1.6.6. AD822

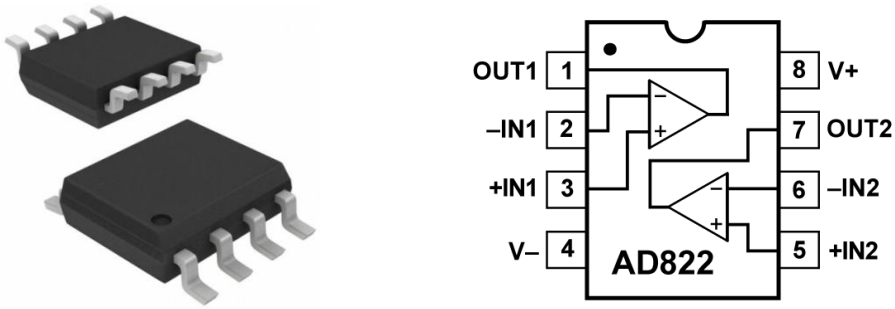


Figura 1.33. Empaquetado smd y esquemático del CI MCP3202

1.6.6.1. Descripción

El AD822 es un operacional de muy bajo offset (800 µV de voltaje de offset de entrada y 25 pA de corriente de polarización), diseñado para aplicaciones de instrumentación. Puede trabajar con fuente sencilla (5 a 30 V) o doble (± 2,5 a ± 15 V) y presenta una salida rail-to-rail (menos de 10 mV de diferencia respecto a la fuente).

1.7. SOFTWARE DE DESARROLLO

1.7.1. LabVIEW

(Laboratory Virtual Instrument Engineering Workbench) es un lenguaje de programación gráfico para el diseño de sistemas de adquisición de datos, instrumentación y control. Labview permite diseñar interfaces de usuario mediante una consola interactivo basado en software. Usted puede diseñar especificando su sistema funcional, su diagrama de bloques o una notación de diseño de ingeniería. Labview es a la vez compatible con herramientas de desarrollo similares y puede trabajar con programas de otra área de aplicación, como por ejemplo Matlab. Tiene la ventaja de que permite una fácil integración con hardware, específicamente con tarjetas de medición, adquisición y procesamiento de datos (incluyendo adquisición de imágenes).

La programación G (gráfica) de Labview consta de un panel frontal y un panel de código como se mencionó antes. En el panel frontal es donde se diseña la interface de usuario y se ubican los controles e indicadores. En el panel de código se encuentran las funciones. Cada control que se utiliza en la interfaz tiene una representación en el panel de código, igualmente los indicadores necesarios para entregar la información procesada al usuario tienen un icono que los identifica en el panel de código o de programación. Los controles pueden ser booleanos, numéricos, strings, un arreglo matricial de estos o una combinación de los anteriores; y los indicadores pueden ser como para el caso de controles pero pudiéndolos visualizar como tablas, gráficos en 2D o 3D, browser, entre otros.

Las funciones pueden ser VIs prediseñados y que pueden ser reutilizados en cualquier aplicación, estos bloques funcionales constan de entradas y salidas, igual que en un lenguaje de programación estándar las funciones procesan las entradas y entregan una o varias salidas, estos VI pueden también estar conformados de otros subVIs y así sucesivamente, de esta forma se pueden representar como un árbol genealógico donde un VI se relaciona o depende de varios SubVIs. Labview tiene VIs de adquisición de datos e imágenes, de comunicaciones, de procesamiento digital de señales.

1.7.2. NI – VISA DE LABVIEW

1.7.2.1. SOBRE NI - VISA

VISA es un API de alto nivel utilizado para comunicarse con buses de instrumentación. Es independiente de la plataforma, del bus y del entorno. En otras palabras, la misma API se utiliza sin importar si un programa está creado para comunicarse con un dispositivo USB con LabVIEW en una máquina que ejecuta Windows 2000, o con un dispositivo GPIB con C en una máquina que ejecuta Mac OS X.

USB es un bus de comunicación basado en mensajes. Esto significa que una PC y un dispositivo USB se comunican enviando comandos y datos a través del bus en forma de texto o datos binarios. Cada dispositivo USB tiene su propio conjunto de comandos. Usted puede utilizar funciones de Lectura y Escritura NI-VISA para enviar estos comandos a un instrumento y leer la respuesta del mismo. Contacte al fabricante de su instrumento para consultar la lista de comandos de su instrumento.

A partir de la versión 3.0, NI-VISA utiliza comunicación por USB. Se pueden utilizar dos clases de recursos VISA: USB INTR y USB RAW: Los dispositivos USB que cumplen con el protocolo USB Test and Measurement Class (USBTMC) utilizan la clase de recursos USB INSTR. Los dispositivos USBTMC cumplen con un protocolo que la clase de recursos USB INSTR de VISA puede entender. No se necesita ninguna configuración para comunicarse con un dispositivo USBTMC. Para comunicarse con instrumento USBTMC, consulte la sección 3. Para obtener más información sobre la especificación USBTMC, consulte la página web del foro de implementadores de USB, cuyo enlace se encuentra al final de esta guía.

Los instrumentos USB RAW son todos aquellos de USB que no cumplen con la especificación USBTMC. Si usted utiliza un dispositivo USB RAW, siga las instrucciones en la sección 2 para configurar NI-VISA para que controle su dispositivo. Contacte al fabricante de su instrumento para obtener detalles sobre el protocolo de comunicación y el juego de comandos que utiliza su instrumento.

1.7.2.2. Comunicación VISA USB RAW

NI-VISA de National Instruments soporta dos formas de comunicación por USB: **USBTMC** y **RAW** mode. Este documento se enfoca en la información para el modo VISA USB RAW.

El modo USB RAW le da un acceso de bajo nivel a los mecanismos de comunicación de USB. Como VISA únicamente expone los mecanismos ya creados por el protocolo, cuando se utiliza el modo NI-VISA USB RAW se soporta la transferencia de datos como **Interrupt**, **Bulk** and **Control**.

1.8. El software PCWHD Compiler, y la programación en C para el Microcontrolador.

1.8.1. Características del lenguaje C para este compilador

El lenguaje C estándar es independiente de cualquier plataforma. Sin embargo, para la programación de microcontroladores es necesario disponer de determinados comandos que se refieran a partes específicas de su hardware, como el acceso a memoria, temporizadores, etc. Por este motivo, además de los comandos, funciones y datos del lenguaje ANSI C, el compilador PCW incluye bibliotecas que incorporan determinados comandos que no son estándar, sino específicos para la familia de microcontroladores PIC. Éstos son básicamente de dos tipos: directivas del preprocesador y funciones pre compiladas

1.8.1.1. Directivas del preprocesador.

Todas las directivas del pre-procesador comienzan con el carácter # seguido por un comando específico. Algunas de estas directivas son extensiones del C estándar. El C proporciona una directiva del preprocesador, *que los compiladores aceptan*, y que permite ignorar o actuar sobre los datos que siguen.

Directivas derivadas del estándar de C.

Permiten, entre otras funciones, un control básico del código y del flujo en el proceso de compilación:

#DEFINE:

Sirve para definir una variable con un determinado nombre, asignarle un valor constante o un asignarle un pin de entrada o salida.

#IFDEF

Sirve para chequear si se ha definido una variable o se ha dado un valor a una variable previamente con el #DEFINE.

#ERROR

Esta directiva para el compilador y emite el mensaje que se incluye a continuación (en la misma línea) de la propia directiva. El mensaje puede incluir macros. También puede utilizarse para alertar al usuario de una situación anómala en tiempo de compilación.

#INCLUDE

Esta directiva hace que el compilador incluya en el fichero fuente el texto que contiene el archivo especificado en <Nombre_Fichero>.

Si el nombre del fichero se incluye entre los símbolos '< >' el compilador busca el fichero en el directorio INCLUDE. Si se pone entre comillas dobles " " el compilador busca primero en el directorio actual o directorio de trabajo y si no lo encuentra, entonces lo busca en los directorios INCLUDE del compilador.

#LIST

Guarda el código fuente en el archivo .LIST.

#NOLIST

NO guarda el código fuente en el archivo .LIST.

#PRAGMA

Esta directiva se usa para mantener compatibilidad entre los compiladores de C. El compilador aceptará esta directiva antes de cualquier otro comando del preprocesador.

#UNDEF

El identificador ID no tendrá ya significando para el pre-procesador.

Directivas asociadas a las bibliotecas precompiladas:

Proporcionan al compilador información relacionada con estas bibliotecas:

#USE DELAY (CLOCK=frecuencia)

Esta directiva indica al compilador la frecuencia del procesador, en ciclos por segundo, a la vez que habilita el uso de las funciones DELAY_MS() y DELAY_US().

Opcionalmente podemos usar la función restart_WDT() para que el compilador reinicie el WDT durante el retardo.

#USE FAST_IO

Esta directiva afecta al código que el compilador generará para las instrucciones de entrada y salida. Este método rápido de hacer I/O ocasiona que el compilador realice I/O sin programar el registro de dirección. El puerto puede ser A-G.

#USE FIXED_IO

Esta directiva afecta al código que el compilador generará para las instrucciones de entrada y salida. El método fijo de hacer I/O causará que el compilador genere código para hacer que un pin de I/O sea entrada o salida cada vez que se utiliza. Esto ahorra el byte de RAM usado en I/O normal.

#USE I2C

La librería I2C contiene funciones para implementar un bus I2C. La directiva #USE I2C permanece efectiva para las funciones I2C_START, I2C_STOP, I2C_READ, I2C_WRITE e I2C_POLL hasta que se encuentre otra directiva #USE I2C.

Se generan las funciones software a menos que se especifique la opción NO FORCE_SW. El modo SLAVE sólo debe usarse con las funciones SSP.

Las opciones son:

MASTER	Establece Modo Maestro
SLAVE	Establece Modo Esclavo
SCL=PIN	Especifica PIN de SCL
SDA=PIN	Especifica PIN de SDA
ADRESS=NN	Establece la dirección I2C esclavo
FAST	Establece modo rápido de I2C
SLOW	Establece modo lento de I2C
RESTART_WDT	Reinicia el WDT mientras espera en I2C Read
NOFORCE_SW	Usa funciones hardware I2C

Tabla 1.10. Subfunciones de la directiva USE I2C en CCS

#USE RS232

Esta directiva le dice al compilador la velocidad en baudios y los pines utilizados para la I/O serie. Esta directiva tiene efecto hasta que se encuentra otra directiva RS232.

La directiva **#USE DELAY** debe aparecer antes de utilizar **#USE RS232**. Esta directiva habilita el uso de funciones tales como GETCH, PUTCHAR y PRINTF. Si la I/O no es estándar es preciso poner las directivas **FIXED_IO** o **FAST_IO** delante de **#USE RS232**

OPCIONES:

RESTART_WDT	Hace que GETC() ponga a cero el WDT mientras espera un carácter.
INVERT	Invierte la polaridad de los pines serie (normalmente no es necesario con el convertidor de nivel, como el MAX232). No puede usarse con el SCI interno.
PARITY = X	Donde X puede ser None = N, Even = E y Odd = O.
BITS = X	Donde X es 5-9 (no puede usarse 5-7 con el SCI).
FLOAT_HIGH	Se utiliza para las salidas de colector abierto.
ERRORS	Indica al compilador que guarde los errores recibidos en la variable

	RS232_ERRORS para restablecerlos cuando se producen.
BRGH1OK	Permite velocidades de transmisión bajas en chips (uC's, memorias, etc) que tienen problemas de transmisión. Cuando utilizamos dispositivos con SCI y se especifican los pines SCI, entonces se usará el SCI. Si no se puede alcanzar una tasa de baudios dentro del 3% del valor deseado utilizando la frecuencia de reloj actual, se generará un error.
ENABLE = PIN	El pin especificado estará a nivel alto durante la transmisión.
FORCE_SW	Usa una UART software en lugar del hardware aun cuando se especifican los pines del hardware. La definición de RS232_ERRORS es como sigue: **Sin UART:** El bit 7 es el 9º bit para el modo de datos de 9 bit. El bit 6 a nivel alto indica un fallo en el modo flotante alto. **Con UART:** Usado sólo para conseguir: Copia del registro RCSTA, excepto: que el bit 0 se usa para indicar un error de paridad.

Tabla 1.11. Subfunciones de la directiva USE RS232 en CCS

#USE STANDARD_IO

Esta directiva afecta al código que el compilador genera para las instrucciones de entrada y salida. El método estándar de hacer I/O causará que el compilador genere código para hacer que un pin de I/O sea entrada o salida cada vez que se utiliza. En los procesadores de la serie 5X esto necesita un byte de RAM para cada puerto establecido como I/O estándar.

Directivas relacionadas con la especificación del dispositivo:

Por un lado, para definir los mapas de memoria y el juego de instrucciones, y por otro, incluir información necesaria para la programación del dispositivo en los ficheros de salida de la compilación:

#DEVICE CHIP

Esta directiva define al compilador, la arquitectura hardware utilizada. Esto determina la memoria RAM y ROM así como el juego de instrucciones. Para los

chips (uC's, memorias, etc) con más de 256 bytes de RAM se puede seleccionar entre punteros de 8 o 16 bits. Para usar punteros de 16 bits hay que añadir *=16 después del nombre del chip (uC, memoria, ...) o en una nueva línea después de la declaración del chip. Se puede obtener información sobre un dispositivo con el programa PICCHIPS.

#ID

Esta directiva define la palabra de ID identificación que se grabará en el chip (uC, memoria, etc). Esta directiva no afecta a la compilación pero la información se pone en el archivo de salida. La primera sintaxis necesita un número de 16-bit y pondrá un nible en cada una de las cuatro palabras del ID. La segunda sintaxis especifica el valor exacto en cada una de las cuatro palabras del ID. Cuando se especifica "nombre_archivo" el ID se lee del archivo indicado; su formato debe ser texto simple con un CR/LF al final. La palabra CHECKSUM indica que el checksum del dispositivo debe tomarse como el ID.

#FUSES

Esta directiva define qué fusibles deben activarse en el dispositivo cuando se programe. Esta directiva no afecta a la compilación; sin embargo, esta información se pone en el archivo de salida. Si los fusibles necesitan estar en formato Parallax, hay que agregar PAR en opciones. Utilizar la utilidad PICCHIPS para determinar qué opciones son válidas para cada dispositivo. La opción SWAP tiene la función especial de intercambiar, los bytes alto y bajo de los datos que no son parte del programa, en el archivo Hex. Esta información es necesaria para algunos programadores de dispositivos. Algunas de las opciones más usadas son:

LP, XT, HS, RC
WDT, NOWDT
PROTECT, NOPROTECT
PUT, NOPUT (Power Up Timer)
BROWNOUT, NOBROWNOUT
PAR (Parallax Format Fuses)
SWAP

Directivas de cualificación de funciones:

Para identificar características especiales de una función:

#INLINE

Esta directiva le dice al compilador que el procedimiento que sigue a la directiva será llevado a cabo EN LÍNEA. Esto causará una copia del código que será puesto en cualquier parte donde se llame al procedimiento. Esto es útil para ahorrar espacio de la pila (stack) y aumentar la velocidad.

Sin esta directiva es el compilador quien decidirá cuando es mejor hacer los procedimientos EN LÍNEA.

#INT_DEFAULT

La función que sigue a la directiva será llamada si el PIC activa una interrupción y ninguno de los flags de interrupción está activo.

#INT_GLOBAL

La función que sigue a esta directiva remplaza al distribuidor de interrupciones del compilador; dicha función toma el control de las interrupciones y el compilador no salva ningún registro. Normalmente no es necesario usar esto y debe tratarse con gran prudencia.

#INT_xxxxx

Estas directivas especifican que la función que le sigue es una función de interrupción.

Las funciones de interrupción no pueden tener ningún parámetro. Como es natural, no todas las directivas pueden usarse con todos los dispositivos. Las directivas de este tipo que disponemos son:

#INT_EXT	INTERRUPCIÓN EXTERNA
#INT_RTCC	DESBORDAMIENTO DEL TIMER0(RTCC)
#INT_RB	CAMBIO EN UNO DE LOS PINES B4,B5,B6,B7
#INT_AD	CONVERSOR A/D
#INT_EEPROM	ESCRITURA EN LA EEPROM COMPLETADA
#INT_TIMER1	DESBORDAMIENTO DEL TIMER1
#INT_TIMER2	DESBORDAMIENTO DEL TIMER2

#INT_CP1	MODO CAPTURA DE DATOS POR CCP1
#INT_CCP2	MODO CAPTURA DE DATOS POR CCP2
#INT_SSP	PUERTO DE SERIE INTELIGENTE (SPI, I2C)
#INT_PSP	PUERTO PARALELO
#INT_TBE	SCI DATO SERIE TRANSMITIDO
#INT_RDA	SCI DATO SERIE RECIBIDO
#INT_COMP	COMPARADOR DE INTERRUPCIONES
#INT_ADOF	DESBORDAMIENTO DEL A/DC DEL PIC 14000
#INT_RC	CAMBIO EN UN PIN Cx
#INT_I2C	I2C DEL 14000
#INT_BUTTON	PULSADOR DEL 14000
#INT_LCD	LCD 92x

El compilador salta a la función de interrupción cuando se detecta una interrupción. Es el propio compilador el encargado de generar el código para guardar y restaurar el estado del procesador.
También es el compilador quien borrará la interrupción (el flag). Sin embargo, nuestro programa es el encargado de llamar a la función ENABLE_INTERRUPT () para activar previamente la interrupción junto con el señalizador (flag) global de interrupciones.

#SEPARATE

#SEPARATE le dice al compilador que el procedimiento o función que sigue a la directiva será llevado a cabo por SEPARADO. Esto es útil para evitar que el compilador haga automáticamente un procedimiento en línea (INLINE).
Esto ahorra memoria ROM pero usa más espacio de la pila. El compilador hará todos los procedimientos #SEPARATE, separados, tal como se solicita, aun cuando no haya bastante pila.

Directivas de control del compilador:

Para definir opciones referidas a la compilación del código del programa:

#CASE

Hace que el compilador diferencie entre mayúsculas y minúsculas. Por defecto el compilador hace esta distinción.

#OPT N

Esta directiva sólo se usa con el paquete PCW y, establece el nivel de optimización. Se aplica al programa entero y puede aparecer en cualquier parte del archivo fuente. El nivel de optimización 5 es el nivel para los compiladores DOS. El valor por defecto para el compilador PCW es 9 que proporciona una optimización total.

#PRIORITY

Esta directiva se usa para establecer la prioridad de las interrupciones.
Los elementos de mayor prioridad van primero.

Directivas de control de la memoria del microcontrolador:

Para gestionar y reservar el uso de determinadas zonas de memoria para variables:

#ASM Y #ENDASM

Las líneas entre *#ASM* y *#ENDASM* se tratan como código ensamblador. La variable predefinida _RETURN_ puede utilizarse para asignar un valor de retorno a la función desde el código en ensamblador.

#BIT

Esta directiva creará un identificador "id" que puede utilizarse como cualquier SHORT INT (entero corto; un bit). El identificador referenciará un objeto en la posición de memoria x más el bit de desplazamiento y.

#BYTE

Esta directiva creará un identificador "id" que puede utilizarse como cualquier NT (un byte). El identificador referenciará un objeto en la posición de memoria x, donde x puede ser una constante u otro identificador. Si x es otro identificador, entonces éste estará localizado en la misma dirección que el identificador "id".

#RESERVE

Permite reservar posiciones de la RAM para uso del compilador.

#RESERVE debe aparecer después de la directiva #DEVICE, de lo contrario no tendrá efecto.

#ROM

Esta directiva permite insertar datos en el archivo .HEX. En particular, se puede usar para programar la EEPROM de datos de la serie 84 de PIC.

#ZERO_RAM

Directiva que pone a cero todos los registros internos que pueden usarse para mantener variables, antes de que comience la ejecución del programa.

Identificadores predefinidos:

Dentro del término genérico de directiva se incluyen, además de estos comandos, unas variables que contienen información sobre el proceso de compilación. Estas variables son lo que se denominan identificadores predefinidos del compilador:

__DATE__

Este identificador del pre-procesador contiene la fecha actual (en tiempo de compilación) en el formato siguiente: "dd-mm-aa"

__DEVICE__

Este identificador del pre-procesador es definido por el compilador con el número base del dispositivo actual. El número base normalmente es el número que sigue a la/s letra/s en el número de componente o referencia de un dispositivo. Por ejemplo los PIC16C84 tienen el número base 84

__**PCB**__

Se utiliza para determinar si es el compilador PCB el que está haciendo la compilación.

__**PCH**__

Se utiliza para determinar si es el compilador PCH el que está haciendo la compilación.

__**PCM**__

Se utiliza para determinar si es el compilador PCM el que está haciendo la compilación.

En un programa, las directivas se reconocen fácilmente porque comienzan por el símbolo #, mientras que los identificadores empiezan y acaban por doble subrayado (__).

1.8.1.2. Funciones precompiladas.

Se puede facilitar considerablemente la tarea de programación si no es necesario construir por nosotros mismos aquellas funciones que son de utilización más frecuente, como leer la entrada de un teclado o imprimir un determinado mensaje en una pantalla LCD conectada como salida.

Existen funciones en C incluidas en el compilador PCW para manejar los diferentes recursos del microcontrolador, desde el bus I2C hasta el conversor A/D.

1.8.1.3. Consideraciones a tener en cuenta cuando vayamos a programar

Inserción de código Ensamblador:

Mediante las directivas #ASM y #ENDASM es posible insertar código Ensamblador directamente en el código C, con lo que determinados procedimientos se pueden implementar directamente en Ensamblador, con el ahorro de código y tiempo de ejecución que ello implica.

Gestión automática de páginas de código:

Los microcontroladores PIC disponen de varias páginas de memoria de programa, lo que en la programación manual supone tener en cuenta si los saltos se producen a otra página de código diferente de la activa, y modificar las páginas de código según corresponda, (y si no que se lo digan a los que programan en Ensamblador). Este aspecto es gestionado de manera automática por el compilador, el cual, de manera transparente al usuario, insertará las instrucciones necesarias para el cambio de página de código. Además, durante el proceso de enlazado se analiza el código objeto, de manera que aquellas funciones que son llamadas frecuentemente son situadas en la misma página de código que la función que las llama, disminuyendo de este modo el tamaño del código y el retardo producido por la llamada a la función.

Gestión automática de bancos de memoria:

El compilador también gestiona de manera automática el cambio entre bancos de memoria RAM, y trata de minimizar el cambio entre bancos, intentando agrupar las variables locales utilizadas en cada función dentro de un mismo banco.

Mapeo de la memoria del microcontrolador desde el código C:

Las directivas #BIT y #BYTE permiten asignar variables del código C a determinadas direcciones de la memoria del microcontrolador, evitando de este modo que el compilador asigne automáticamente variables a zonas de memoria que interesa mantener libres. Esta característica también se puede realizar con otros tipos de datos, como las estructuras.

Mapeo de tipos de datos de tamaño bit:

El compilador permite utilizar datos de tamaño 1, 8 y 16 bits, y 32 bits en notación de coma flotante. De estos datos, especial interés tienen los datos de tamaño 1 bit (tipo Short Int), que permite utilizar de manera muy eficiente la memoria del microcontrolador. Por ejemplo, en lugar de utilizar una variable de tipo #BYTE para implementar un indicador o flag, es posible utilizar una variable de tipo #BIT, con lo que la memoria ocupada es menor y el código relacionado con esta variable se

optimiza. Además, las estructuras y uniones también permiten definir campos de bits, optimizando el almacenamiento de información de estos tipos de datos.

Almacenamiento de constantes en memoria de programa:

Los tipos de datos constantes, las cadenas de texto y las matrices o arrays son almacenados en la memoria de programa del microcontrolador, en lugar de en la memoria de datos. Como consecuencia, y debido a la arquitectura Harvard de los PIC, en la que los segmentos de memoria de programa y datos son independientes, no es posible tratar las zonas de memoria de programa como datos, por lo que no se permite el empleo de punteros a arrays de constantes, ni tampoco a funciones, una nota a tener en cuenta para los que saben programar en C, pero desconocen las características de este compilador.

Soporte de punteros y referencias:

El compilador PCW permite el uso de punteros de 8 y 16 bits. Además, también permite el paso de parámetros por referencia de variables, lo que proporciona la misma potencia que los punteros, mejorando al mismo tiempo la legibilidad del código (esto lo veremos con ejemplos en el curso de C).

Eficiente implementación de variables y funciones:

La eficiencia de los algoritmos de optimización empleados en este compilador permite que, en el caso de las variables, éstas sean asignadas a zonas de memoria que se emplee la menor memoria posible de forma global, reutilizando para ello posiciones de memoria. Por ejemplo, el compilador asigna de manera automática a una misma dirección de memoria varias variables definidas como locales en funciones diferentes a main, ya que en ningún momento podrá existir más de una de estas variables de manera simultánea, pues su valor dejará de existir al salir de la función donde se encuentran definidas. En cuanto a las funciones, es posible realizar anidaciones de funciones en niveles más profundos que el permitido por la pila hardware, debido a una implementación eficiente de las llamadas a estas funciones.

Generación del código estrictamente necesario:

En la compilación del código objeto final sólo se incluyen aquellas funciones de las bibliotecas pre compiladas que son utilizadas en el programa. De la misma forma, las funciones de usuario que no son llamadas nunca y las sentencias que no se pueden utilizar debido a la lógica del programa no son incluidas en el código pre compilado.

CAPITULO II

DISEÑO DEL HARDWARE

Introducción

Este capítulo trata el tema correspondiente al diseño y construcción del hardware necesario para la realización del siguiente proyecto, incluyendo diagramas esquemáticos de cada parte del diseño.

Hay que señalar que el módulo está constituido por tarjetas tales como la Master de Comunicaciones, Módulos de entrada y salidas, tanto análogas como digitales, la fuente de alimentación, el módulo Master de la parte de la mini planta de procesos y la tarjeta donde se conectarán los sensores (RTD, Termocupla y Celda de carga).

Se han adjuntado las imágenes de cada una de las tarjetas que han resultado del diseño y construcción descritos en éste capítulo, estas imágenes se encuentran adjuntas en el Anexo III de este documento.

2.1. Diseño de la Fuente de Alimentación.

En electrónica, una fuente de alimentación es un dispositivo que convierte la tensión alterna de la red comercial, en una o varias tensiones, prácticamente continuas, que alimentan los distintos circuitos del aparato electrónico al que se conecta.

Las fuentes de alimentación, para dispositivos electrónicos, pueden clasificarse básicamente como fuentes de alimentación; lineales y conmutadas. Las lineales tienen un diseño relativamente simple, que puede llegar a ser más complejo cuanto mayor es la corriente que deben suministrar, sin embargo su regulación de tensión es poco eficiente. Una fuente conmutada, de la misma potencia que una lineal, será más pequeña y normalmente más eficiente pero será más complejo y por tanto más susceptible a averías.

2.1.1. Fuentes de alimentación lineales

Las fuentes lineales siguen el esquema: transformador, rectificador, filtro, regulación y salida.

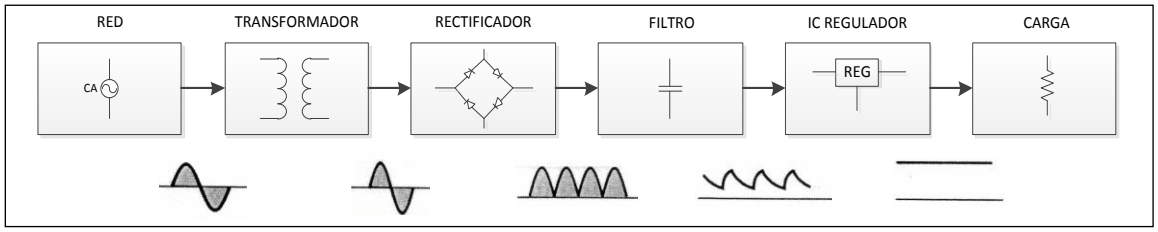


Figura 2.1. Esquema de una fuente de alimentación

En primer lugar el transformador adapta los niveles de tensión y proporciona aislamiento galvánico. El circuito que convierte la corriente alterna en continua se llama rectificador, después suelen llevar un circuito que disminuye el rizado como un filtro de condensador. La regulación, o estabilización de la tensión a un valor establecido, se consigue con un componente denominado regulador de tensión. La salida puede ser simplemente un condensador. Esta corriente abarca toda la energía del circuito.

2.1.1.1. Circuito rectificador de media onda

Debido a que un diodo pude mantener el flujo de corriente en una sola dirección, se puede utilizar para cambiar una señal de AC a una de dc. En la figura se muestra un circuito rectificador de media onda. Cuando la tensión de entrada es positiva, el diodo se polariza en directo y se puede sustituir por un corto circuito. Si la tensión de entrada es negativa el diodo se polariza en inverso y se puede remplazar por un circuito abierto. Por tanto cuando el diodo se polariza en directo, la tensión de salida a través del resistor se puede hallar por medio de la relación de un divisor de tensión sabemos además que el diodo requiere 0.7 voltios para polarizarse así que la tensión de salida esta reducida en esta cantidad (este voltaje depende del material de la juntura del diodo). Cuando la polarización es inversa, la corriente es cero, de manera que la tensión de salida también es cero. Este rectificador no es muy eficiente debido a que durante la mitad de cada ciclo la entrada se bloquea completamente desde la salida, perdiendo así la mitad de la tensión de alimentación.

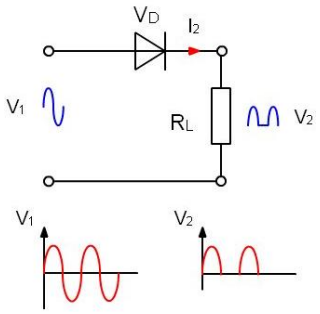


Figura 2.2. Circuito rectificador de media onda

El valor medio de la tensión de salida viene dado por:

$$V_2 = \frac{1}{2\pi} \int_0^{\pi} V_1 sen(\omega t) d(\omega t) = \frac{V_1}{\pi}$$

2.1.1.2. Circuito rectificador de onda completa en puente.

El circuito conocido como rectificador en puente por la similitud de su configuración con la del puente de Wheatstone, no requiere de transformador con derivación central.

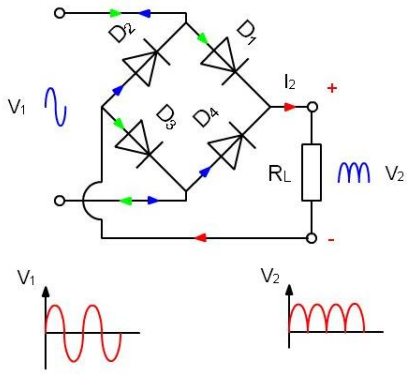


Figura 2.3. Circuito rectificador de onda completa

Este circuito rectifica el semiciclo negativo de tensión y lo convierte en positivo, para conseguirlo uno de los métodos es utilizar un puente de diodos. La eficiencia de éste montaje es muy alta por lo que es muy utilizado.

Se trata de un montaje con cuatro diodos, en el semiciclo positivo los diodos D1 y D3 permiten el paso de la corriente hasta la carga, con la polaridad indicada. En el semiciclo negativo son D2 y D4 los que permiten el paso de la corriente y la entregan a la carga con la misma polaridad que en el caso anterior.

2.1.1.3. Reguladores lineales de tensión

Los reguladores lineales de tensión, también llamados reguladores de voltaje, son circuitos integrados diseñados para entregar una tensión constante y estable.

Estos dispositivos están presentes en la gran mayoría de fuentes de alimentación, pues proporcionan una estabilidad y protección sin apenas necesidad de componentes externos haciendo que sean muy económicos.

La tensión y corriente que proporcionan es fija según el modelo y va desde 3.3v hasta 24v con un corriente de 0.1A a 3A.

Figura 2.4. Encapsulado TO220 del regulador

La identificación del modelo es muy sencilla. Las dos primeras cifras corresponden a la familia:

- 78xx para reguladores de tensión positiva
- 79xx para reguladores de tensión negativa

Las dos cifras siguientes corresponden al voltaje de salida:

- xx05 para tensión de 5v
- xx12 para 12v
- xx24 para 24v

¿Cómo funciona?

Una visión simplificada, para entender su funcionamiento, sería verlos como un divisor de tensión que se reajusta constantemente para que la tensión entregada sea siempre la misma. Evidentemente no es tan simple como una par de resistencias ajustables. En el interior de un regulador lineal de tensión pueden encontrarse componentes activos, como transistores trabajando en su zona lineal, y/o pasivos, como diodos zener, en su zona de ruptura.

Los tres terminales corresponden a la Tensión de entrada (Vin), Tierra (ground) y Tensión de salida (Vout). Según el encapsulado, TO92, TO220 o TO3, la asignación de los pinouts puede variar.

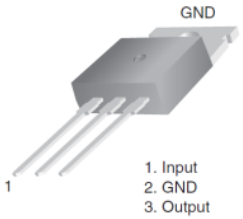


Figura 2.5. Pines de conexión del regulador.

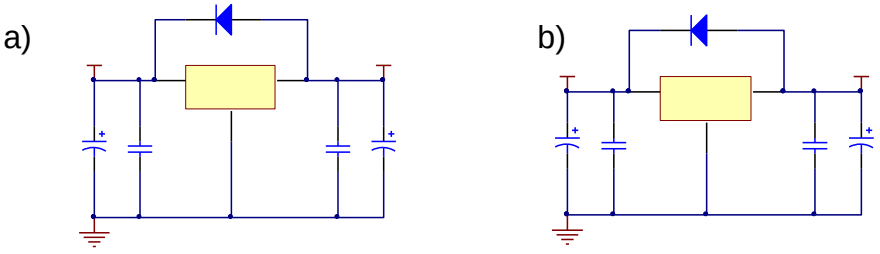


Figura 2.6. A) Fuente regulada de 5 VDC. B) Fuente regulada de 12 VDC

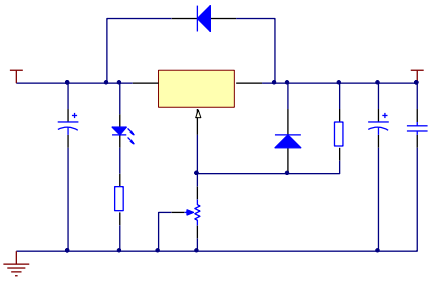


Figura 2.7. Fuente regulada de 24 VDC.

2.2. Tarjeta Base “Back Panel”

La tarjeta base o Back Panel [1] es la que se encarga de la de la comunicación de cada una de las tarjetas que serán colocadas en el módulo y de la distribución de los diferentes niveles de energía a todo el módulo. Por esta tarjeta base o Back Panel se realizara comunicaciones como SPI, I^2C, Modbus y se distribuyen niveles de energía como 0 (GND), 5, 12 y 24 VDC.

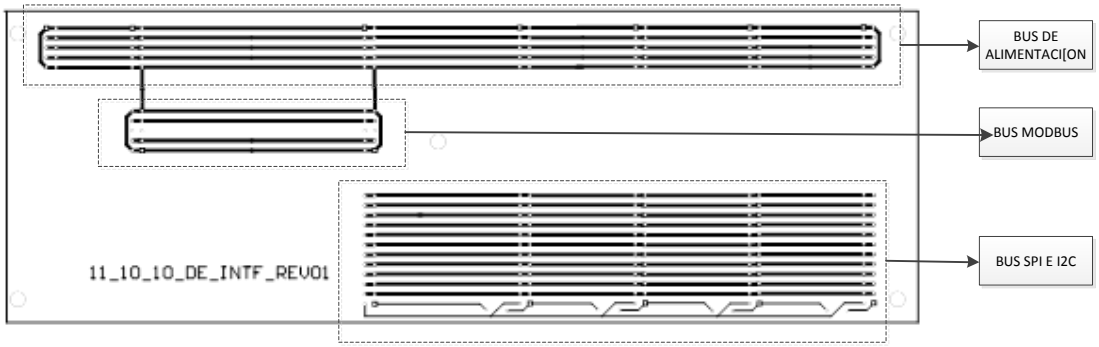


Figura 2.8. Ruteado de la placa base, Back Panel

Se puede observar que existen varios slots los cuales se encuentran interconectados entre si, esto se debe a que estas rutas son las que generarán las vías de comunicación de los buses I^2C, SPI y Modbus, las otras de las líneas interconectadas corresponden a la distribución de energía para cada una de las tarjetas, a continuación se detalla a que corresponde cada uno de los puntos de los slots de la placa base.

[1] *Ver en anexo I el diagrama esquemático y PCB para esta tarjeta (Back Panel)*

2.2.1. Bus de alimentación.

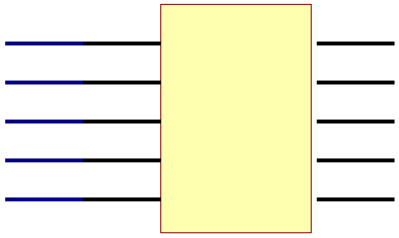


Figura 2.9. Slot de alimentación

El bus de alimentación[1] como se puede observar en al figura entregara voltajes de 0, 5, 12 y 24 VDC hacia las tarjetas de del módulo, este bus es el mas grande que tenemos en la placa ya que este debe alimentar a todos los elementos del módulo con sus diferentes niveles de tensión. Estas tensiones vienen de la placa de alimentación en a que se encuentran los circuitos de regulación para cada uno de los niveles indicados. A este bus lo reconoceremos debido a que hay 7 slots para cumplir con la distribución de energía hacia las diferentes tarjetas que conectemos. Este recibe la energía directamente de la placa que contiene la fuente de alimentación, se conecta a través de conectores de tipo ICD y PCD.

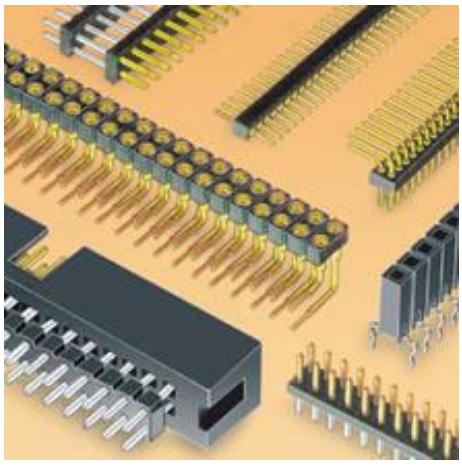


Figura 2.10. Diferentes modelos de conectores ICD y PCB

[1] *Ver en anexo I el diagrama esquemático y PCB de la fuente de alimentación y de la tarjeta Back Panel,*

2.2.2. Bus Modbus.

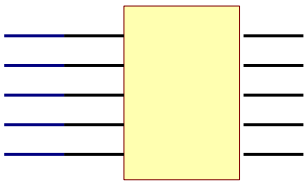


Figura 2.11. Slot de comunicación Modbus

El bus Modbus[1] es similar al bus citado anteriormente debido a que tienen el mismo número de pines de conexión hacia las tarjetas mas este no lleva niveles de como 5, 12 o 24 VDC ya que tiene una conexión directa hacia el común del módulo (GND). Este bus se encargara de la comunicación entre las tarjetas master de comunicaciones en su funcionamiento como esclava en la comunicación Modbus y la tarjeta master de procesos en su funcionamiento como esclava en la comunicación Modbus, con los dispositivos Master.

Será complicado confundirse entre un slot del bus de alimentación y un slot del bus de comunicación Modbus ya que el primer bus consta de 7 slots para las alimentación de las tarjetas en línea recta y 2 slots para la comunicación de las dos tarjetas antes mencionadas igualmente colocados en línea recta.

2.2.3. Bus SPI e I2C.

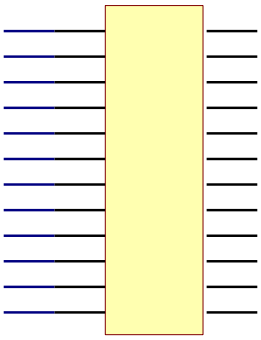


Figura 2.12. Slot de comunicación I2C y SPI

[1] *Ver en anexo I el diagrama esquemático y PCB de las tarjeta CPU-MCM, MPM y Back Panel.*

Estos slots serán los que se encargue de la comunicación SPI e I^2C[1] a través de los puntos que se indican en el conector, siendo los puntos SDA y SCL los encargados de la comunicación I^2C y los puntos SDO, SDI y SCK los encargados de la comunicación SPI. Este bus también se distribuirá en todas las tarjetas a excepción de la tarjeta master de procesos. Se nota también que esta consta de puntos como SS1,….SS7, estos son los encargados de generar el "Set Select" para realizar la comunicación SPI entre la tarjeta master y los diferentes esclavos del módulo, específicamente las tarjetas de entradas y salidas análogas.

2.3. Diseño de la Tarjeta Master de Adquisición de Datos (CPU-MCM)

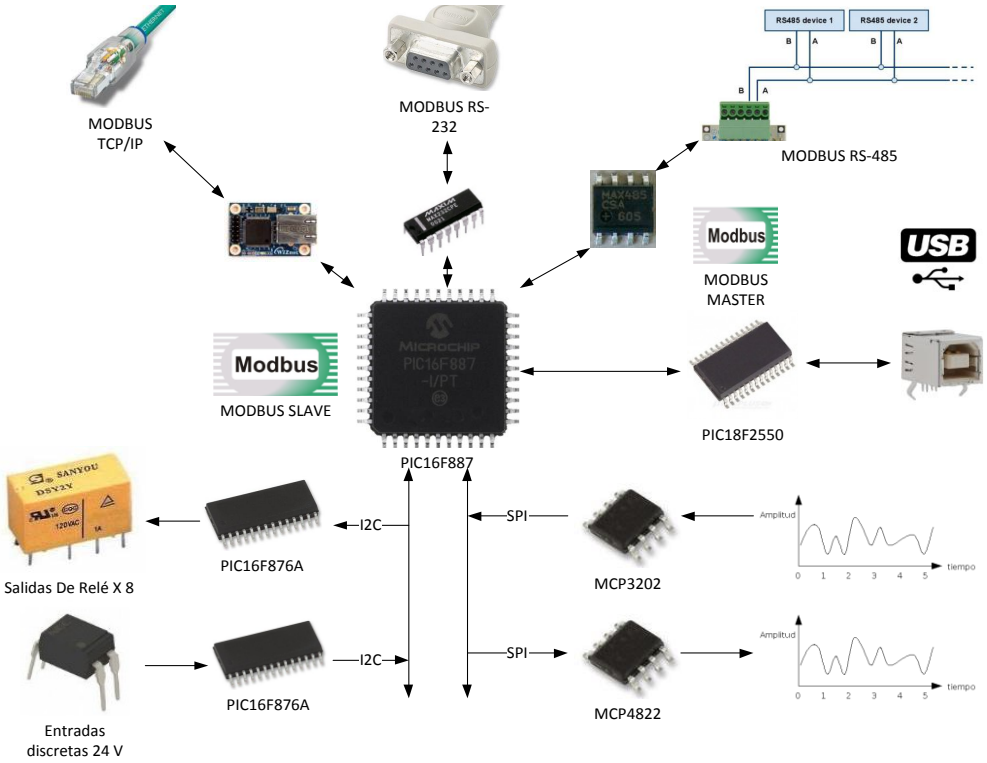


Figura 2.13. Esquema general de la tarjeta master (CPU-MCM)

[1] *Ver en anexo I el diagrama esquemático y PCB para las tarjetas CPU-MCM, MAE, MAS, MDE, MDS y Back Panel.*

La tarjeta master de adquisición de datos se la va a denominar (CPU-MCM) que nos indica que es la Unidad Central de Proceso y será identificado como Módulo de Comunicaciones Master la cual se encargara del control y adquisición de señales de tipos digitales y análogas del módulo en su totalidad. Este consta de comunicaciones como USB[1], Modbus sobre RS – 232, Modbus sobre RS – 485 y Modbus sobre TCP/IP[2], a continuación se detalla el diseño del hardware de la tarjeta master. Ésta tarjeta esta basada en un micro controlador PIC16F887 de la industria Microchip, el cual actúa como un elemento principal de la tarjeta master ya que este se encarga de interactuar con todos los elementos existentes en el módulo. La tarjeta se complementa con un PIC18F2550 para la comunicación USB el cual a su vez realiza las funciones de comunicación Modbus pero como microprocesador *master* a la vez que el PIC16F887 realiza las funciones de microprocesador esclavo.

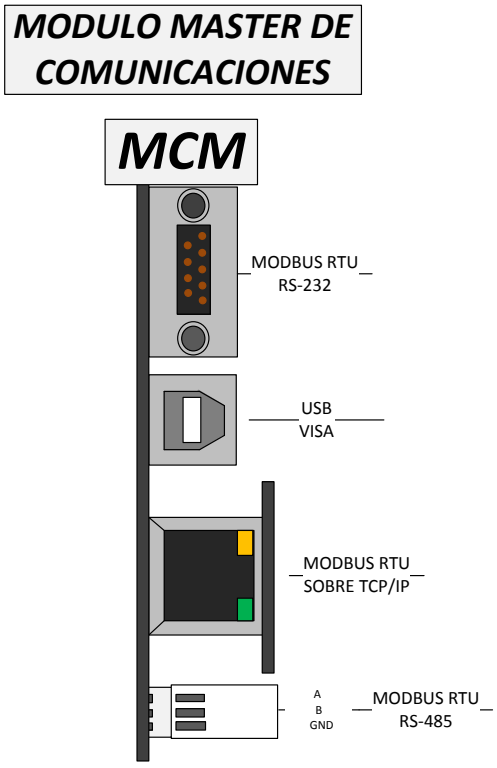


Figura 2.14. Diagrama frontal tarjeta Master de comunicaciones.

[1] *Ver en el capítulo 1 la sección Comunicación USB y en capítulo 3 la sección Configuración NI-VISA*
[2] *Ver en el capítulo 1 la sección Protocolo MODBUS y en capítulo 3 la sección Configuración MODBUS.*

2.3.1. Comunicación RS – 232.

El microcontrolador cuenta con un puerto USART el cual usa 2 pines uno para transmisión y otro para recepción. USART[1] es el acrónimo de *Universal Synchronous/Asynchronous Receiver Transmitter*, que traducido al español viene a ser algo parecido a *Transmisor y Receptor Sincrónico/Asincrónico Universal*.

Se trata de un periférico para la transmisión de datos en formato serie, utilizando técnicas de transmisión sincrónica o asincrónica, según se configure el periférico.

Para el propósito de diseño del hardware se utilizara el circuito de aplicación más común del CI MAX232 y eficaz, el cual se encargara de convertir lo niveles lógicos de tipo CMOS a niveles lógicos TTL y viceversa.

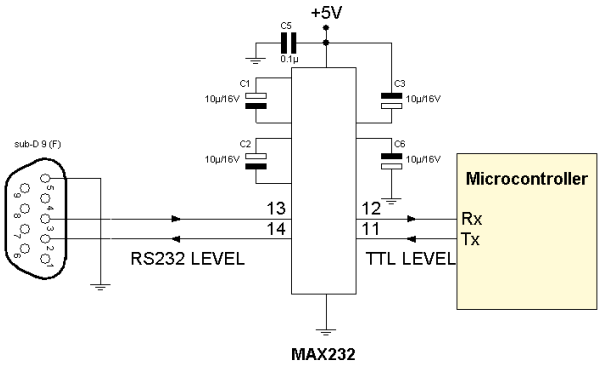


Figura 2.15. Circuito básico de conexión de un CI MAX232 a un microcontrolador

En el caso de los microcontroladores PIC se tienen pines asignados para realizar esa función, el microcontrolador al que se le ha asignado la función de comunicación RS-232[2] es el PIC16F887 el cual tiene como pines de comunicación serial (USART) los pines RC6 para la transmisión y RC7 para la recepción serial. Estos pines pueden ser configurados mediante software para ser remplazados por otros pines del mismo controlador y cumplirán con la misma función transmisión y recepción serial lo cual no se realizó para nuestra aplicación. A continuación se presenta el diagrama de conexiones realizado para la comunicación uC – PC.

[1] *Revisar para más información http://www.ucontrol.com.ar*
[2] *Ver en anexo II, las hojas de datos del CI. MAX232*

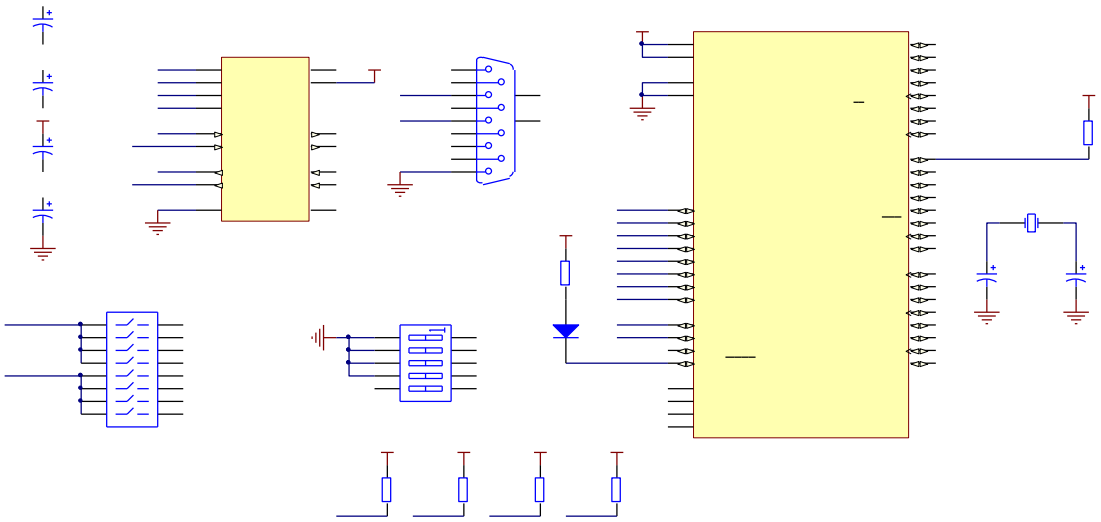


Figura 2.16. Diagrama de conexiones RS-232, PIC16F887 y el circuito integrado MAX232.

Se puede observar que existe un bloque dip switch el cual se encarga de la conexión de los periféricos externos de comunicación a los pines ya que la comunicación RS – 232 no es la única en la tarjeta master. Los puntos en el dip switch correspondientes a la comunicación RS – 232 son el 4 para la transmisión desde el microcontrolador a la PC y el punto 8 para la recepción de datos hacia el microcontrolador. Cabe recalcar que en este tipo de comunicación únicamente podremos comunicarnos con un solo dispositivo desde el computador.

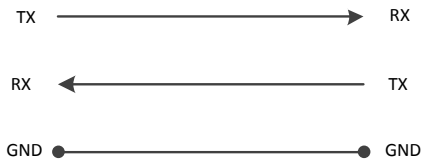


Figura 2.17. Cableado típico para el estándar RS – 232.

2.3.2. Comunicación RS-485.

Este tipo de comunicación al igual que el estándar RS – 232 se realiza de manera serial hacia la USART del microcontrolador y se utilizarán los mismos pines. La diferencia entre el estándar RS – 232 y RSS – 485 es la distancia de comunicación que nos ofrecen y la capacidad de interactuar con otros dispositivos colocados en una red lo cual en una comunicación RS – 232 no se puede realizar.

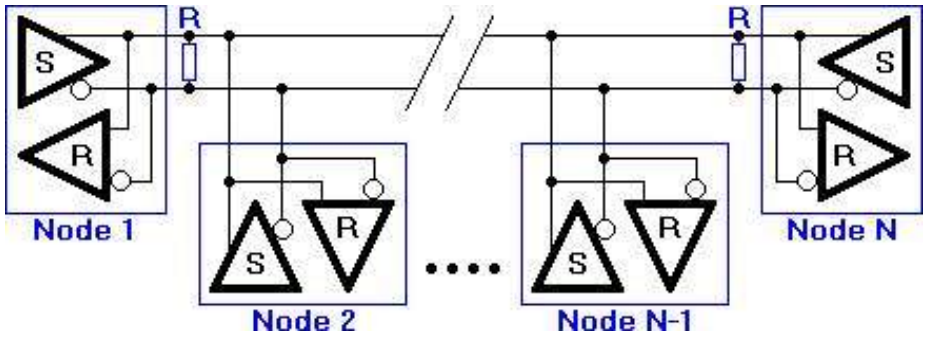


Figura 2.18. Estructura de la interconexión de dispositivos utilizando la configuración de dos hilos.

A continuación se presenta el circuito utilizado para la comunicación RS – 485[1] del dispositivo, se puede observar que consta del mismo dip switch para direccionar los pines RX y TX del microcontrolador hacia los puntos de transmisión y recepción del CI SN75176[2] (CI equivalente a un MAX485), los puntos correspondientes la comunicación RS – 485 son los puntos 1 para la transmisión desde el microcontrolador hacia la red y 5 para la recepción de datos desde al red hacia el microcontrolador.

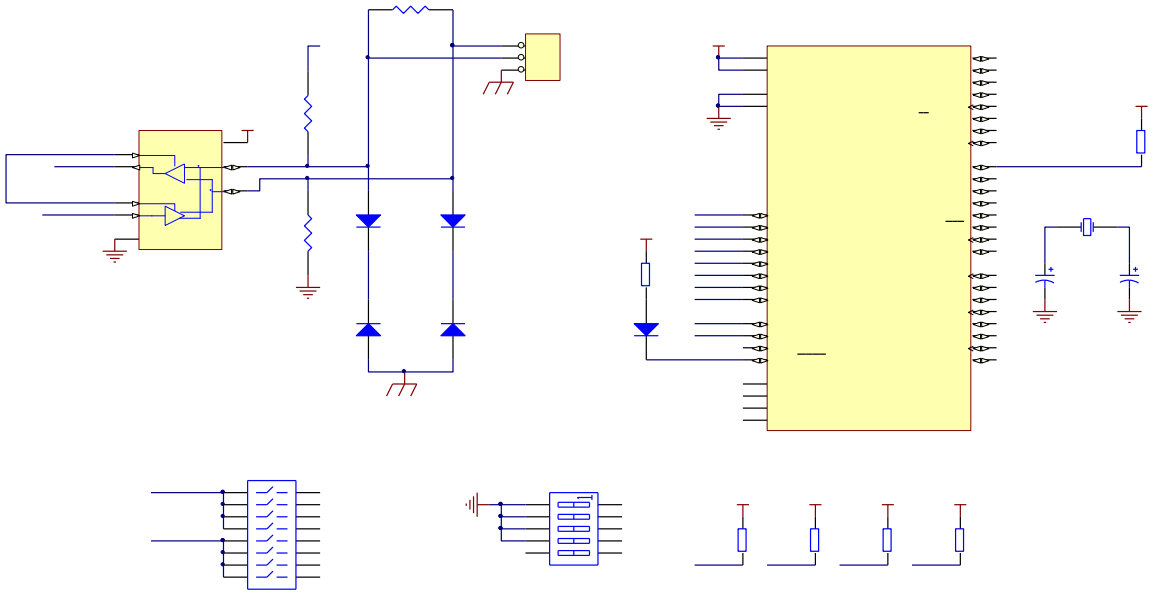


Figura 2.19. Diagrama de conexiones de la comunicación RS-485 con el PIC16F887

[1] *Revisar para mas información http://www.i-micro.com*
[2] *Ver en anexos II, las hojas de datos del CI. SN75176*

2.3.3. USB.

La comunicación USB se dará a través de un microcontrolador PIC18F2550[1] utilizando su periférico USB.

Este microcontrolador a su vez hace las veces de Master Modbus, tendrá una comunicación directa con el microcontrolador PIC16F887 que se encuentra en la tarjeta master de comunicaciones MCM y con el PIC16F887 que se encuentra montado sobre la tarjeta master de procesos, esta comunicación se llevara a cabo tomando en cuenta cada una de las direcciones de los microcontroladores.

La tarjeta Master de Procesos MPM utilizará la dirección Modbus número 20, la dirección del microcontrolador esclavo de la tarjeta Master de Comunicaciones puede ser seleccionable en forma binaria con la ayuda de un Dip Switch de 4 puntos de selección lo cual nos permite seleccionar valores entre 0 y 15 en decimal, tomando en cuenta que las direcciones 0 y 1 están reservadas para propósitos de configuración interna del dispositivo esclavo mediante mensajes distribuidos.

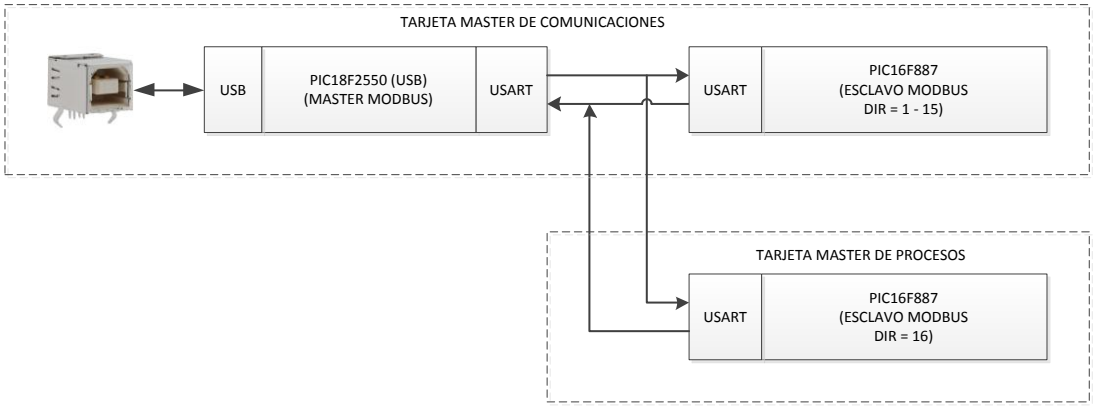


Figura 2.20. Diagrama de bloques comunicación entre Microcontroladores

El microcontrolador de PIC18F2550 se encargara única y exclusivamente de la comunicación USB para así pasar la información vía Modbus a los microcontroladores esclavos antes mencionados.

[1] *Ver en anexo II, las hojas de datos del Microcontrolador PIC18F2550*

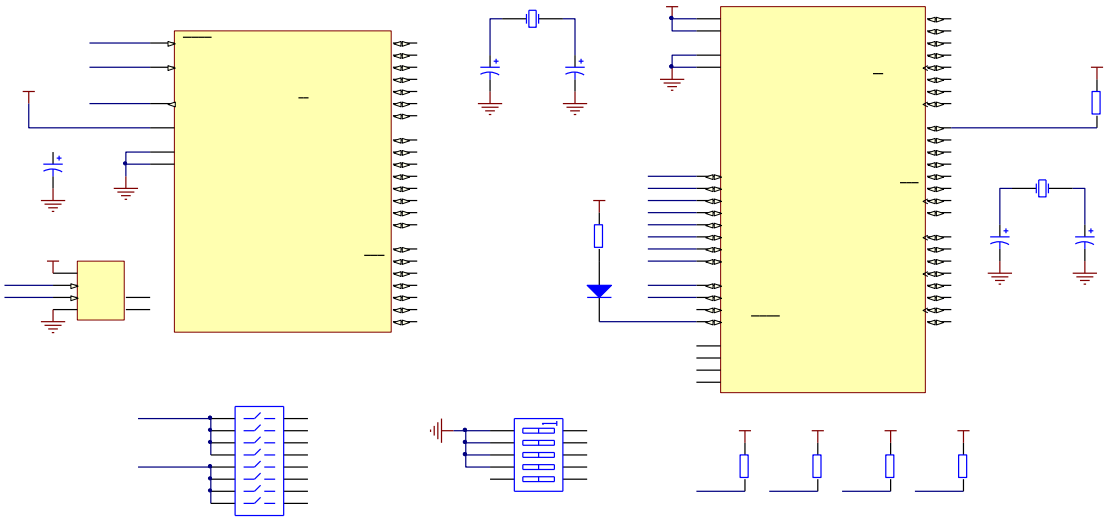


Figura 2.21. Diagrama de conexiones de la comunicación RS-2485 con el PIC16F887

2.3.4. ETHERNET.

Para poder integrar el módulo de adquisición de datos a una red Ethernet, hemos utilizado el Módulo WIZ107SR[1], el cual se describe a continuación.

2.3.4.1. El WIZ107SR

Figura 2.22. Módulo WIZ107SR

Módulo WIZ107SR, es un módulo Gateway entre dispositivos seriales y Ethernet, es decir, puede transmitir datos seriales hacia una red Ethernet y viceversa de

[1] *Ver en anexo II, las hojas de datos del Módulo WIZ107RSR.*

esta manera se podrá generar conexiones seriales – Ethernet de una manera fácil y rápida. El módulo puede ser configurado utilizando comandos por el puerto serial o a través de la red Ethernet utilizando el software de configuración[1].

Características principales

- Tamaño compacto RS-232 a Ethernet.
- Alimentación de 3.3 VDC
- Conexión a Internet rápida y sencilla a los Dispositivos Serie.
- Garantizar la comunicación de datos estable y fiable mediante el uso de un chip W7100.
- Proporciona una herramienta de programación fácil y amigable para el usuario.
- Soporte de protocolo PPPoE y configuración de autenticación para los usuarios de ADSL.
- Soporte de contraseña para el usuario a través de software.
- Velocidades de comunicación: 10/100 Mbps para la comunicación Ethernet y 230 Kbp máximo para la comunicación serial.
- Soporte de configuración de IP - estática DHCP y PPPoE.
- Soporta DNS.
- Diseño compacto (48mm X 30mm X 18mm).

Este módulo irá montado sobre la tarjeta master de comunicaciones, se comunicara vía RS – 232 con el microcontrolador por lo que será necesario ayudarnos del circuito básico de comunicaciones RS – 232 a través de un CI MAX232 ya que la comunicación serial enviada por el módulo WIZ107SR es de niveles lógicos CMOS siendo necesario el uso de este circuito adicional.

[1] *Revisar para más* información acerca del Sofware de configuración http://www.olimex.cl

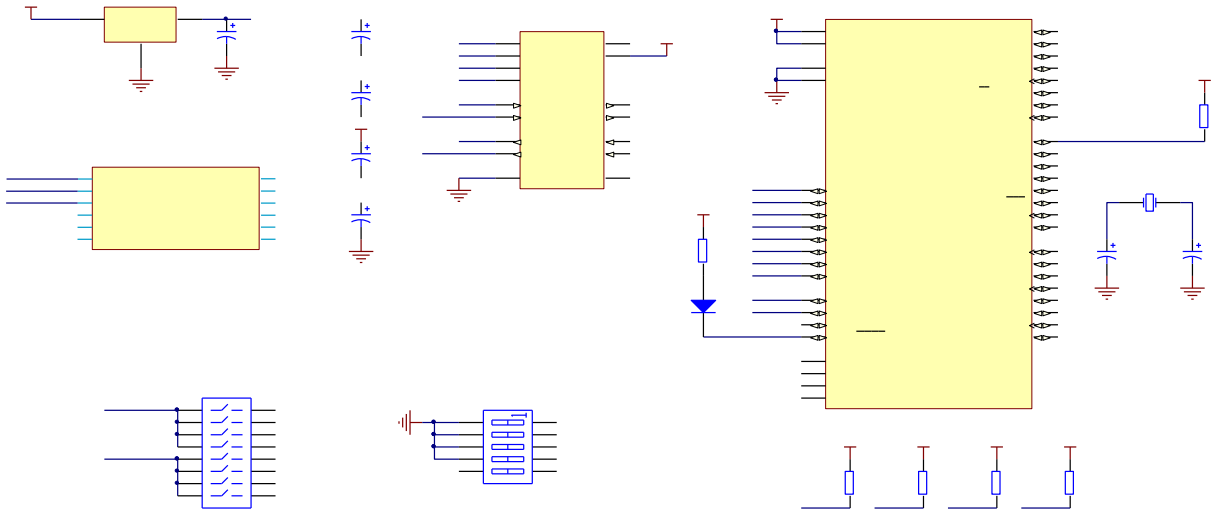


Figura 2.23. Diagrama de conexiones de la comunicación Ethernet con el PIC16F887

Como se puede notar en al figura anterior la comunicación se módulo será habilitada a través de los dip switch siendo los puntos que corresponden a la comunicación los puntos 2 para la transmisión serial del microcontrolador y el 6 para la recepción de microcontrolador.

2.3.5. Funcionamiento de los dip switch de Selección de vía de comunicación y direccionamiento.

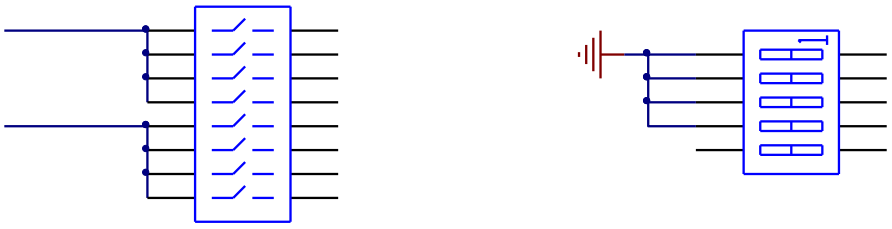


Figura 2.24. Dip Swicth utilizados para selección de comunicación y direccionamiento

En la tarjeta master de comunicaciones MCM[1] existen dos dip switch que se encargan de la selección de la vía de comunicación y de setear la dirección del esclavo Modbus, esto se lo realizará simplemente activando y desactivando cada uno de los puntos de los dip switch.

2.3.5.1. Dip switch de vía de comunicación

Para habilitar las diferentes conexiones de comunicación al módulo se deberán activar y desactivar los puntos de los dip switch de la siguiente manera.

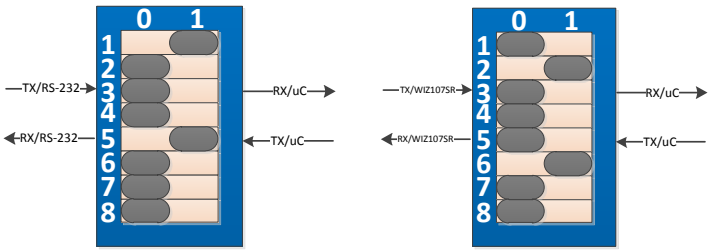


Figura 2.25. Vía RS – 232 Vía Ethernet (WIZ107SR)

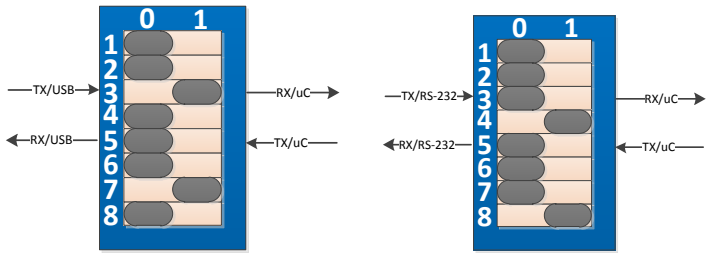


Figura 2.26. Vía USB Vía RS – 485

2.3.5.2. Dip switch de direccionamiento

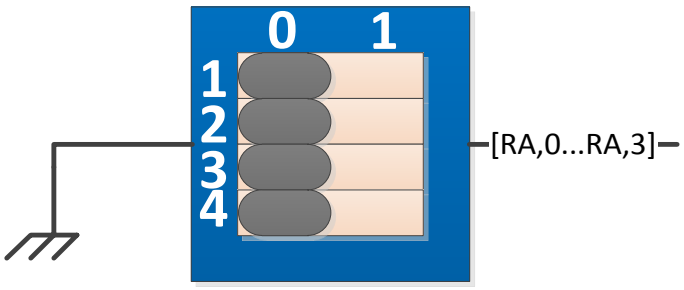


Figura 2.27. Dip Switch para asignar dirección Modbus

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta CPU-MCM.*

El funcionamiento de este dip switch consiste en enviar 1's y 0's hacia el Puerto A del microcontrolador PIC16F887 de la tarjeta master de comunicaciones, específicamente hacia los bit's 0, 1, 2 y 3 del Puerto A.

Al utilizar únicamente 4 bit's del puerto podremos generar 16 combinaciones que irán desde el numero 0 al 15 las cuales será utilizadas como dirección Modbus[1] del esclavo.

A continuación se presentan las posibles combinaciones en el dip switch.

RA3	RA2	RA1	RA0	#
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Tabla 2.1. Tabla de direcciones Modbus

Se deberá tener en cuenta que las 16 combinaciones son válidas mas la dirección 0 se encuentra reservada para caso especial del protocolo Modbus ya que esta se utilizara para enviar un dato específico a todos los esclavos Modbus.

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta CPU-MCM.*

2.4. Diseño de la Tarjeta Master de Procesos.

La tarjeta Master de procesos[1] se encarga realizar el control sobre los elementos actuadores como motores y monitorea señales de entradas discretas y análogas, además actúa como Master I2C[2] y SPI[3] para establecer comunicación con los integrados MCP3202, MCP4822, MCP3551 y MCP3421.

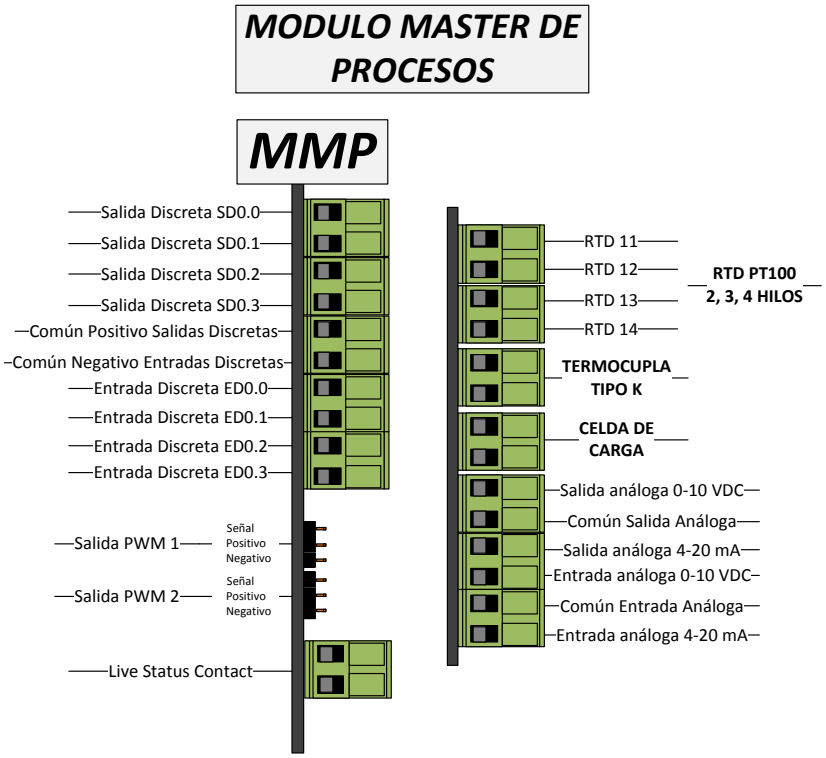


Figura 2.28. Vista frontal del Módulo Master de Proceso (MPM)

Este módulo tiene como elemento primario un microcontrolador PIC16F887 el cual actúa como un dispositivo Modbus Esclavo de dirección 16, puede establecer comunicación utilizando los mismos puertos físicos del módulo Master de comunicaciones, es decir puede comunicarse vía RS232, RS485, Ethernet y USB, ya el puerto USART de este microcontrolador está conectado al bus MODBUS perteneciente al Back Panel.

---

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta MPM.*
[2] *Ver en el capítulo 1 la sección Protocolo SPI y en capítulo 3 la sección Configuración SPI.*
[3] *Ver en el capítulo 1 la sección Protocolo I2C y en capítulo 3 la sección Configuración I2C.*

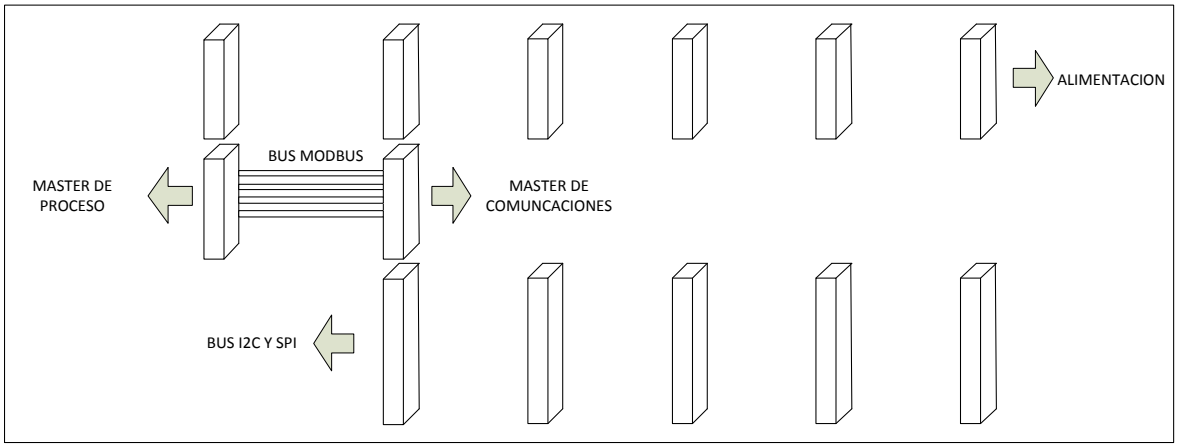


Figura 2.29. Back panel, indicando los conectores de Modbus para el módulo Master de procesos

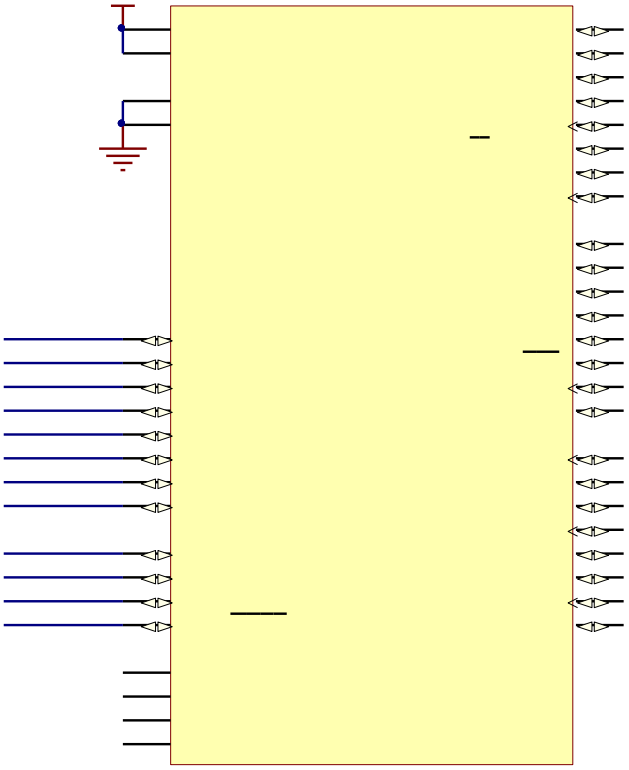


Figura 2.30. Distribución de pines del microcontrolador PIC16F887 para el módulo master de procesos

El módulo Master de Procesos va a ejercer control y monitoreo sobre los elementos que se detallan a continuación.

ELEMENTO	CONTROL	TIPO DE CONTROL	MONITOREO	TIPO DE MONITOREO

MOTOR DC	SI	Inversión de giro Control de velocidad proporcional	SI	Contador para conectar encoder (Timer 1)
MOTOR DE PASOS	SI	Control de giro paso a paso	SI	Contador para determinar número de pasos (Timer 0)
SERVOMOTOR	SI	Control de Posición (PIC16F819)	SI	Verificar la posición de acuerdo al valor en el módulo CCP
DICROICO	SI	Control de Posición (PWM)	SI	Verificar la iluminación de acuerdo al valor en el módulo CCP
RTD PT100	---	---	SI	Adquisición de datos por medio de CI MCP3553[1]
Termocupla	---	---	SI	Adquisición de datos por medio de CI MCP3421[2]
Celda de carga	---	---	SI	Adquisición de datos por medio de CI MCP3553
Entradas discretas	---	---	SI	Conectadas al puerto A RA0, RA1, RA2 y RA3
Entradas análogas	---	---	SI	Conectadas a ambos canales del CI MCP3202[3]
Salidas discretas	SI	Conectadas al puerto D RD0, RD1, RD2 y RD3	---	---
Salidas análogas	SI	Conectadas a ambos canales del CI MCP4822[4]	---	---

Tabla 2.2. Elementos de control de la tarjeta master de procesos

[1] *Ver en anexo II, las hojas de datos del CI MCP3553.*
[2] *Ver en anexo II, las hojas de datos del CI MCP3421.*
[3] *Ver en anexo II, las hojas de datos del CI MCP3202.*
[4] *Ver en anexo II, las hojas de datos del CI MCP4822.*

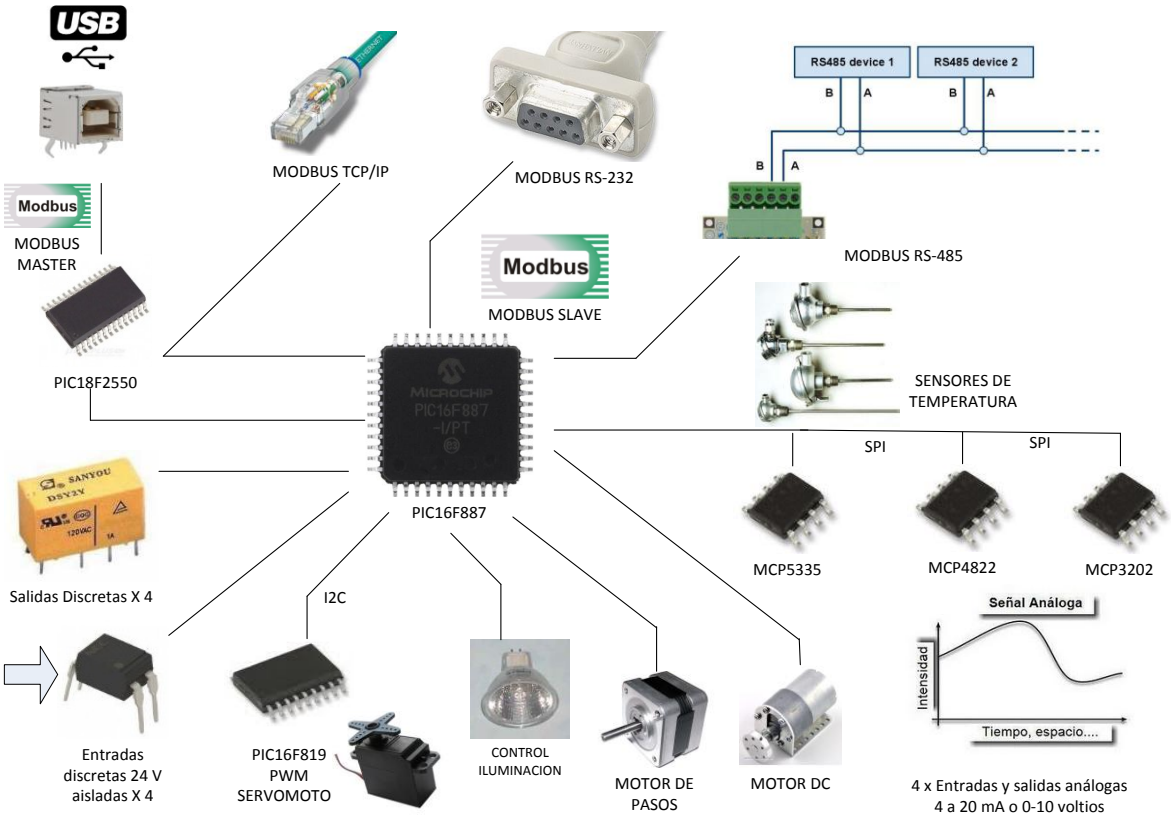


Figura 2.31. Esquema general Master de procesos

2.4.1. Circuito para control y monitoreo del Motor DC

Para poder realizar el control de velocidad del motor DC se utilizará una de las salidas PWM del microcontrolador usando el módulo CCP para este propósito, como driver se usará el integrado L293D[1], el cual por sus características, actúa como elemento de fuerza sobre el motor, la señal PWM esta optoacoplada con la ayuda de un PC817[2].

El siguiente circuito sirve para obtener el complemento de la señal PWM con la ayuda de un transistor 2N3904 (Q1), y luego ambas señales de modulación de ancho de pulso, la original y la complementada son optoacopladas con la ayuda de un PC817.

[1] *Ver en www.datasheetcatalog.net/es/datasheets_pdf/L/2/9/3/L293D.html las hojas de datos del CI L293D*
[2] *Ver en www.elektroda.net/pub/Karty%20katalogowe/pc817xx.pdf, las hojas de datos del CI PC817.*

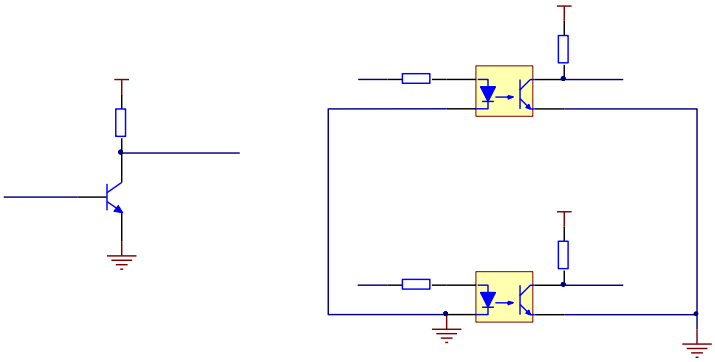


Figura 2.32. Circuito acondicionador de la señal de control.

Estas señales aisladas de la parte de control se conectan al integrado L293D para poder ejercer el control sobre el motor de corriente continua.

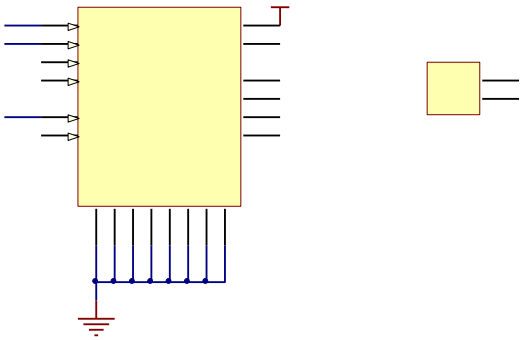


Figura 2.33. Circuito de fuerza para actuar sobre el motor DC.

Para realizar el monitoreo del motor DC se utilizará el timer 1 como contador, el cual recibirá los pulsos generados por el encoder que estará acoplado al eje del motor para poder determinar la velocidad a la que está girando este dispositivo motriz.

2.4.1.1. El circuito integrado L293D

El integrado L293D incluye cuatro circuitos para manejar cargas de potencia media, en especial pequeños motores y cargas inductivas, con la capacidad de controlar corriente hasta 600 mA en cada circuito y una tensión entre 4,5V a 36V.
Los circuitos individuales se pueden usar de manera independiente para controlar cargas de todo tipo y, en el caso de ser motores, manejar un único sentido de giro. Pero además, cualquiera de estos cuatro circuitos sirve para configurar la mitad de un puente H.

El integrado permite formar, entonces, dos puentes H completos, con los que se puede realizar el manejo de dos motores. En este caso el manejo será bidireccional, con frenado rápido y con posibilidad de implementar fácilmente el control de velocidad.

2.4.1.2. El optoacoplador PC817

Es un circuito integrado de 4 pines el cual en los pines 1 y 2 posee un led al cual al ser activado actúa sobre un optotransistor que está ubicado en los pines 3 y 4, para de esta manera lograr aislar los circuitos de control de los de fuerza.

2.4.2. Circuito para control y monitoreo del Motor de Pasos

Para poder realizar el control de un motor de pasos se utilizarán los 4 bits más significativos del puerto D, el cual está conectado a un driver ULN2003[1] el cual va a ser la parte de la fuerza dentro del control del motor de pasos.

Hay que tomar en cuenta que un motor de pasos cuenta con 4 bobinas las cuales deben ser activadas en forma secuencial para poder tener el desplazamiento giratorio paso a paso del motor.

El pin que va conectado a negativo a modo de punto común por medio de una resistencia de 20 ohmios, por medio de la cual se va a generar una corriente que vamos a enviarla a un circuito acondicionador con la ayuda de un 2N3904 para poder enviar los pulsos hacia el timer 0 del microcontrolador.

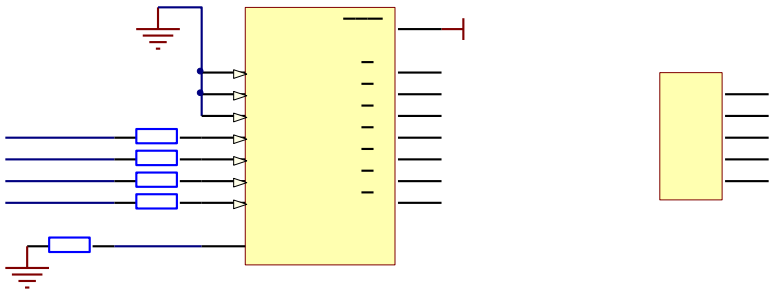


Figura 2.34. Circuito de fuerza para actuar sobre el motor de pasos

[1] *Ver en www.ti.com/lit/ds/symlink/uln2003a.pdf , las hojas de datos del CI ULN2003*

2.4.2.1. El circuito integrado ULN2003

El ULN2003 en un driver constituido por un arreglo de transistores Darlington tipo NPN la cual maneja alto voltaje y alta corriente.

La corriente de colector de cada darlington es de 500mA.

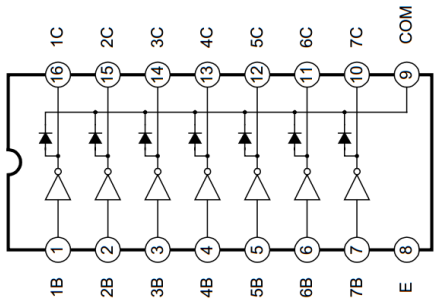


Figura 2.35. Diagrama de pines ULN2003

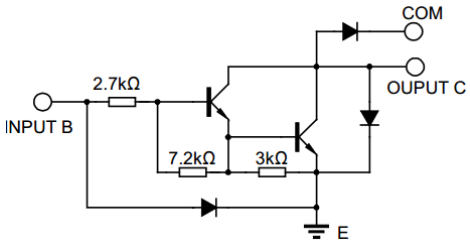


Figura 2.36. Esquemático equivalente de cada par de Darlington

2.4.3. Diagrama para control de un servomotor.

El servomotor[1] requiere de una señal que puede variar en su ancho de pulso para poder controlar su posición, es decir vamos a utilizar una de las salidas PWM para poder ejercer control sobre un servomotor, debido a que las señales de este tipo van a ser utilizadas para el control de iluminación y control de velocidad del motor DC y el microcontrolador cuenta solo con 2 módulos PWM, vamos a utilizar un PIC16F819[2] el cual va a estar configurado como un dispositivo I2C esclavo y va a recibir el dato de modulación de ancho de pulsos para poder ejercer control sobre el servomotor.

---

[1] *Revisar para más información acerca de los servomotores en http://www.todorobot.com.ar.*

[2] *Ver en anexo II, las hojas de datos del Microcontrolador PIC16F819*

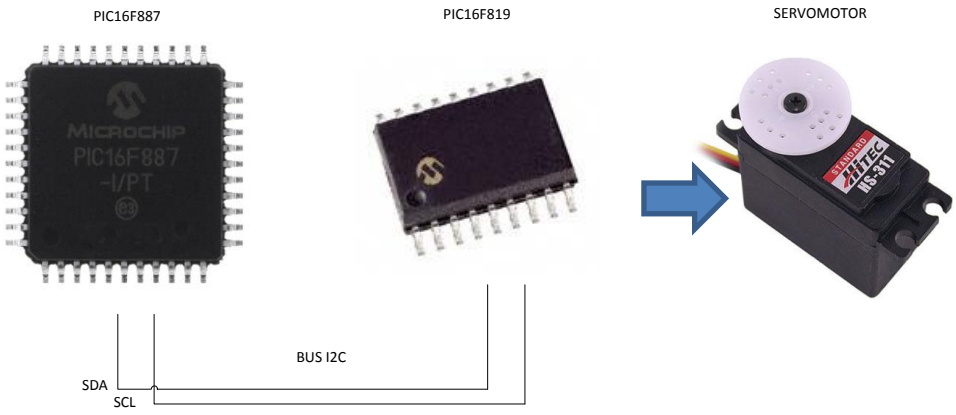


Figura 2.37. Esquema general para control de posición del servomotor

2.4.4. Circuito para control de iluminación.

Para el control de iluminación se va a utilizar una luminaria tipo dicroico de 50 vatios, y un transistor TIP122[1] como elemento actuador, la señal de control será tipo PWM aislada con la ayuda de un optoacoplador PC817.

El circuito de control y fuerza se presentan a continuación.

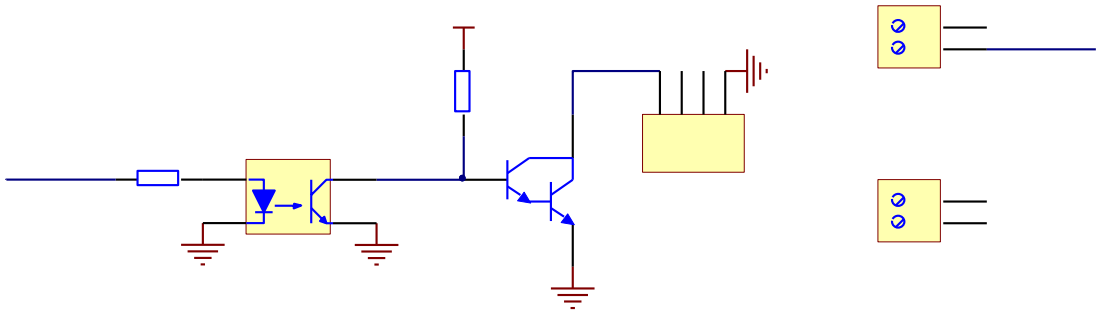


Figura 2.38. Diagrama de control y fuerza para control de iluminación.

2.4.5. Circuito acondicionador de señal RTD P100.

Para poder obtener un valor equivalente de temperatura obtenida con un sensor tipo RTD PT100[2], se utilizará un circuito integrado de Microchip llamado MCP3551, el cual es un conversor análogo digital tipo Delta-Sigma de 22 bits de resolución y trabaja como un dispositivo SPI esclavo.

---

[1] *Ver en www.fairchildsemi.com/ds/TI/TIP120.pdf, las hojas de datos del TIP122*
[2] *Ver en www.arian.cl/downloads/nt-004.pdf, las hojas de datos de una RTD PT100, y en capítulo 1 una reseña.*

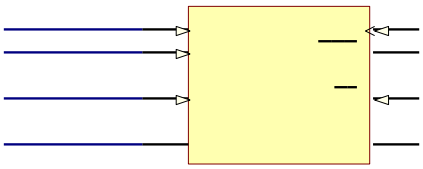


Figura 2.39. Distribución de pines circuito integrado MCP3553 o MCP3551

El circuito acondicionador es el siguiente:

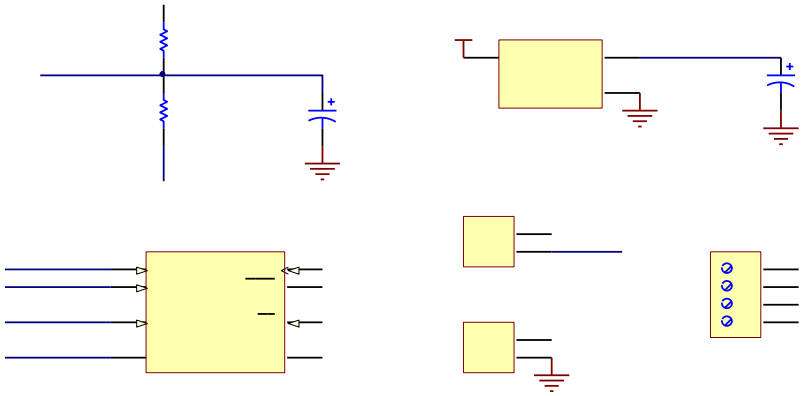


Figura 2.40. Circuito acondicionador de señal para RTD PT100

Se utiliza el regulador de 3.3 voltios de Microchip llamado MCP1702[1], el cual nos va a entregar los 3.3 voltios con los cuales se generará un divisor de tensión con 2 resistencias de 4k7 ohmios, este valor lo vamos a usar como referencia para la conversión.

El resultado es el siguiente, en el caso de tener el sensor a 0 grados centígrados.

$$V_{RTD} = 3.3Vx\frac{RTD}{9k4\Omega + RTD}$$

$$V_{RTD} = 3.3Vx\frac{100\Omega}{9k4\Omega + 100\Omega}$$

$$V_{RTD} = 34{,}7mV$$

Con ese valor como referencia se va a obtener los datos correspondientes a temperatura dados por el circuito acondicionador en función de la variación de la resistencia de la RTD.

[1] *Ver en anexo II, las hojas de datos del regulador de 3.3V MCP1702 de Microchip.*

Esta señal ingresa al canal análogo del MCP3551 y posteriormente el dato equivalente es enviado al microcontrolador vía SPI.

Al realizar la conversión con 22 bits de resolución podemos detectar pequeñas variaciones de voltaje y de esta manera se puede obtener una variación de datos muy útil para obtener el equivalente de temperatura.

2.4.6. Circuito acondicionador de señal para termocupla.

Para poder obtener un valor equivalente de temperatura obtenida con un sensor tipo termocupla[1], se utilizará un circuito integrado de Microchip llamado MCP3421[2], el cual es un conversor análogo digital tipo Delta Sigma[3] de 18 bits de resolución y trabaja como un dispositivo I2C esclavo.

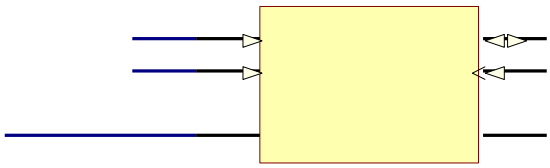


Figura 2.41. Distribución de pines circuito integrado MCP3421

Este integrado tiene un registro de configuración el cual nos permite darle una ganancia al valor de voltaje que ingresa por los pines VIN+ y VIN-, que es donde se conectará la termocupla, por ser de resolución de 18 bits, podemos obtener un buen rango de medidas para poder realizar la equivalencia entre voltaje y temperatura, su característica de ganancia configurable fue la razón por la cual elegimos este integrado para realizar esta labor de obtener el dato de la termocupla.

---

[1] *Ver en www.metring.com/notes/HI-10-10-MT2009.pdf, las hojas de datos de una Termocupla, y en capítulo 1 una reseña.*
[2] *Ver en anexo II, las hojas de datos del CI MCP3421 de Microchip.*
[3] *Revisar para más información acerca de la conversión Delta Sigma en http://www.todorobot.com.ar*

2.4.7. Circuito acondicionador de señal Celda de carga.

Para poder obtener un valor equivalente de peso obtenido con un sensor tipo celda de carga[1], se utilizará un circuito integrado de Microchip llamado MCP3551, el cual es un conversor análogo digital tipo Delta Sigma de 22 bits de resolución y trabaja como un dispositivo SPI esclavo.

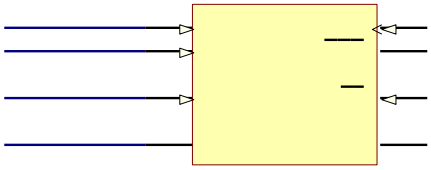


Figura 2.42. Distribución de pines circuito integrado MCP3553 o MCP3551

El circuito acondicionador es el siguiente:

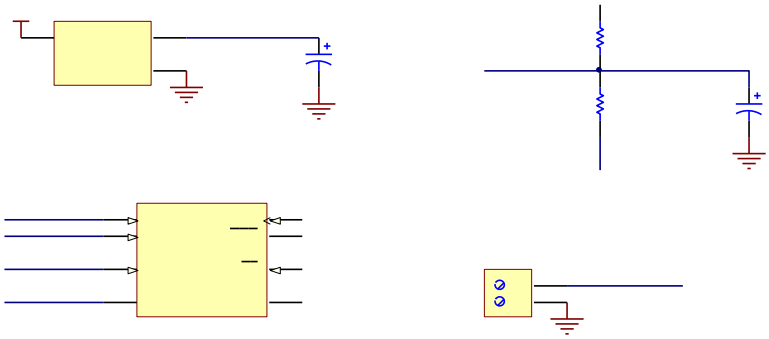


Figura 2.43. Circuito acondicionador de señal para Celda de Carga

Se utiliza el regulador de 3.3 voltios de Microchip llamado MCP1702, el cual nos va a entregar los 3.3 voltios con los cuales se generará un divisor de tensión con 2 resistencias de 4k7 ohmios, este valor lo vamos a usar como referencia para la conversión. El resultado es el siguiente, en el caso de tener el sensor a 0 gramos de peso.

$$V_{LC} = 3.3Vx\frac{Celda\ de\ carga}{9k4\Omega + Celda\ de\ carga}$$

[1] *Ver en anexo II, las hojas de datos de una Celda de carga, y en capítulo 1 una reseña.*

$$V_{LC} = 3.3Vx\frac{RX\Omega}{9k4\Omega + RX\Omega}$$

$$V_{LC} = Dato\ referencial$$

Con ese valor como referencia se va a obtener los datos correspondientes a peso dados por el circuito acondicionador en función de la variación de la resistencia de la Celda de carga.

Esta señal ingresa al canal análogo del MCP3551 y posteriormente el dato equivalente es enviado al microcontrolador vía SPI.

Al realizar la conversión con 22 bits de resolución podemos detectar pequeñas variaciones de voltaje y de esta manera se puede obtener una variación de datos muy útil para obtener el equivalente de peso.

Como se puede ver el circuito acondicionador es similar el que se utiliza para obtener el dato de la RTD, según el dato que nos entregue la celda de carga cuando está en vacío, se debe calibrar este sensor para poder obtener una medición real del peso.

2.4.8. Circuito para entradas discretas.

Se va a tener 4 entradas discretas de 24VDC, serán optoacopladas y estarán conectadas a los 4 bits menos significativos de puerto A del microcontrolador PIC16F887. El diagrama es el siguiente:

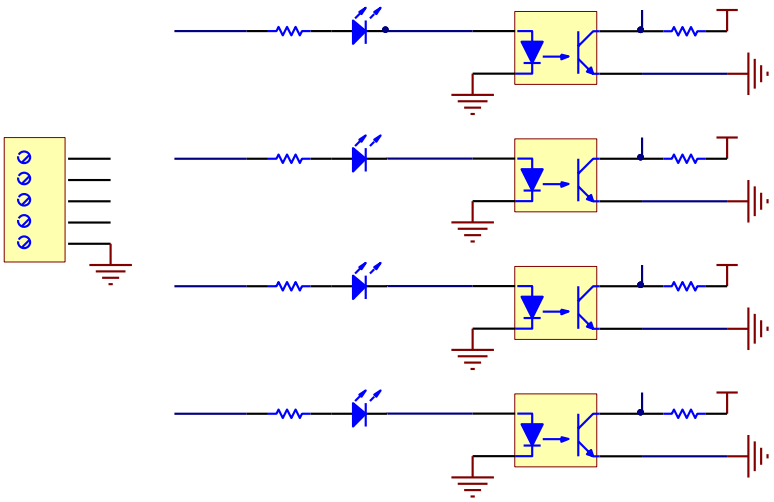


Figura 2.44. Circuito entradas discretas

2.4.9. Circuito acondicionador de entradas análogas.

Se va a tener 2 tipos de entradas análogas, uno de 0 a 10 voltios y otro de 0 a 20mA, las cuales van a ser conectadas al circuito integrado MCP3202, el cual trabaja como SPI esclavo y tiene una resolución de 12 bits.

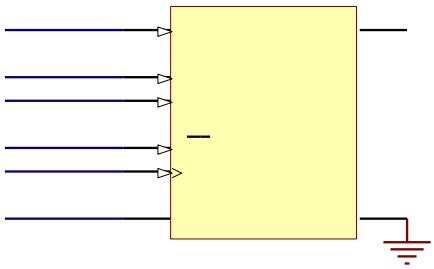


Figura 2.45. Distribución de pines circuito integrado MCP3202

El circuito acondicionador para entrada análoga de 0 a 20mA es el siguiente:

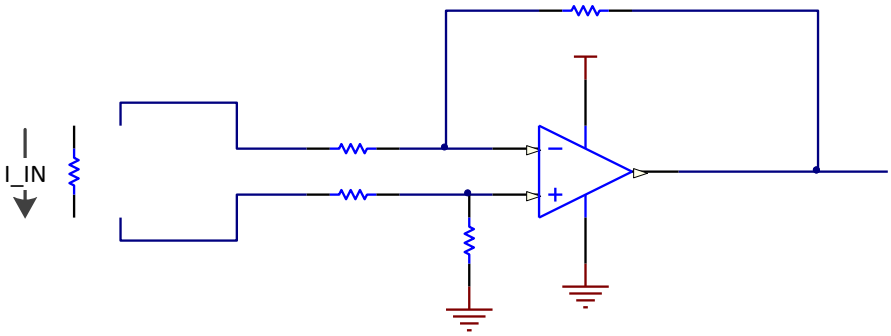


Figura 2.46. Circuito Acondicionador de Señal, Conversor de Corriente a Tensión.

En el circuito se puede observar el arreglo de un acondicionador de señal con la ayuda del amplificador operacional AD822[1], para convertir señales de tensión a corriente el cual ha sido aplicado a la tarjeta de entradas análogas. Los valores de resistencias han sido calculados de la siguiente manera:

[1] *Ver en anexo II, las hojas de datos del AO AD822.*

$$IA = \frac{V1 - \frac{R2}{R2 + R1} * V2}{R1} \qquad IA = IB \Rightarrow Vo = \frac{R2}{R1}\ V2 - V1$$
$$IB = \frac{\frac{R2}{R2 + R1} * V2 - Vo}{R2}$$

Por propósitos de cálculo tenemos que la corriente mínima de ingreso será 0 mA y la máxima será 20 mA asumiendo un valor de V1 = 0 VDC, entonces:

$$Vo = \frac{R2}{R1}\ V2$$

Por ley de ohm tenemos que:

$$V2 = I * R$$

Donde: *I = 20 mA y V2 = 4,095 VDC*

Entonces *R = 204,75 ohm.*

Como no deseamos amplificar la señal decimos que *R1 = R2,* entonces

$$Vo = \frac{R2}{R1}\ V2$$
$$Vo = \frac{1000\ ohm}{1000\ ohm}\ 4{,}095$$
$$Vo = 4{,}095\ VDC$$

La tensión que hemos obtenido de los cálculos es la que ingresar al MCP3202 para realizar las respectiva conversión análoga digital y ser entregada al microprocesador vía SPI.

Entonces el ciclo que se cumplirá es el siguiente:

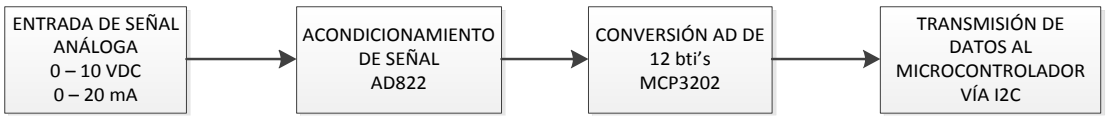


Figura 2.47. Proceso conversión análoga digital dentro de un MCP3202

Para obtener el dato de voltaje de 0 a 10 voltios, tan solo fue necesario hacer un divisor de tensión con un potenciómetro, con lo siguientes datos para poder obtener la relación de resistencias:

$$4{,}095 = 10\frac{R1}{R1 + R2}$$

$$Asumiendo\ R1 = 5k\ tenemos: 4{,}095 = 10\frac{5k}{5k + R2}$$

$$20{,}25k + 4{,}095kxR2 = 50k$$

$$R2 = \frac{29{,}75k}{4{,}095k}$$

$$R2 = 7{,}26k$$

Valores con los que se obtendrá un equivalente de 0 a 10 voltios para poder leer el dato de entrada análoga.

2.4.10. Circuito para salidas discretas.

Las 4 salidas discretas son de 24 voltios DC, éstas están conectadas a los 4 bits menos significativos del puerto D, y posteriormente se conectan a un ULN2803 el cual trabaja como actuador sobre la señal de voltaje que finalmente será entregada a la salida.

2.4.10.1. Circuito integrado ULN2803

Dentro del ULN2803[1] se encuentran 8 transistores NPN Darlington. Es un circuito integrado ideal para ser empleado como interfaz entre las salidas de un PIC o cualquier integrante de las familias TTL o CMOS y dispositivos que necesiten una corriente más elevada para funcionar, como por ejemplo, un relé.

Todas sus salidas son a colector abierto y se dispone de un diodo para evitar las corrientes inversas. El modelo ULN2803 esta especialmente diseñado para ser compatible con entradas TTL.

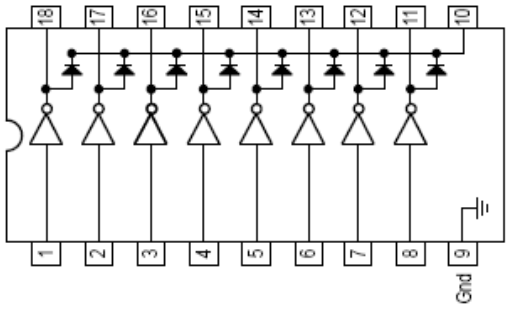


Figura 2.48. Distribución de pines del ULN2803

[1] *Ver en www.datasheetcatalog.net/es/datasheets_pdf/U/L/N/2/ULN2803-D.shtml, las hojas de datos del CI ULN2803*

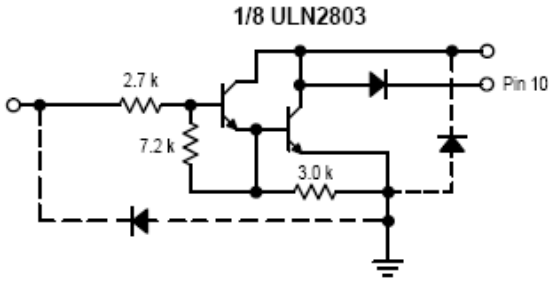


Figura 2.49. Esquemático equivalente de cada par de Darlington

A continuación el circuito diseñado para las 4 salidas discretas.

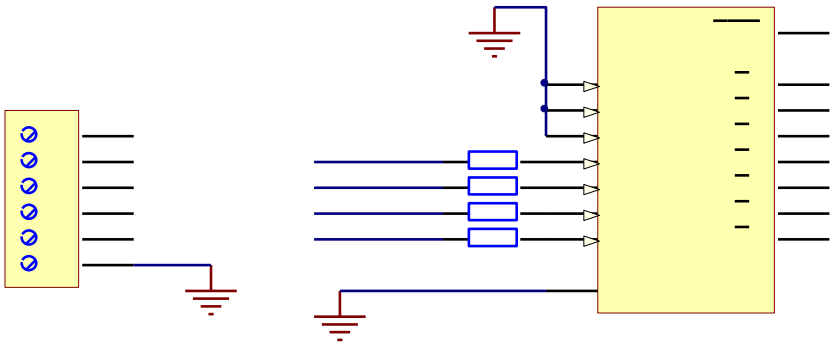


Figura 2.50. Circuito salidas discretas

2.4.11. Circuito acondicionador de salidas análogas

Se va a tener 2 tipos de salidas análogas, uno de 0 a 10 voltios y otro de 0 a 20mA, las cuales van a ser conectadas al circuito integrado MCP4822, el cual trabaja como SPI esclavo y tiene una resolución de 12 bits.

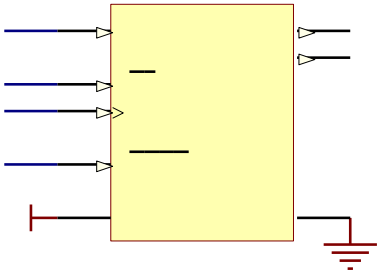


Figura 2.51. Distribución de pines circuito integrado MCP4822

El acondicionamiento de señal mediante el circuito integrado AD822 se lo realizara en dos etapas, la primera será la de una amplificación de tensión mediante un amplificador no inversor de tensión y la siguiente será un conversor de señal de tensión a corriente para generar la señal de 0 a 20 mA de salida de corriente.

A continuación se detalla el diseño de cada uno de los circuitos acondicionadores.

Anteriormente se hablo de que se requería una ganancia de 2.44, esta ganancia la lograremos mediante el circuito básico de un amplificador no inversor a través de amplificadores operacionales[1].

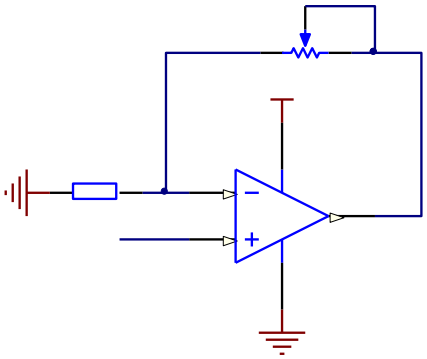


Figura 2.52. Amplificador no inversor

En el esquema podemos observar el arreglo típico de un amplificador no inversor, en el diagrama R2 es una resistencia variable la que usaremos con propósitos de calibración. Entonces, calculamos la ganancia requerida de la siguiente manera.

Vout_max_mcp4822 = 4,095 VDC.

Vout_max_requerida = 10 VDC.

Entonces ΔG = Vout_max_requerida / Vout_max_mcp4822.

ΔG = 2,44.

Se tiene que la fórmula del cálculo de resistencias en el en un circuito amplificador no inversor es la siguiente:

$$\Delta G = 1 + \frac{R2}{R1}$$

Done: R1 = 1000 ohm

Entonces: $2{,}44 = 1 + \frac{R2}{1000ohm}$

R2 = 1400 ohm.

[1] *Revisar para más información acerca de Amplificadores Operacionales en* http://www.ifent.org

Para realizar el acondicionamiento de tensión a corriente tomaremos los valores ya amplificados en el circuito anterior (amplificador no inversor), este nos entregara valores que se encontraran en rangos de 0 a 10 VDC los cuales serán transformados a señales de corriente para generar el lazo de corriente de 0 a 20 mA requerido.

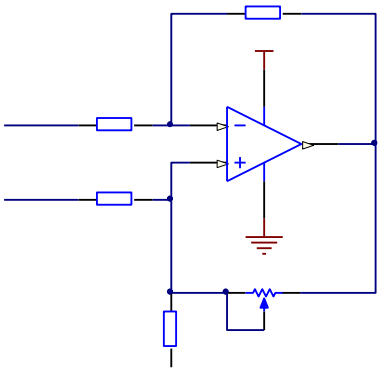


Figura 2.53. Acondicionador de señal de Tensión a Corriente

La fórmula que se utilizará para el cálculo de la corriente en función de la R1 es la siguiente:

$$I_{OUT} = \frac{1}{R1} V_{IN} - V_{REFF}$$

Donde: Vin = 0 a 10 VDC.

Vreff = 0 VDC (GND)

Iout (deseada) = 0 a 20 mA.

Entonces: R1 = 500 ohm.

Calculados ya los valores de resistencias a utilizarse en cada uno de los circuitos acondicionadores procedemos al acople de dichos circuitos al resto des sistema. El cual se encontrara interconectado de la siguiente manera.

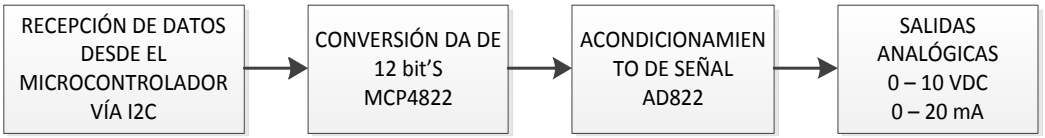


Figura 2.54. Proceso conversión digital análoga dentro de un MCP4822

2.5. Diseño de la Tarjeta de Entradas Discretas.

La tarjeta de entradas digitales[1] cumple la función de recibir señales discretas externas de niveles en rangos de 12 a 30 VDC para ser procesadas en la tarjeta, esta consta de 16 puntos de entradas discretas aisladas mediante opto acopladores de la serie PC817 como protección en caso de sobre voltajes o cortocircuitos.

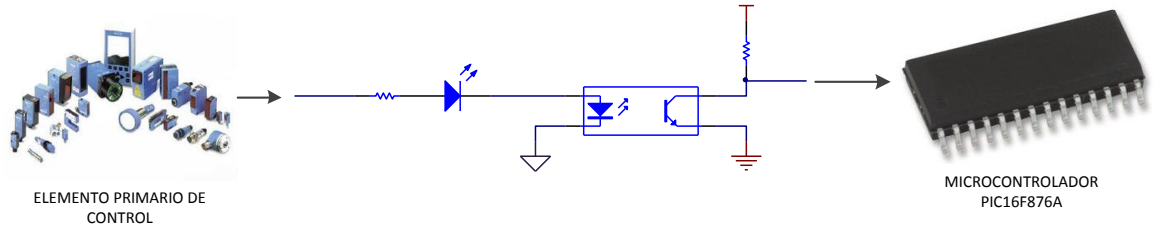


Figura 2.55. Diagrama de conexión de una entrada discreta aislada.

Como se puede observar en la figura se podrán conectar a la tarjeta elementos primario de control que entreguen señales de tensión en los niveles especificados anteriormente. El punto negativo de la entrada de bornera deberá ser puesto en común con la fuente de alimentación de los elementos primarios de control para tener un aislamiento completo de la tensión de entrada hacia la tarjeta de entradas digitales, esa realizado con el fon de evitar circulación de corrientes no deseadas a través de del punto negativo de la tarjeta. Cabe recalcar que los microcontroladores que se han utilizado para la adquisición de datos son dos PIC16F876A.

La tarjeta master de comunicaciones será la encargada de comunicarse vía I^2C con la tarjeta de entradas digitales, los dos microprocesadores tendrán direcciones fijas y distintas para realizar dicha comunicación y serán asignados como esclavos, de esta manera se tendrán los datos completos de las entradas en tiempo real.

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta MDE*

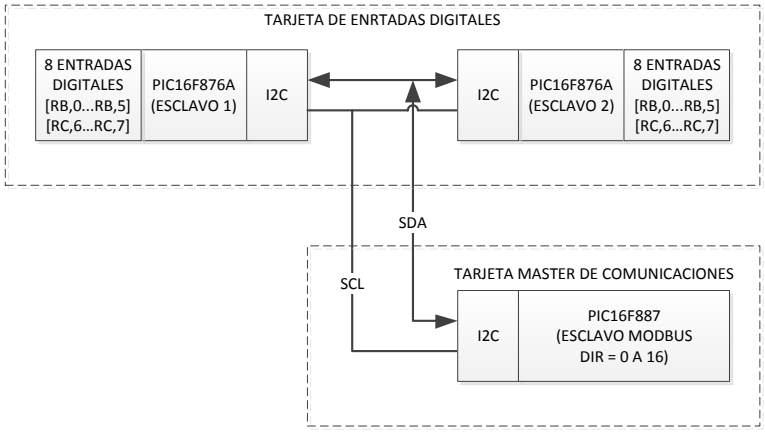


Figura 2.56. Diagrama de conexión I2C entre microcontroladores

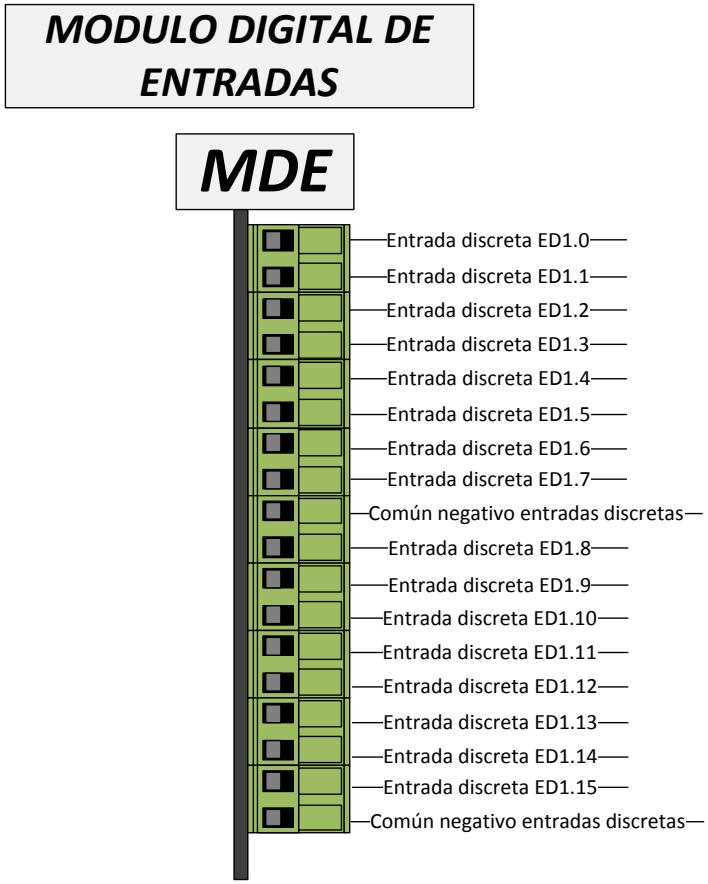


Figura 2.57. Vista frontal del Módulo de entradas discretas (MDE)

Ésta comunicación será llevada a cabo a través de la tarjeta base q lleva en si los “slots” en los que se alojarán las diferentes tarjetas que se comunican vía I^2C.

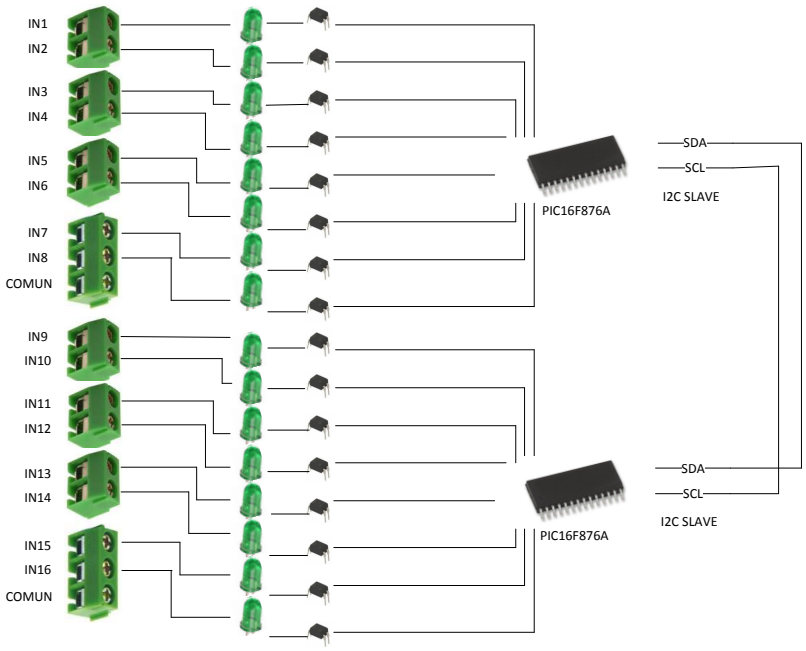


Figura 2.58. Esquema general de la tarjeta de entradas digitales

Se han dividido las entradas digitales en dos grupos de 8 entradas con su propio punto negativo el cual podría ser común si se coloca un puente externo entre los dos puntos de bornera.

Como se puede observar en la figura cada entrada consta de un diodo de tipo led como indicador de que existe un nivel de tensión diferente a cero en su punto de bornera correspondiente, pero deberá cumplir con los niveles de tensión mínimos y máximos para que funcionen correctamente.

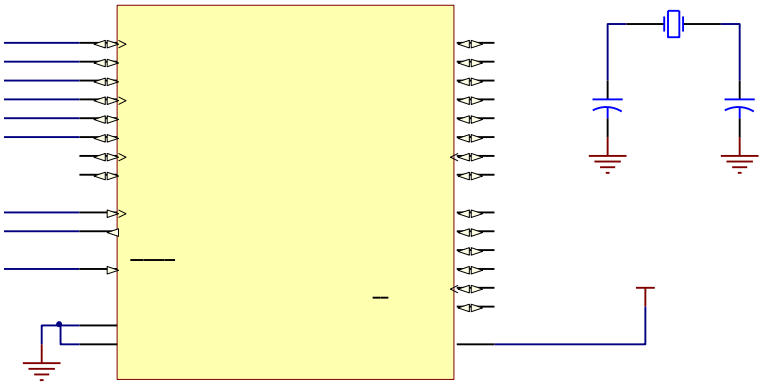


Figura 2.59. Microcontrolador PIC16F876A y distribución de pines

A continuación se describe la función que cumple cada un de los pines de los microcontroladores utilizados (se detallaran pines de un solo microcontrolador ya que para los dos microcontroladores se utiliza los mismos pines):

RB,0…RB,5	ESTRADAS 1…6	uC 1
	ENTRADAS 9…14	uC 2
RC,6…RC,7	ENTRADAS 7, 8	uC 1
	ENTRADAS 15, 16	uC 2
RC,3	SCL	uC 1, uC 2
RC,4	SDA	uC 1, uC 3

Tabla 2.3. Pines del microcontrolador para entradas discretas

2.6. Diseño de la Tarjeta de Salidas de Relé.

La tarjeta de salidas[1] cumple la función de activar o desactivar las salidas de relé, en total se tiene 8 salidas de relé aisladas mediante opto acopladores desde el micro hacia cada uno de los relés de la tarjeta, esta consta de 8 puntos de relé los cuales tendrán una capacidad nominal de 0,5 Amp. a 110 VAC y 1 Amp. a 24 VDC con accionamiento de 12 VDC.

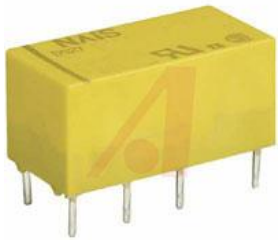


Figura 2.60. Relé 12 VDC.

Esta tarjeta tiene como soporte de comunicaciones un PIC16F876A el cual se encarga de la comunicación de la tarjeta con la tarjeta master de comunicaciones vía I^2C, al igual que los microprocesadores de la tarjeta de entradas digitales el microcontrolador tendrá una dirección I^2C previamente asignada el cual se encontrara en el mismo bus sobre la tarjeta base, back panel.

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta MDS*

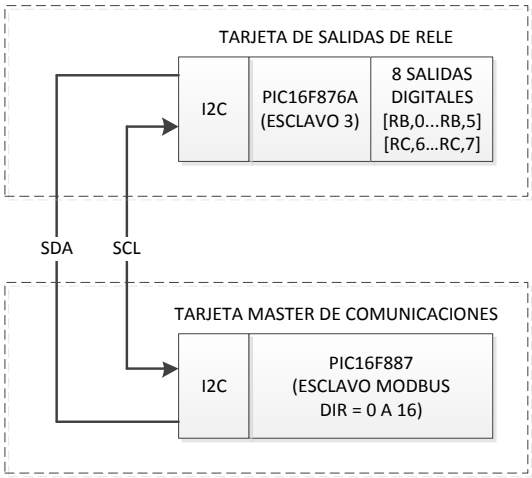


Figura 2.61. Diagrama de conexión I2C entre microcontroladores

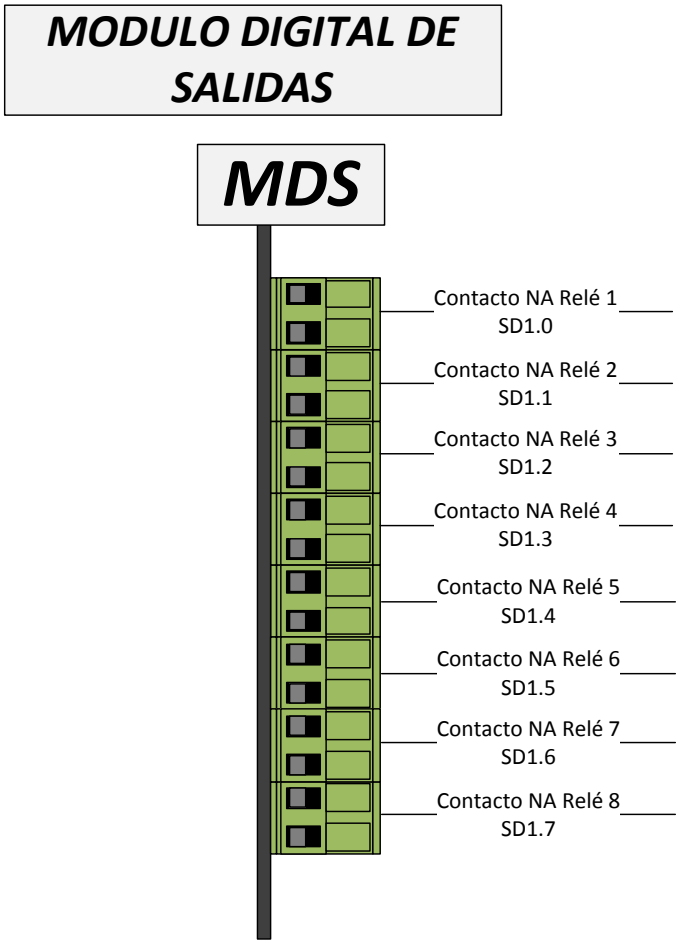


Figura 2.62. Vista frontal del Módulo de salidas discretas

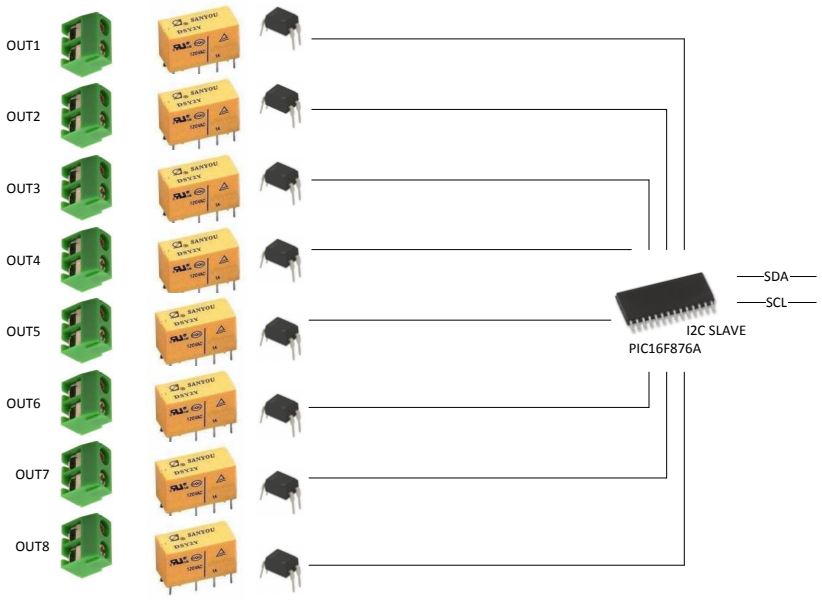


Figura 2.63. Esquema general de la tarjeta de salidas digitales

A continuación se describe la función que cumple cada uno de los pines de los microcontroladores utilizados (se detallaran pines de un solo microcontrolador ya que para los dos microcontroladores se utiliza los mismos pines):

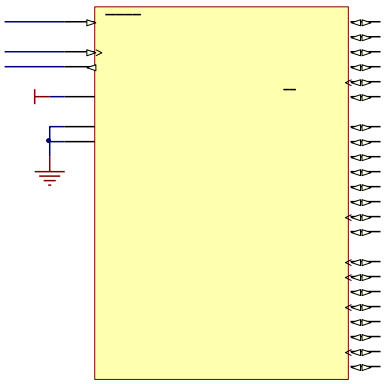


Figura 2.64. Microcontrolador PIC16F876A y distribución de pines

RB,0…RB,5	SALIDAS 1…6	uC 1
RC,6…RC,7	SALIDAS 7, 8	uC 1
RC,3	SCL	uC 1
RC,4	SDA	uC 1

Tabla 2.4. Pines del microcontrolador para salidas discretas

2.7. Diseño de la Tarjeta de Entradas Análogas.

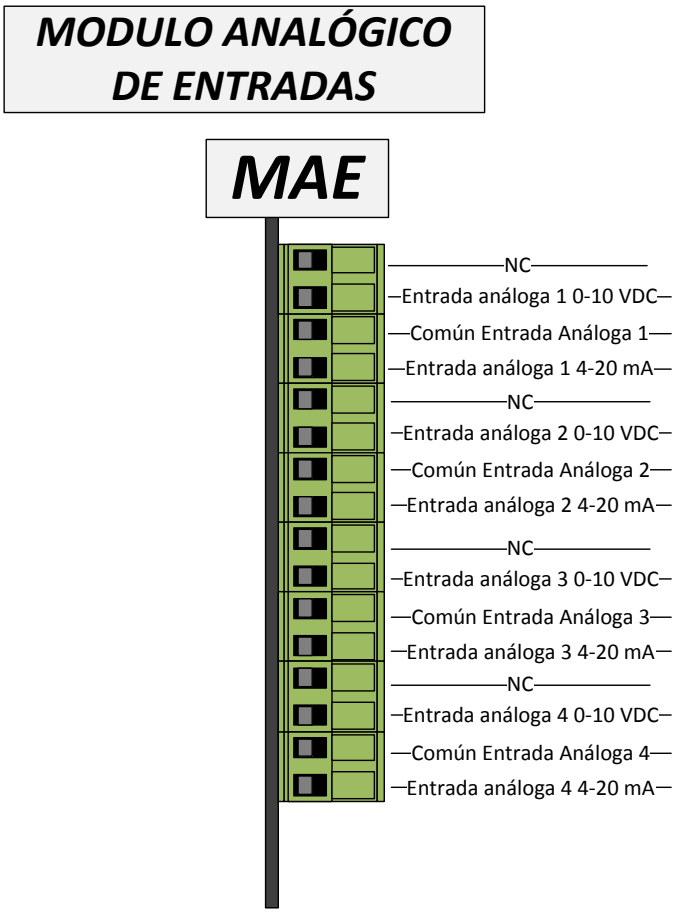


Figura 2.65. Vista frontal de la tarjeta de entradas análogas

La tarjeta de entradas análogas[1] a diferencia de las anteriores no lleva ningún microprocesador para realizar la adquisición de datos, en remplazo de este se han utilizado circuitos integrados de la industria Microchip Inc. los cuales se encargaran de realizar la conversión análoga/digital, los circuitos integrados utilizados en la tarjeta son los MCP3202 de los cuales hemos tratado anteriormente.

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta MAE*

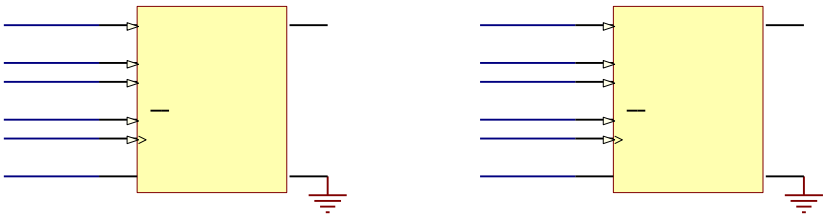


Figura 2.66. Integrados MCP3202 para entradas análogas

Para la construcción de la tarjeta de entradas análogas se han utilizado dos de estos circuitos integrados ya que estos llevan en si dos entradas análogas y la placa consta de cuatro entradas análogas.

Anteriormente se habló acerca de este circuito integrado, se decía que el circuito integrado constaba de dos canales análogos seleccionables, que el tipo de comunicación hacia el circuito integrado es a través de protocolo SPI, su resolución de 12 bit's, etc. Estos datos y otros lo hicieron el circuito integrado ideal para la aplicación.

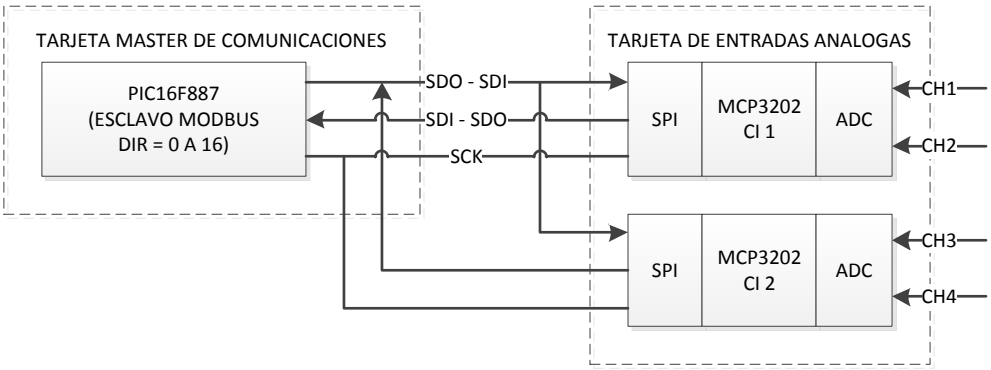


Figura 2.67. Diagrama de conexiones para comunicación SPI hacia la tarjeta de entradas análogas

El que el integrado nos entregue un dato con una resolución de 12 bit's nos quiere decir que tendremos valores desde 0 a 4095 por cada uno de los canales lo que nos ayuda a tener una mejor definición al momento de realizar las transformaciones respectivas en los niveles de las entradas análogas. El circuito integrado MCP3202 puede recibir como máximo el valor de 4,095 VDC que es el equivalente a los 12 bit's de resolución, por lo que se deberá diseñar un circuito acondicionador de señal para la recepción de señales análogas de tipo lazos de corriente y tensión ayudándonos del amplificador operacional de Analog Devices AD822.

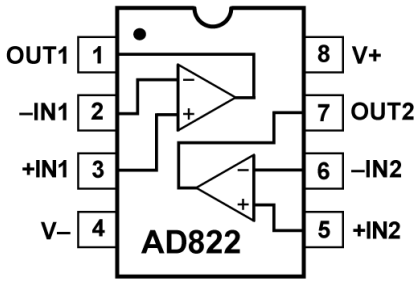


Figura 2.68. Circuito integrado AD822

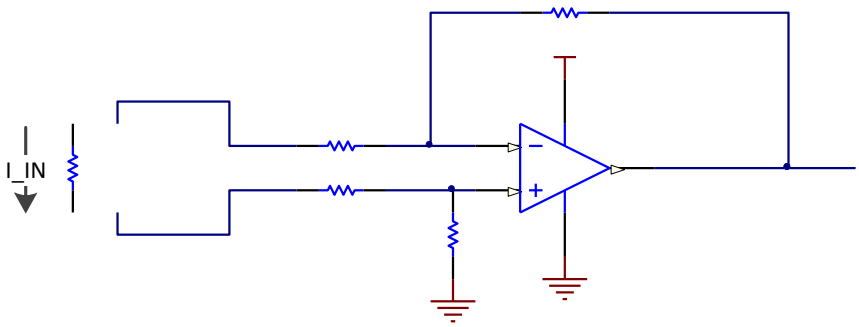


Figura 2.69. Circuito Acondicionador de Señal, Conversor de Corriente a Tensión.

En el circuito se puede observar el arreglo de un acondicionador de señal para convertir señales de tensión a corriente el cual ha sido aplicado a la tarjeta de entradas digitales. Los valores de resistencias han sido calculados de la siguiente manera:

$$IA = \frac{V1 - \frac{R2}{R2 + R1} * V2}{R1}$$

$$IB = \frac{\frac{R2}{R2 + R1} * V2 - Vo}{R2}$$

$$IA = IB \Rightarrow Vo = \frac{R2}{R1}\ V2 - V1$$

Por propósitos de cálculo tenemos que la corriente mínima de ingreso será 0 mA y la máxima será 20 mA asumiendo un valor de V1 = 0 VDC, entonces:

$$Vo = \frac{R2}{R1}\ V2$$

Por ley de ohm tenemos que:

$$V2 = I * R$$

Donde: *I = 20 mA y V2 = 4,095 VDC*

Entonces *R = 204,75 ohm.*

Como no deseamos amplificar la señal decimos que *R1 = R2,* entonces

$$Vo = \frac{R2}{R1}\ V2$$

$$Vo = \frac{1000\ ohm}{1000\ ohm}\ 4{,}095$$

$$Vo = 4{,}095\ VDC$$

La tensión que hemos obtenido de los cálculos es la que ingresar al MCP3202 para realizar las respectiva conversión análoga digital y ser entregada al microprocesador vía SPI.

Entonces el ciclo que se cumplirá es el siguiente:

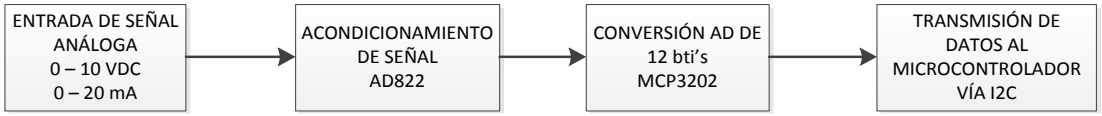


Figura 2.70. Proceso conversión análoga digital dentro de un MCP3202

Los lazos de corriente o señales de tensión pueden ser seleccionables mediante los jumpers que se encuentran montados sobre la tarjeta los cuales deberán ser habilitados como se muestra en la figura. En la placa nos encontraremos con cuatro jumpers de 3 puntos cada uno, estos deberán ser cortocircuitados para realizar la selección de tipo de señal que se desea medir (corriente o tensión) en cada uno de los puntos de bornera ya que se podrán realizar medidas te tensión y corriente según nuestras necesidades.

Figura 2.71. Jumper para selección de entrada de voltaje o corriente

Cabe recalcar que para el uso de estas entradas análogas se deberá poner en común los puntos negativos de las fuentes de alimentación tanto del módulo como de la fuente de señal analógica.

2.8. Diseño de la Tarjeta de Salidas Análogas.

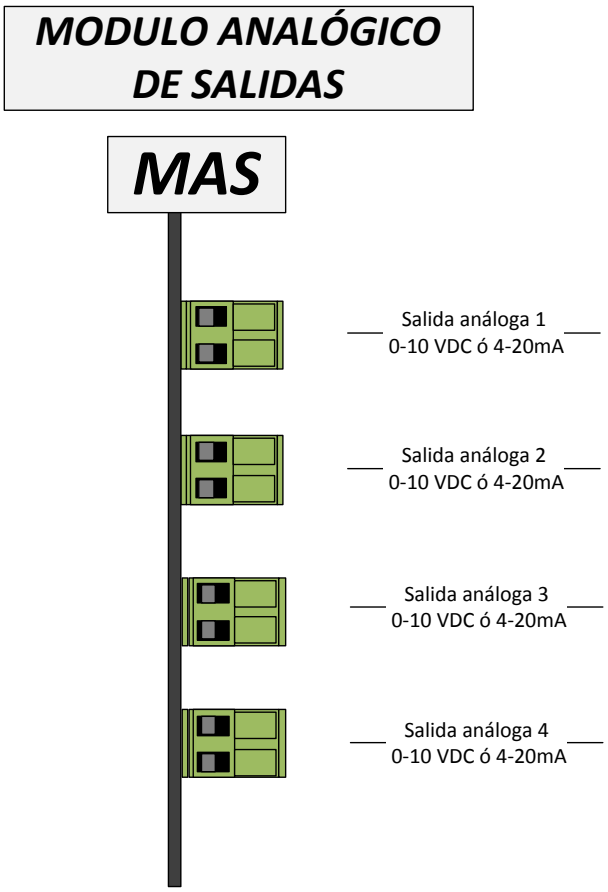


Figura 2.72. Vista frontal de la tarjeta de salidas análogas

Al igual que la tarjeta de entradas análogas[1] esta tarjeta no lleva microprocesador para realizar la adquisición de datos, en remplazo de este se han utilizado circuitos integrados de la industria Microchip Inc. los cuales se encargaran de realizar la conversión digital/análoga, los circuitos integrados utilizados en la tarjeta son los MCP4822 de los cuales hemos tratado anteriormente.

[1] *Ver en anexo I el diagrama esquemático y PCB para la tarjeta MAS*

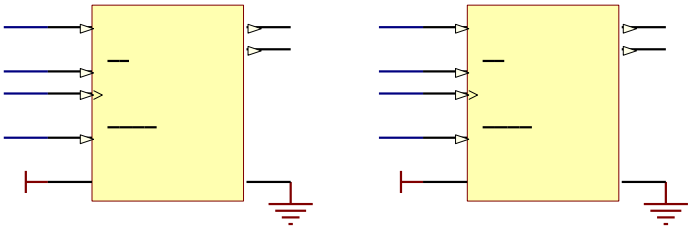


Figura 2.73. Integrados MCP4822 para entradas análogas

Para la construcción de la tarjeta de salidas análogas se han utilizado dos de estos circuitos integrados ya que estos llevan en si dos salidas análogas y la placa consta de cuatro salidas análogas..

Anteriormente se habló acerca de este circuito integrado, se decía que el circuito integrado constaba de dos canales análogos seleccionables, que el tipo de comunicación hacia el circuito integrado es a través de protocolo SPI, su resolución de 12 bit's, etc. Estos datos y otros lo hicieron el circuito integrado ideal para la aplicación.

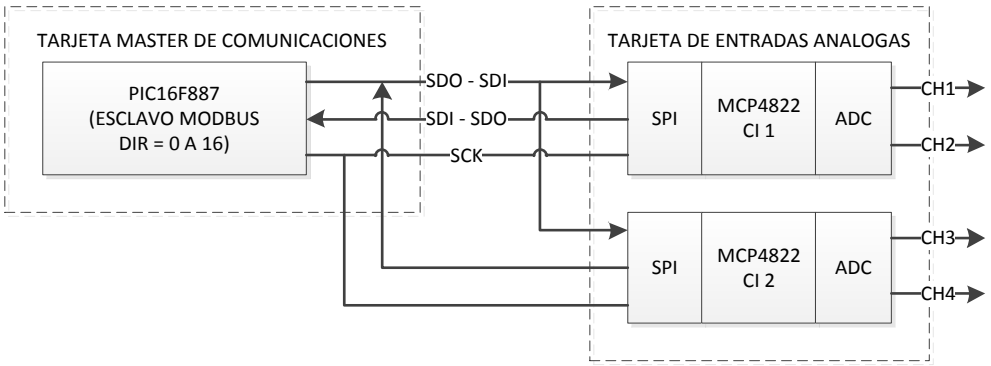


Figura 2.74. Diagrama de conexiones para comunicación SPI hacia la tarjeta de salidas análogas

Al circuito integrado le debemos entregar un dato de 12 bit´s lo que quiere decir que podremos generar variaciones en un rango de 0 a 4095 teniendo como valores análogos de salida valores 0 a 4,095 VDC. Para logra una salida de tensión de 10 VDC nos ayudaremos de un amplificador operacional AD822 de la industria Analog Devices los cuales cumplen con las características técnicas requeridas. Para el caso de las salidas análogas se requirieron 2 circuitos, un amplificador no inversor y un convertidor de voltaje a corriente.

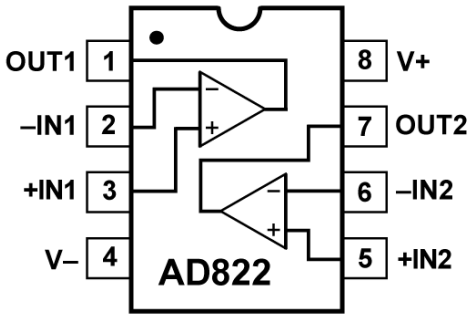


Figura 2.75. Circuito integrado AD822

El acondicionamiento de señal mediante el circuito integrado AD822 se lo realizara en dos etapas, la primera será la de una amplificación de tensión mediante un amplificador no inversor de tensión y la siguiente será un conversor de señal de tensión a corriente para generar la señal de 0 a 20 mA de salida de corriente.

A continuación se detalla el diseño de cada uno de los circuitos acondicionadores.

Anteriormente se habló de que se requería una ganancia de 2.44, esta ganancia la lograremos mediante el circuito básico de un amplificador no inversor a través de amplificadores operacionales.

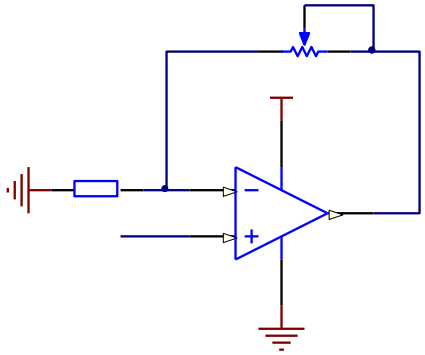


Figura 2.76. Amplificador no inversor

En el esquema podemos observar el arreglo típico de un amplificador no inversor, en el diagrama R2 es una resistencia variable la que usaremos con propósitos de calibración. Entonces, calculamos la ganancia requerida de la siguiente manera.

Vout_max_mcp4822 = 4,095 VDC.

Vout_max_requerida = 10 VDC.

Entonces ΔG = Vout_max_requerida / Vout_max_mcp4822.

$$\Delta G = 2{,}44.$$

Se tiene que la fórmula del cálculo de resistencias en el en un circuito amplificador no inversor es la siguiente:

$$\Delta G = 1 + \frac{R2}{R1}$$

Done: R1 = 1000 ohm

Entonces: $2{,}44 = 1 + \frac{R2}{1000ohm}$

R2 = 1400 ohm.

Para realizar el acondicionamiento de tensión a corriente tomaremos los valores ya amplificados en el circuito anterior (amplificador no inversor), este nos entregara valores que se encontraran en rangos de 0 a 10 VDC los cuales serán transformados a señales de corriente para generar el lazo de corriente de 0 a 20 mA requerido.

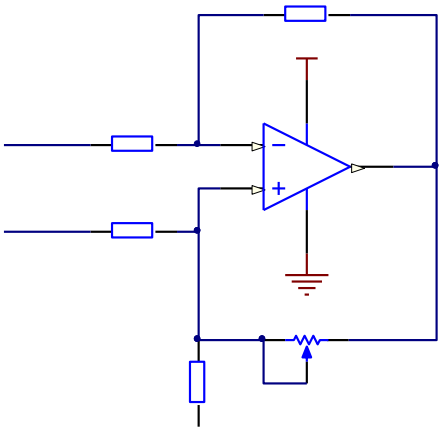


Figura 2.77. Acondicionador de señal de Tensión a Corriente

La fórmula que se utilizará para el cálculo de la corriente en función de la R1 es la siguiente:

$$I_{OUT} = \frac{1}{R1}\ V_{IN} - V_{REFF}$$

Donde: Vin = 0 a 10 VDC.

Vreff = 0 VDC (GND)

Iout (deseada) = 0 a 20 mA.

Entonces: R1 = 500 ohm.

Calculados ya los valores de resistencias a utilizarse en cada uno de los circuitos acondicionadores procedemos al acople de dichos circuitos al resto des sistema. El cual se encontrara interconectado de la siguiente manera.

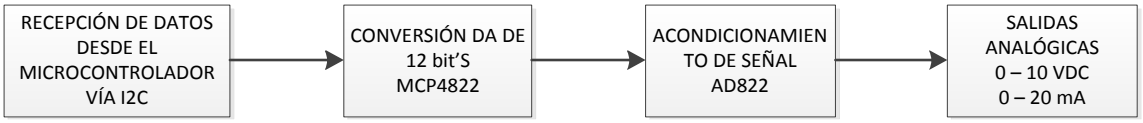


Figura 2.78. Proceso conversión digital análoga dentro de un MCP4822

Los lazos de corriente o señales de tensión pueden ser seleccionables mediante los jumpers que se encuentran montados sobre la tarjeta los cuales deberán ser habilitados como se muestra en la figura. En la placa nos encontraremos con cuatro jumpers de 3 puntos cada uno, estos deberán ser cortocircuitados para realizar la selección de tipo de señal que se desea enviar (corriente o tensión) a cada uno de los puntos de bornera.

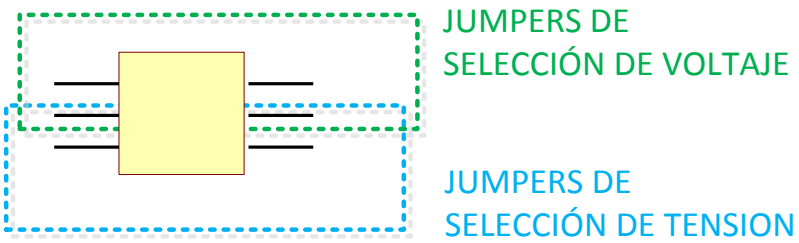


Figura 2.79. Jumper para selección de salida de voltaje o corriente

Cabe recalcar que para el uso de estas salidas análogas se deberá poner en común los puntos negativos de las fuentes de alimentación tanto del módulo como de la fuente de señal analógica.

CAPITULO III

DESARROLLO DEL SOFTWARE Y FIRMWARE

Introducción

Este capítulo corresponde al diseño y desarrollo del software y firmware requerido para el módulo de adquisición de datos. En él se expone el funcionamiento parcial y total de cada una de las subrutinas y programas principales.

Entre los programas realizados tenemos el que controla el Módulo Master de comunicaciones, el cual trabaja como Slave Modbus y Master SPI e I2C, y se encarga de recibir y enviar datos hacia los módulos esclavos tanto SPI como I2C que actúan como enlace entre las variables físicas y elementos primarios de control y el módulo de control y monitoreo.

En el mismo Módulo Master de comunicaciones tenemos un PIC18F2550 el cual tiene un programa que le permite actuar como Gateway entre USB-VISA y Modbus, ya que trabaja como maestro Modbus y a su vez como dispositivo USB-VISA.

Para el módulo Master de Proceso se ha creado un programa el cual trabaja como Slave Modbus y Master SPI e I2C lo cual permite adquirir datos de circuitos integrados tales como MCP3202, MCP4822, etc. Actuar sobre los motores, y demás salidas ya sean análogos o digitales.

En los módulos de entradas y salidas discretas se ha configurado los microcontroladores como Slave I2C con lo cual podemos enviar y recibir datos del microcontrolador Master I2C.

3.1. Configuración NI-VISA

Esta sección lo lleva por una serie de pasos para configurar su dispositivo USB RAW y así controlarlo por NI-VISA 3.0[1] en una PC basada en Windows. NI-VISA debe estar instalada en una PC, y el dispositivo USB no debe estar conectado. Además, no debe haber ningún controlador para el dispositivo USB instalado. Hay tres pasos para configurar su dispositivo USB y así utilizar NI-VISA:

Generar el archivo INF[2] utilizando el Driver Development Wizard (asistente para desarrollo de controladores).

Instalar el archivo INF y el dispositivo USB utilizando el archivo INF.

Probar el dispositivo con el Control Interactivo de NI-VISA.

3.1.1. Generar el Archivo INF Utilizando el Driver Development Wizard

Para utilizar NI-VISA, hay que indicarle a Windows que utilice NI-VISA como el controlador predeterminado para el dispositivo. En el entorno Windows, se puede hacer esto con un archivo INF. NI-VISA 3.0 y posteriormente incluir el VISA Driver Development Wizard (DDM) para crear el archivo INF el dispositivo USB.

Para abrir el DDW, seleccionar las opciones **Start»Programs»National Instruments»VISA»VISA Driver Developer Wizard**.

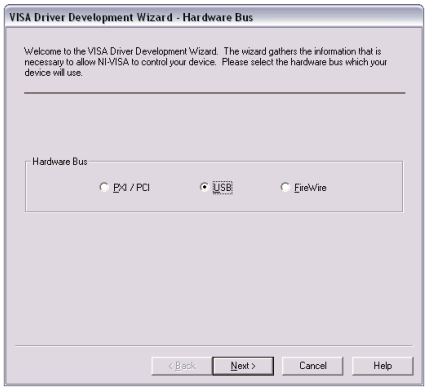


Figura 3.1. Ventana de Selección del Bus de Hardware en el VISA DDW

[1] *Revisar para más información acerca de NI-VISA http://digital.ni.com*
[2] *Un archivo generado por el programa Driver Wizard de NI-VISA es de extensión *.INF*

Se puede utilizar este asistente para crear el archivo para un dispositivo PXI/PCI, USB o IEEE 1394. En esta ocasión se está creando un controlador para un dispositivo USB, por lo tanto se debe hacer clic con el mouse en **USB** y **Next**, es decir, siguiente. La ventana de Información Básica del Dispositivo en el VISA DDW aparecerá, como se muestra en la siguiente figura.

Figura 3.2. Información Básica del Dispositivo en el VISA DDW

Para este paso, se debe conocer los números de identificación del proveedor y de producto del instrumento USB, en nuestro caso los números están dentro de la configuración del microcontrolador que actúa como Gateway entre Modbus y USB-VISA. Estos números identifican el dispositivo USB cuando se lo instala; también ubican el dispositivo cuando se desea establecer comunicación con él. De acuerdo a la especificación USB, ambos números son números hexadecimales de 16 bits. Estos números serán configurados y asignados al momento de desarrollar el firmare del microcontrolador.

Para confirmar que el dispositivo fue instalado correctamente. Ejecutar el Administrador de Dispositivos del Panel de Control y ubicar el dispositivo en la lista, generalmente se lo encontrará bajo "Otros Dispositivos." Podría mostrar una marca con un signo de exclamación amarillo indicando que es un dispositivo desconocido. Seleccionar la pestaña de Detalles y asegurarse de que "Identificador del Dispositivo" aparezca en la ventana de atributos desplegable. Los cuatro caracteres a la derecha de "VID_" y "PID_" son los números de identificación del proveedor y del producto, respectivamente. Anotar los caracteres

del dispositivo, cerrar el Administrador de Dispositivos y desconectar el dispositivo de la PC. Otra manera de conseguir estos números es contactando al proveedor del dispositivo.

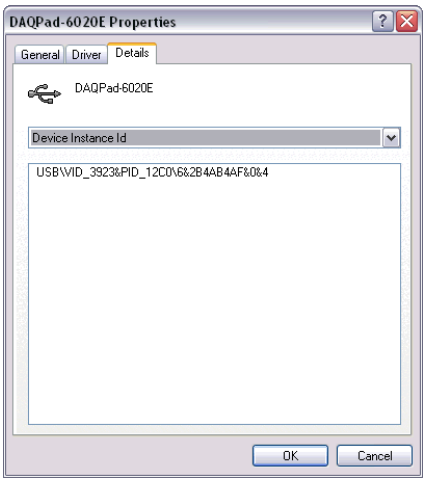


Figura 3.3. Números de Identificación del Proveedor y del Producto en el Administrador de Dispositivos

Nota: Antes de proceder con el Driver Development Wizard, asegurarse de que el dispositivo ha sido desconectado de la PC.

Introducir los números de identificación del proveedor y del producto, el nombre del fabricante y el nombre del modelo de su dispositivo en los campos adecuados.

Hacer clic en **Next**, siguiente. Se mostrará la ventana de propiedades de los archivos generados, como se muestra en la Figura 3.4.

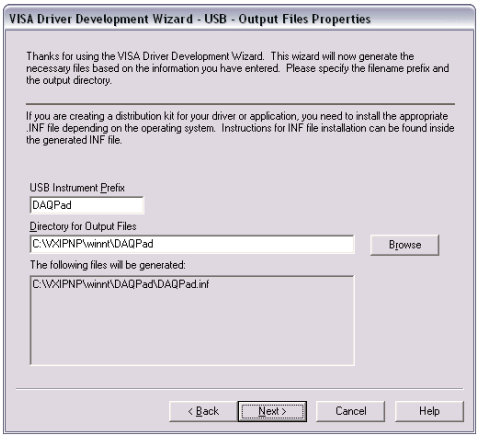


Figura 3.4. Ventana de Propiedades de Archivos Generados en el VISA DDW

El "USB Instrument Prefix", o prefijo de instrumento USB, es simplemente un descriptor que se utilizará para identificar los archivos utilizados para este

dispositivo. Introducir un prefijo de instrumento USB, seleccionar el directorio en el cual se desea ubicar estos archivos y hacer clic en **Next**, siguiente. La siguiente ventana proporcionará opciones de instalación. La selección predeterminada es instalar la información de configuración en el sistema operativo y por lo general es la mejor opción. Una vez que haya seleccionado una opción, haga clic en **Finish** para terminar de usar el asistente. El archivo INF se habrá generado en el archivo se especificó en el campo de directorio de archivos generados en la ventana previa.

3.1.2. Instalar los archivos INF y el dispositivo USB.

La instalación de los archivos INF es diferente para cada versión de Windows. Cuando el DDW crea un archivo INF, las instrucciones de instalación se incluyen en el encabezado al principio de cada archivo INF. Debido a que los archivos INF son archivos de texto ASCII, se pueden leer utilizando cualquier editor de texto tal como Notepad. Para obtener información detallada sobre la instalación de su archivo INF, abrir el archivo INF en un editor de texto y siga las instrucciones que aparecen al principio del archivo.

- Copiar el archivo INF a la carpeta INF. Esta carpeta generalmente se encuentra ubicada en **C:\WINDOWS\INF**. Esta carpeta puede estar escondida, por lo que podría necesitar cambiar las opciones de carpeta para ver los archivos escondidos.
- Hacer clic derecho con el mouse en el archivo INF en **C:\WINDOWS\INF** y hacer clic en **Instalar**. Este proceso crea un archivo PNF para su dispositivo. Ahora está listo todo para que usted instale su dispositivo USB.
- Conectar el dispositivo USB. Windows debería detectar el dispositivo USB, y el Asistente de Agregar Nuevo Hardware debería aparecer automáticamente tan pronto como se conecte el dispositivo al puerto USB. Seguir las instrucciones en la pantalla del asistente. Cuando se solicite seleccionar un controlador para este dispositivo, buscar la

carpeta INF y seleccionar el archivo INF que se generó utilizando el DDW.

Se debe tomar en cuenta que en algunos casos, Windows puede tener un controlador ya predeterminado asociado con el dispositivo USB. Si este es el caso, Windows intentará instalar ese controlador primero. Una vez que se haya conectado el dispositivo USB y Windows haya instalado el controlador predeterminado, hacer clic derecho en Mi PC y seleccionar Propiedades. En la ventana de Propiedades, en la pestaña de Hardware hacer clic en Administrador de Dispositivos. Una vez que la ventana de Administrador de Dispositivos esté abierta, expandir el menú de "Dispositivos de Interfaz Humana." Después ubicar qué instancia de "Dispositivos de Interfaz Humana USB" corresponde al dispositivo USB. Para lograr esto se debe hacer clic derecho, seleccionar Propiedades y seleccionar la pestaña de Detalles para encontrar el VID y PID que coincida con el dispositivo USB.

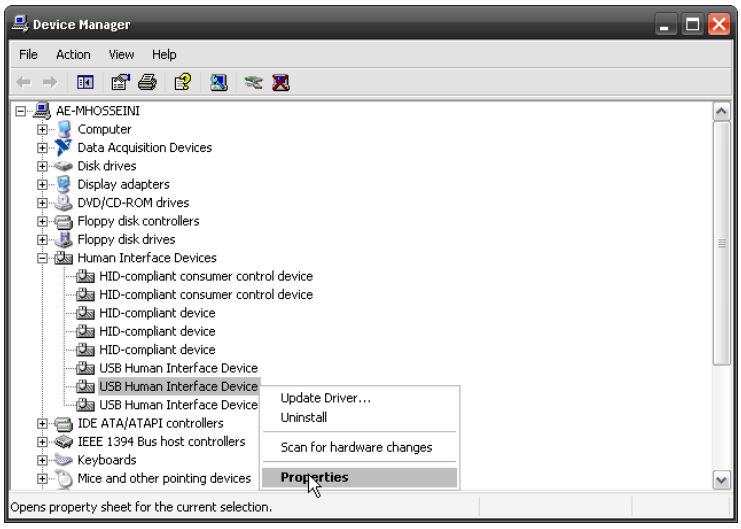


Figura 3.5. Encontrando la Instancia Correcta para el Dispositivo de Interfaz Humana USB

Una vez que se ha encontrado el "Dispositivo de Interfaz Humana de USB" cuyo VID y PID coinciden con el del Dispositivo USB, hacer clic derecho y seleccionar la opción Actualizar Controlador del menú desplegable, como se muestra en la Figura.

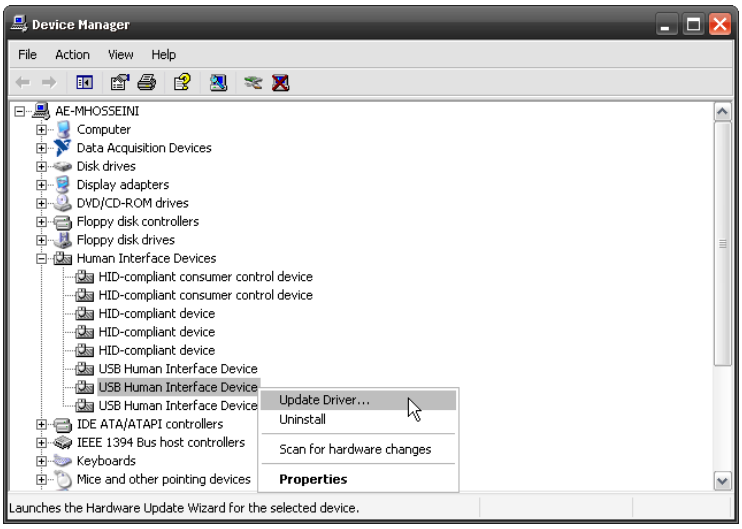


Figura 3.6. Actualice el Controlador de su Dispositivo USB

En la primera pantalla, seleccionar "Por ahora no" y después hacer clic en Siguiente. En la segunda pantalla, seleccione "Instalar desde una lista o una ubicación específica (Avanzado)" y después hacer clic en Siguiente. En la tercera pantalla, seleccionar "No buscar. Elegir el controlador para instalar." La cuarta pantalla será similar a la que se muestra en la Figura, en la cual el controlador marcado es el controlador que se creó. Una vez que ya haya seleccionado su controlador, hacer clic en Siguiente. Cuando el controlador se haya terminado de instalar, hacer clic en Terminar.

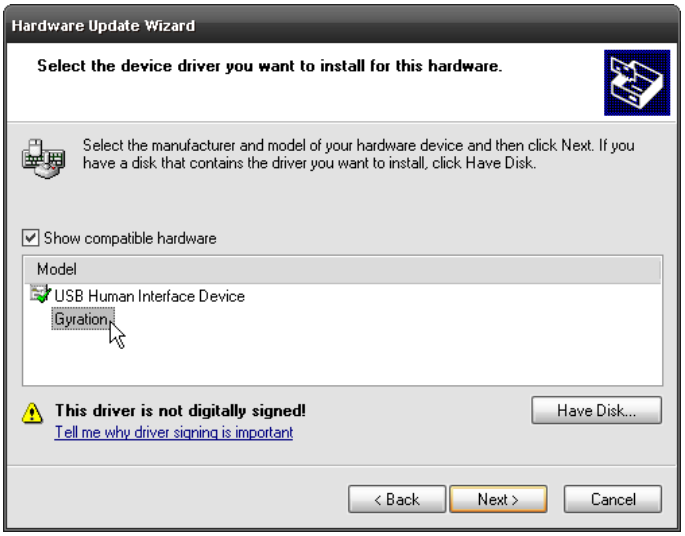


Figura 3.7. Seleccione el Controlador de su Dispositivo USB

3.1.3. Probar la Comunicación con VISA Interactive Control.

- Abrir Measurement & Automation Explorer (MAX).
- Seleccionar **Tools»Refresh** para actualizar la visualización. El dispositivo USB debería aparecer en la lista como Dispositivo USB bajo **Devices and Interfaces**, dispositivos e interfaces, como se muestra en la Figura. El dispositivo USB ahora está instalado y configurado para utilizar NI-VISA. Si seleccionamos el dispositivo USB, la información del dispositivo aparecerá en la ventana de configuraciones, USB Settings. Al utilizar esta ventana, se puede tener acceso a información tal como número de identificación del fabricante, código del modelo y número de serie de su dispositivo.

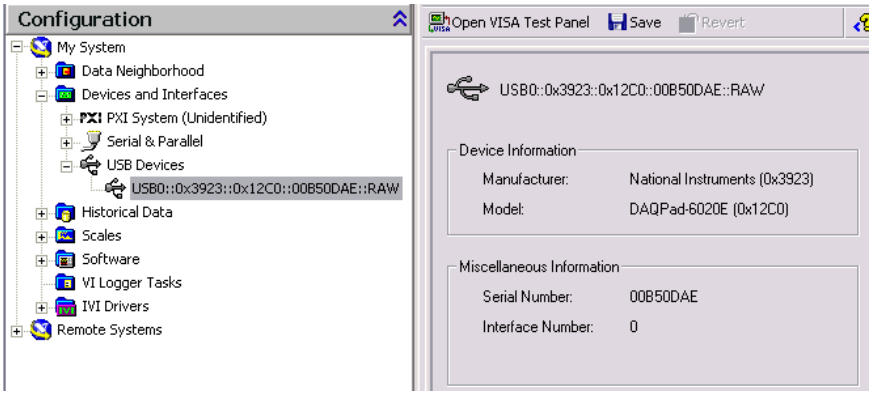


Figura 3.8. Dispositivo USB Mostrado en MAX

- Para comunicarse con el dispositivo utilizando VISA, utilizar el descriptor de instrumentos VISA del dispositivo. El formato del descriptor de instrumentos para un dispositivo USB INSTR es USB[tarjeta]:: identificación del fabricante:: código de modelo:: número de serie[:: USB número de interfaz]::INSTR. El formato del descriptor de instrumentos para un dispositivo USB RAW es USB[tarjeta]:: identificación del fabricante:: código de modelo:: número de serie[:: USB número de interfaz]::RAW.
- De acuerdo a la especificación USBTMC, todos los dispositivos USBTMC deben tener un número de serie. Algunos dispositivos RAW USB pueden no tener número de serie. Si el dispositivo no tiene un número de serie, NI-VISA

automáticamente asigna un número de serie de VISA específico para ese dispositivo. El formato del número de serie es NI-VISA-#, donde # es un número generado automáticamente.

- Algunos dispositivos USB tienen múltiples interfaces. Esto es similar a la manera en que un dispositivo PCI puede tener múltiples funciones. Si el dispositivo solamente puede utilizar una interfaz, no se necesita incluir el número de interfaz USB.
- Para probar la comunicación con este dispositivo, abrir MAX. Seleccionar **Tools»NI-VISA»VISA Interactive Control**. Una ventana similar a la mostrada en la Figura deberá aparecer.

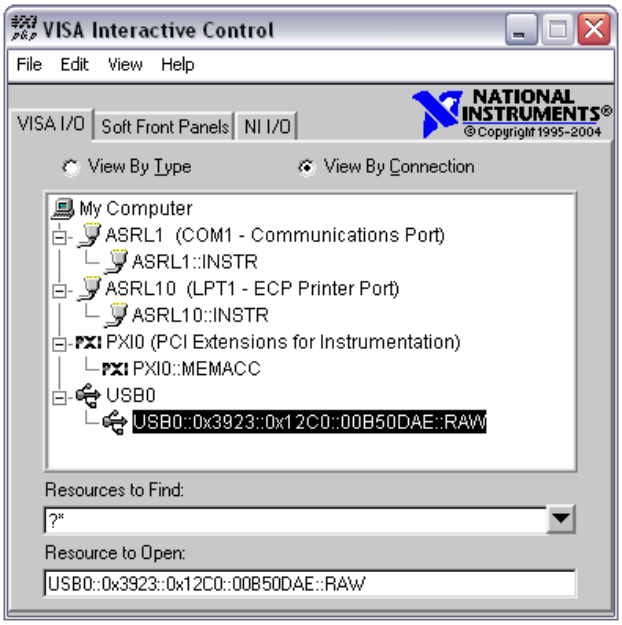


Figura 3.9. Control Interactivo VISA

- El Control Interactivo VISA (VISAIC) es un programa de herramientas utilizado para comunicarse fácilmente con cualquier recurso VISA. Después de haber configurado el dispositivo USB para utilizar VISA, éste debe aparecer en la lista de la rama USB. Hacer doble clic en el ícono del dispositivo para iniciar una sesión VISA con el dispositivo. La ventana mostrada en la Figura siguiente deberá aparecer.

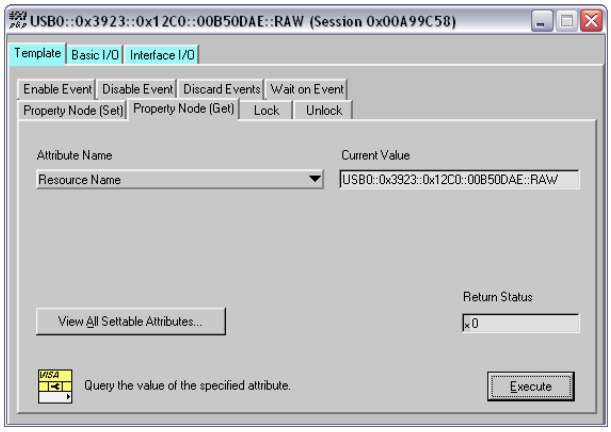


Figura 3.10. Sesión VISA Iniciada en el Control Interactivo VISA

3.2. Measurement & Automation Introducción

Figura 3.11. Programa Measurement & Automation (MAX)

"Measurement and Automation Explorer" (MAX), es un software utilizado para configurar dispositivos e instrumentos, también es utilizado para probar que el dispositivo funcione correctamente.

Abrir el programa "Measurement and Automation Explorer" (MAX) haciendo doble clic sobre el icono del escritorio.

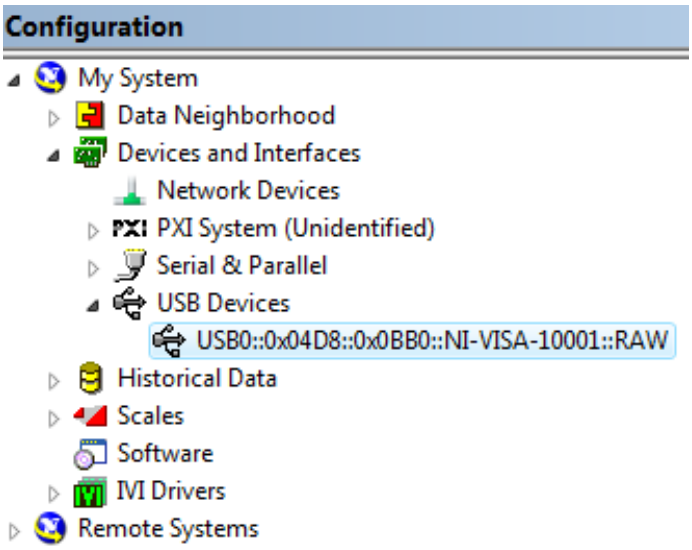


Figura 3.12. Explorador de dispositivos

En la imagen se muestra el explorador de dispositivos del programa Measurement & Automatión, como se puede apreciar una vez que le dispositivo externo se conecta, el explorador lo detecta en la parte USB Devices y se muestran los valores con los que éste fue configurado.

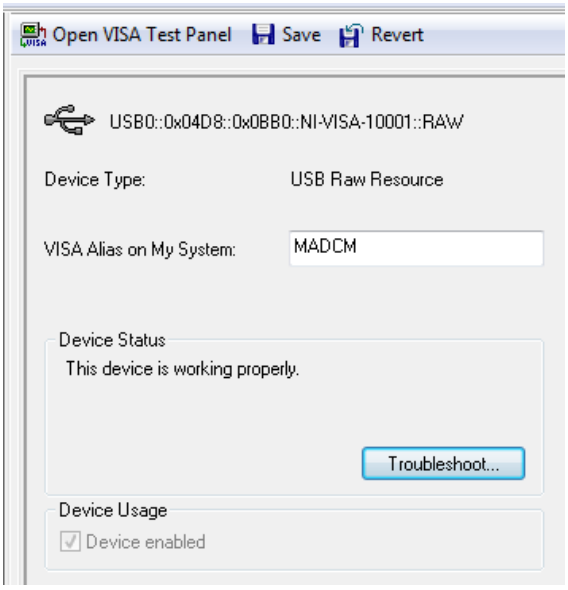


Figura 3.13. Características del dispositivo NI-VISA instalado

En la parte derecha del explorador, se muestra el nombre del dispositivo conectado y un botón que sirve para probar la comunicación, así como también los detalles del driver y versión del dispositivo.

3.3. Establececiendo la comunicación entre el Hardware y Labview vía USB.

Una vez que el dispositivo ha sido correctamente instalado y reconocido como un dispositivo USB-VISA es hora de realizar una aplicación en Labview para poder realizar control y monitoreocon nuestro Módulo de adquisisción de datos controlo y monitoreo.

Dentro de la pantalla de diagrama de bloques de Labview vamos a seleccionar el siguiente bloque.

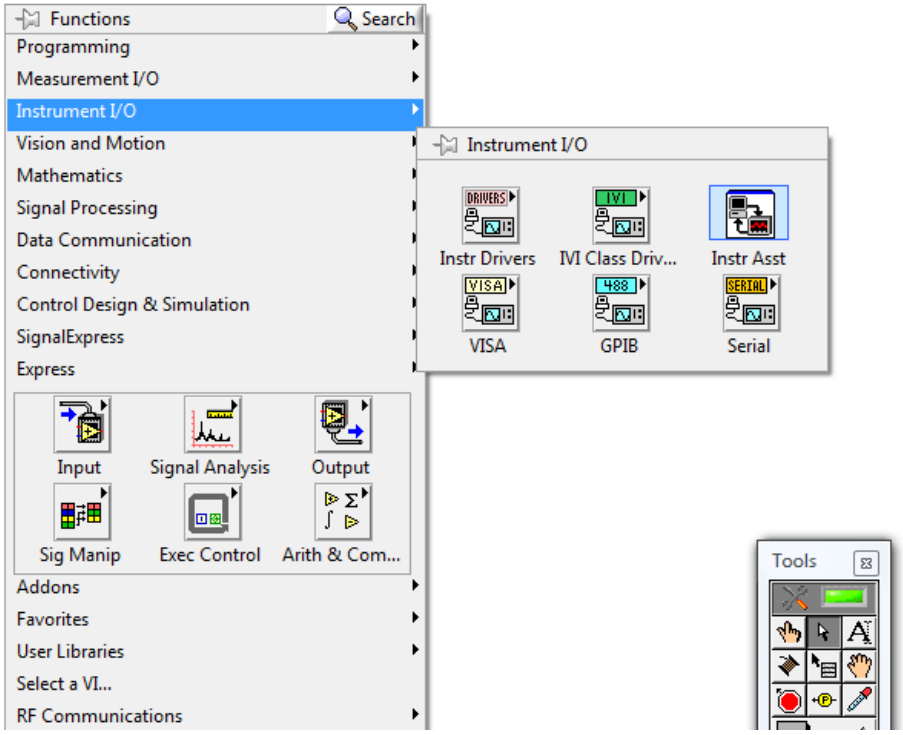


Figura 3.14. Seleccionar el Bloque Instrument I/O Assistant

3.3.1. Bloque Instrument I/O Assistant.

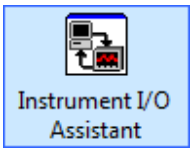


Figura 3.15. Bloque Instrument I/O Assistant de Labview

Es un bloque VI que permite la comunicación con un equipo externo ya sea serial, Ethernet o USB, y organiza la comunicación en pasos ordenados para lo cual se debe agregar una secuencia dentro de la configuración del bloque Instrument I/O Assistant.

Para empezar a configuración del Instrument I/O Assistant, es necesario dar doble click sobre él y se obtendrá la siguiente ventana de configuración.

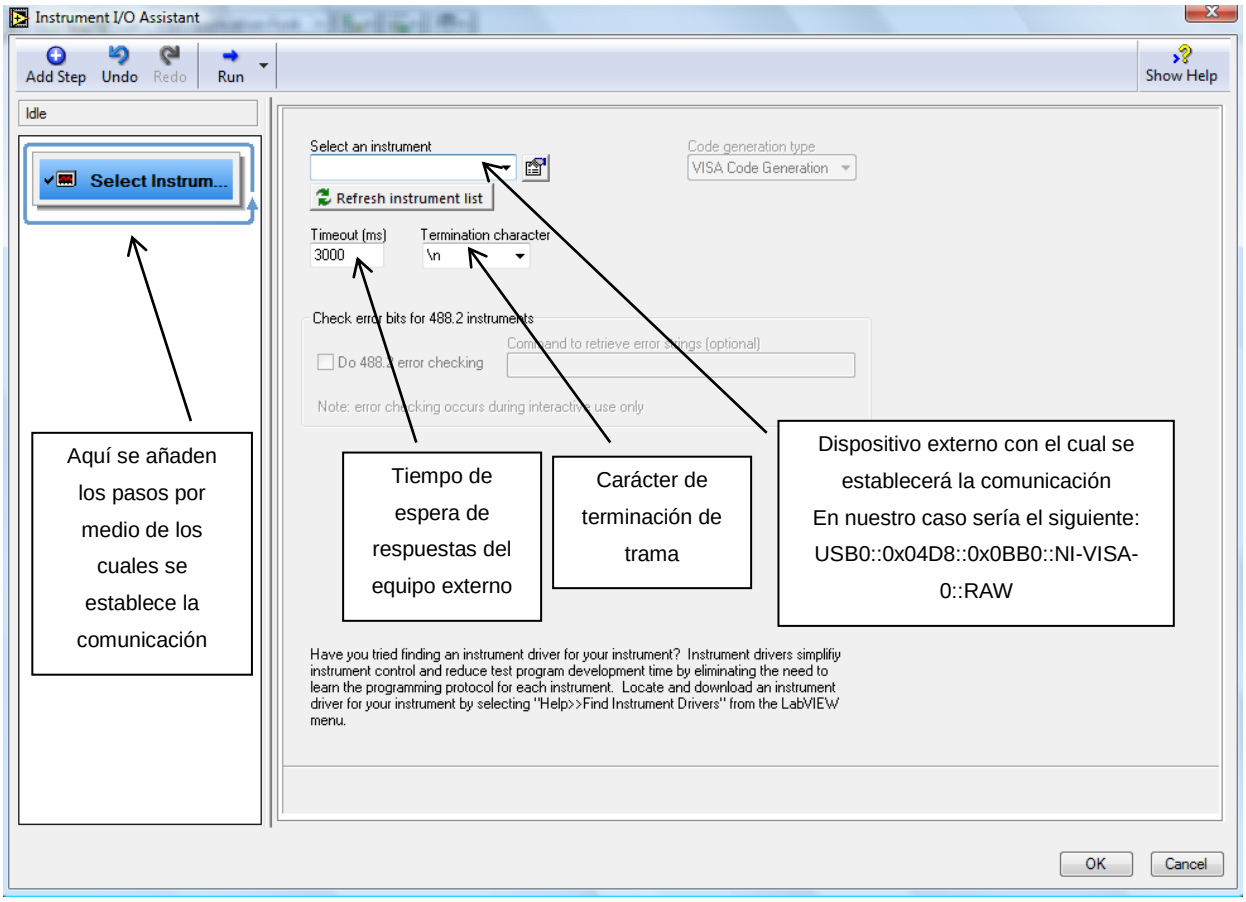


Figura 3.16. Cuadro de configuración principal Instrument I/O Assistant.

Tomar en cuenta lo siguiente, para establecer el nombre del dispositivo externo establece comunicación con la ayuda del driver de USB-VISA.

El nombre del dispositivo externo se escribe de la siguiente manera:

USB0::0x04D8::0x0BB0::NI-VISA-0::RAW

Donde:

USB0: Indica que la comunicación se la realizará vía USB hacia.

:0x04D8: VID del dispositivo, dado en forma hexadecimal.

:0x0BB0: PID del dispositivo, dado en forma hexadecimal.

:NI-VISA-0: Indica que utiliza el driver creador por NI-VISA.

:0: Número de serie que asigna VISA al dispositivo una vez que se lo conecta.

:RAW Dispositivo USB de tipo RAW.

3.3.2. Proceso de creación del bloque para entradas y salidas con Instrument I/O Assistant

A continuación vamos a añadir pasos, que se cumplirán secuencialmente dentro del bloque de Instrument I/O Assistant.

Para esto se debe hacer clic derecho sobre el cuadro que dice Select Instrument.

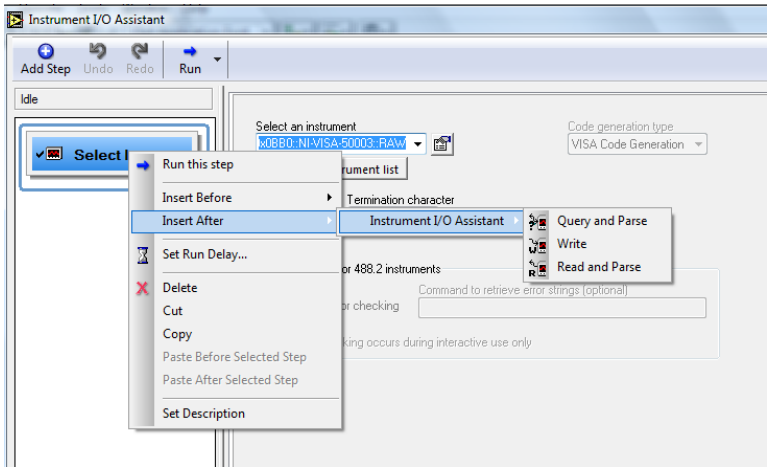


Figura 3.17. Menú para añadir funciones dentro del bloque I/O Assistant

Y se despliegan 3 opciones de pasos de comunicación para seleccionar.

Query and Parse: realiza una pregunta al equipo y realiza un análisis de la respuesta. Esta función combina, escritura, lectura y análisis.

Write: envía un comado al dispositivo.

Read and Parse: Lee y analiza los datos recibidos por el instrumento.

3.3.2.1. QUERY AND PARSE:

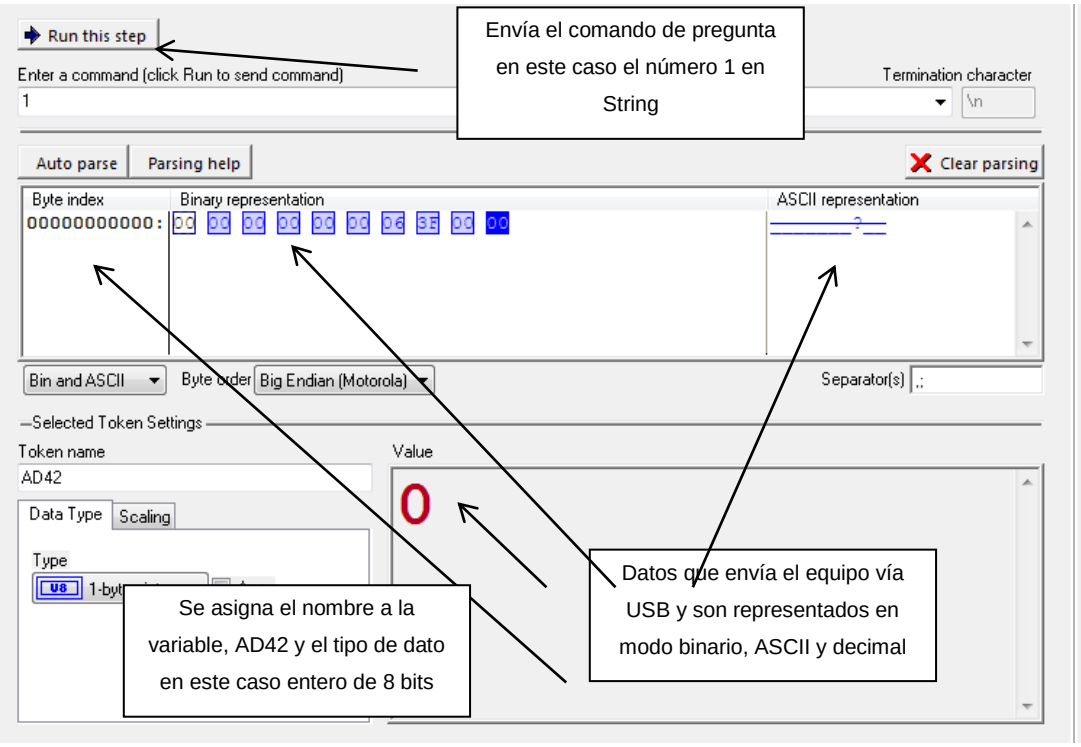


Figura 3.18. Configuración de la función Pregunta y análisis del I/O Assistant

La función Query and Parse se encarga de enviar una pregunta al dispositivo USB, esta pregunta será un byte o una palabra configurada previamente en el Módulo de adquisición de datos, por medio de la cual realizará las funciones correspondientes a la escritura de los datos necesarios para posteriormente realizar el análisis en mismo bloque de función.

En nuestro caso la pregunta la realiza enviando el número 1 en forma de carácter ASCII, con lo cual se le indica al módulo de adquisición de datos que se le está preguntando sobre los datos correspondientes a los input registers de Modbus.

Es decir una vez que el módulo recibe el carácter ACII “1”, aplica la función 4 del protocolo Modbus con lo cual recibe los datos correspondientes a los Input Registers del Esclavo y los traslada a vís USB para que sean analizados en este bloque.

3.3.2.2. WRITE:

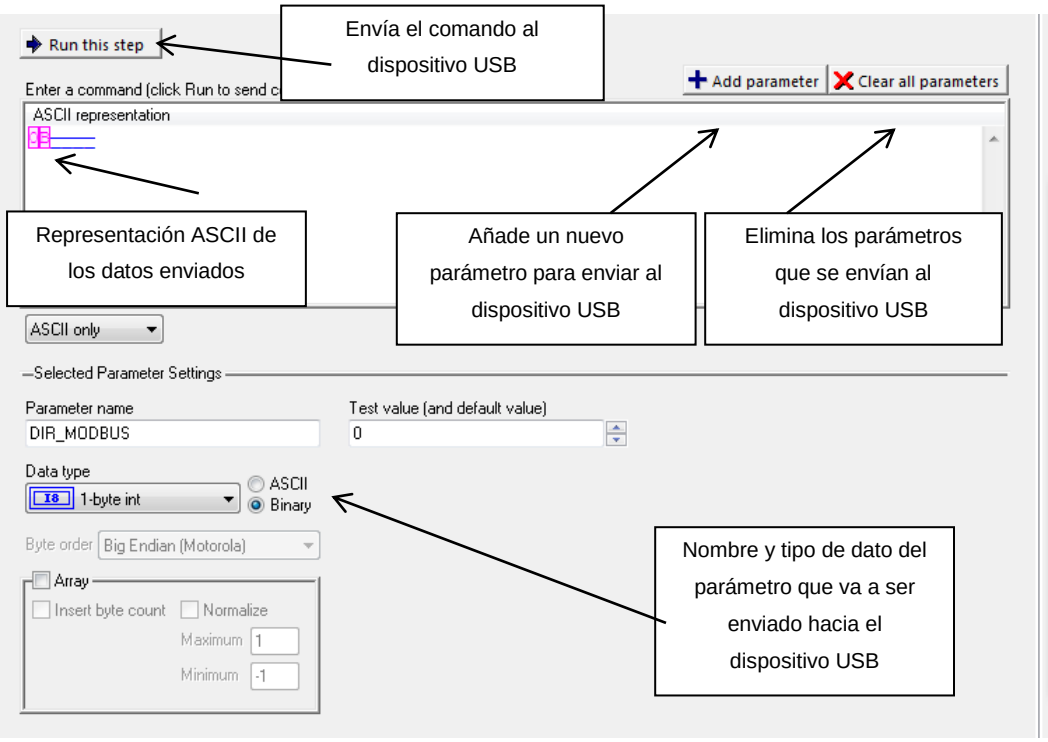


Figura 3.19. Configuración de la función Escritura del I/O Assistant

La función Write va acompañada del carácter ASCII “0”, lo cual indica al módulo de adquisición de datos que recibirá una trama de datos.

Estos datos tienes su función específica, ya que entre ellos está la dirección del Modbus esclavo a ser escrito y la dirección del registro del dispositivo Modbus esclavo donde se va aguardar el dato y que a continuación será aplicada ya sea en las salidas análogas o discretas según sea el caso.

Para la aplicación actual los comandos a utilizar son Query and Parse y Write únicamente ya que la función Read and Parse, se encuentra dentro de Query and Parse por lo que no hace necesario leer nuevamente los datos que provienen desde el equipo USB, por otra parte los datos serán enviados desde el módulo de adquisición de datos siempre y cuando exista primero una pregunta y esto se lo puede hacer con Query and Parse.

Una vez construido el bloque vamos a tener algo como lo que se muestra a continuación.

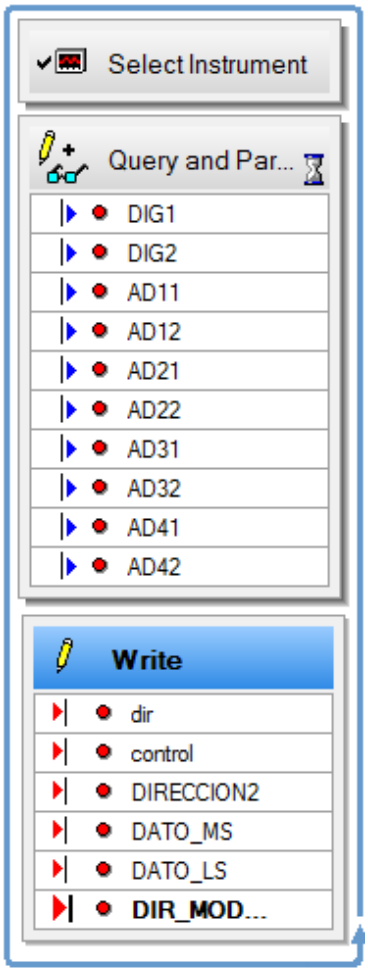


Figura 3.20. Bloque Instrument I/O Assistant luego de ser configurado.

Como se puede ver en la figura, existen 10 bytes de entrada, los cuales en pares de 2 en 2 forman 5 palabras de 16 bits, las cuales corresponden a una palabra de entradas discretas y 4 palabras de 12 bits correspondientes a la lectura de los canales análogos.

Además cuenta con 6 salidas de 1 byte cada 1 las cuales se detallan a continuación.

Dir: puede ser “1” ó “0” en formato ASCII este byte indica si va a leer o a escribir el módulo de adquisición de datos, en este caso va a ser “0” ya que va a escribir datos al Módulo.

Control: Este byte puede ser “A” ó “B” en modo ASCII, si es A indica que habilita la función escritura, si es B indica que deshabilita la función escritura, este byte actúa como un bloqueo antes de realizar cualquier escritura al módulo.

DIRECCION2: Indica la dirección dentro de los Holding Registers del Equipo esclavo Modbus en donde se va a escribir un dato.
DATO_MS y DATO_LS: el dato que va a ser enviado dividido en dos bytes, el más significativo y el menos significativo.
DIR_MOD: Es la dirección Modbus del dispositivo esclavo que va a recibir los datos.

3.3.2.3. Retardos entre funciones

Hay que tomar en cuenta que estas dos funciones se estarán repitiendo de manera continua por lo que se hace necesario establecer un tiempo de retardo entre funciones de esta manera:
Click derecho sobre la función y se despliega el siguiente menú.

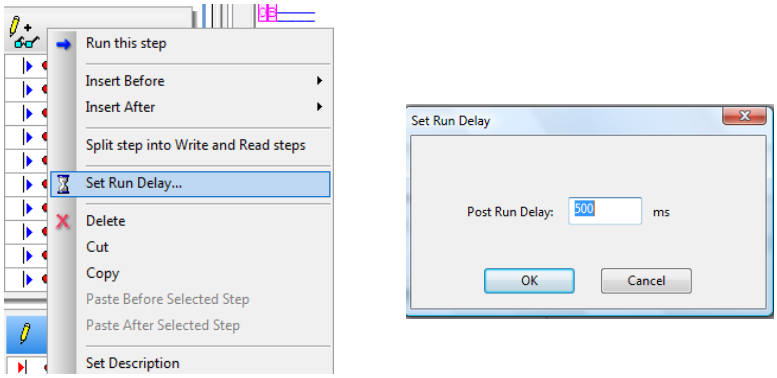


Figura 3.21. Añadiendo un retardo luego de cada función dentro del bloque

Donde se puede establecer el tiempo de retardo en milisegundos. Una vez realizadas las configuraciones necesarias, damos clic en OK y en el diagrama de bloque se generará el siguiente bloque Instrument I/O Assistant con todas las variables de entrada y salida que fueron creadas.

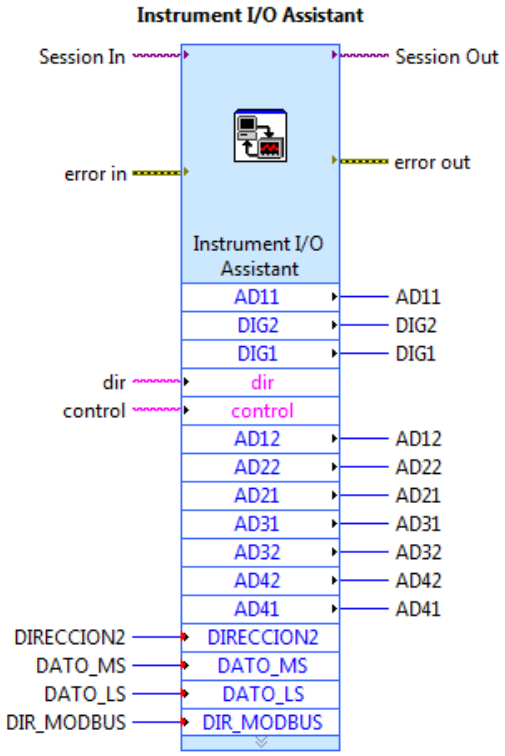


Figura 3.22. Bloque Instrument I/O Assistant creado y ubicado en el diagrama de bloque de Labview

El bloque tiene 10 datos de entradas los cuales corresponden datos de entradas digitales y análogas divididos en sus datos más significativo y menos significativo, ya que es una palabra de 16 bits las entradas digitales y 4 palabras de 12 bits las entradas análogas.

También se han generado 6 datos de salida, los cuales son control y dirección de módulo, los cuales determinan si va a estar en modo lectura o escritura, además se escribe la dirección Modbus del equipo esclavo, la dirección del registro dentro de este equipo y el dato a escribir dividido su estructura en byte más significativo y menos significativo.

De esta forma se obtiene el bloque de función Intrument I/O Assistant el cual comandará y monitoreará el módulo de adquisición de datos diseñado.

Cabe señalar que el microcontrolador que posee el driver de VISA-USB hace de Gateway entre un dispositivo VISA-USB y un dispositivo Modbus.

3.4. Firmware para las tarjetas que conforman el módulo de adquisición de datos.

A continuación se va a detallar cada programa de cada microcontrolador utilizado en el presente proyecto, cabe señalar que los programas más importantes del módulo de adquisición de datos tales como los que gestionan las comunicaciones Modbus esclavos y Maestro y USB, fueron realizados en C en el compilador CCS y en Assembler con el programa MPLAB de Microchip se realizaron los programas para los microcontroladores que actúan como esclavos I2C.

3.4.1. Programa para el Módulo Master de comunicaciones (Slave Modbus, Master I2C, Master SPI).

El programa fue realizado en lenguaje C con la ayuda del compilador CCS, el microcontrolador utilizado fue el PIC16F887, y las funciones que cumple este programa se las detalla a continuación ordenadas de acuerdo a su importancia.

- Trabajar como un dispositivo Esclavo Modbus el cual puede obtener su dirección con la ayuda de 1 dip switch conectado en uno de sus puertos, y capaz de conectarse vía RS-232 para una comunicación Maestro – Esclavo, vía USB con el Microcontrolador PIC18F2550 que actúa como Gateway Modbus-USB-VISA, en una red Ethernet mediante el Gateway USART-TCP/IP, y en una red RS-485 en modo half dúplex.
- Trabajar como un dispositivo Master I2C y Master SPI, para de esta forma establecer una comunicación con los PIC16F876A que actúan como esclavos I2C y con los integrados MCP4822 y MCP3202 que son esclavos SPI.

3.4.2. Desarrollo del firmware para comunicación Modbus Modo Esclavo.

Dentro del desarrollo de la comunicación Modbus modo esclavo, para poder realizar un monitoreo de la señales de entrada tanto análogas como digitales y poder realizar un control de sobre las señales de salida discretas y análogas, hemos utilizado las siguientes funciones Modbus.

- Función 3: Read Holding Registers (Registros de Lectura y escritura), donde vamos a direccionar las señales de salidas discretas y salidas análogas para poder monitorear su valor actual.
- Función 4: Read Input Registers (Registros solo de lectura), Donde vamos a direccionar las señales que provienen de entradas discretas y análogas.
- Función 6: Write Single Register (Escribir un solo registro), con la cual vamos a escribir uno de los registros llamados Holding Registers, señalando antes la dirección del registro que va a ser escrito.

3.4.2.1. Mapa de Registros Modbus Módulo Master de comunicaciones.

Registros de Lectura y Escritura (Holding Registers)

Los registros de escritura y lectura del protocolo Modbus, se caracterizan por estar ubicados a partir de la diercción 40001, y se los conoce como Holding Registers, son de 16 bits y son leídos desde un equipo Maestro Modbus con la función 3, con lo que se genera una trama como la que se indica a continuación.

- Petición del maestro

#Esclavo	Función	Dirección Primera Palabra	Número de Palabras a leer	CRC 16 bits
0Ah	03h	00h	0Ah	H L

Tabla 3.1. Trama Modbus función 3 petición del Maestro

Con lo cual el maestro le está solicitando al esclavo de dirección 10 que le envíe los datos correspondientes a 10 registros a partir de la dirección 00h que para el caso del mapa de direcciones correspondería a la dirección 40001.

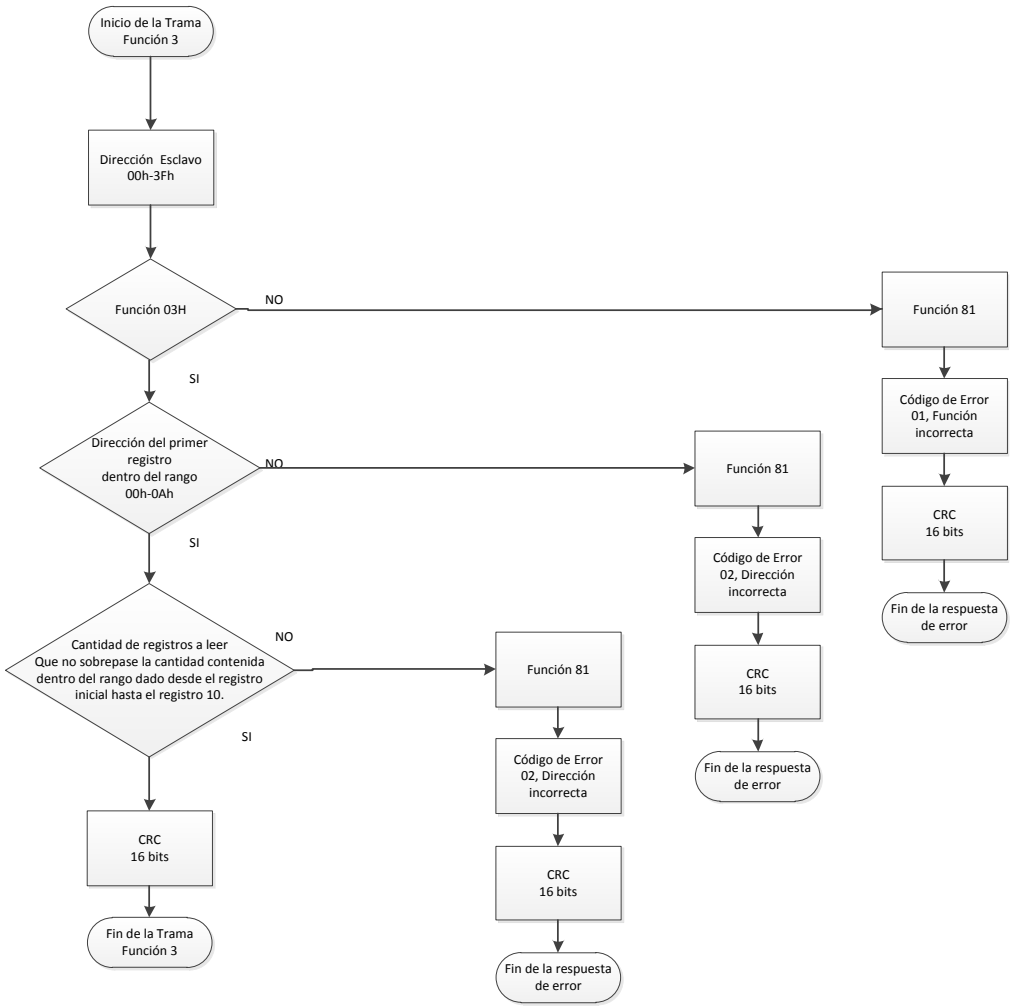


Figura 3.23. Diagrama de Bloques para la función 3 Modo general

Hay que tomar en cuenta que para poder direccionar bien y no tener errores de dirección ilegal el momento de leer los registros es necesario tomar en cuenta lo siguiente.

Los registros utilizados y válidos dentro del programa van desde el 00h hasta el 09H, es decir si la dirección del primer registro es la 00H, máximo podrán leerse 10 registros, ahora si la dirección del primer registro la establecemos como la 05H, como máximo podrán leerse 5 registros y no se tomarán en cuenta los registros desde el 00h hasta 04h, si no se cumple estas opciones muy lógicas, el esclavo generará una trama de Error la cual se detalla a continuación.

#Esclavo	Función	Código de error dirección incorrecta	CRC 16 bits

0Ah	81h	02h	H L

Tabla 3.2. Trama Modbus código de error.

Para leer 10 registros de lectura y escritura a partir de la dirección 00h el diagrama de bloques sería así:

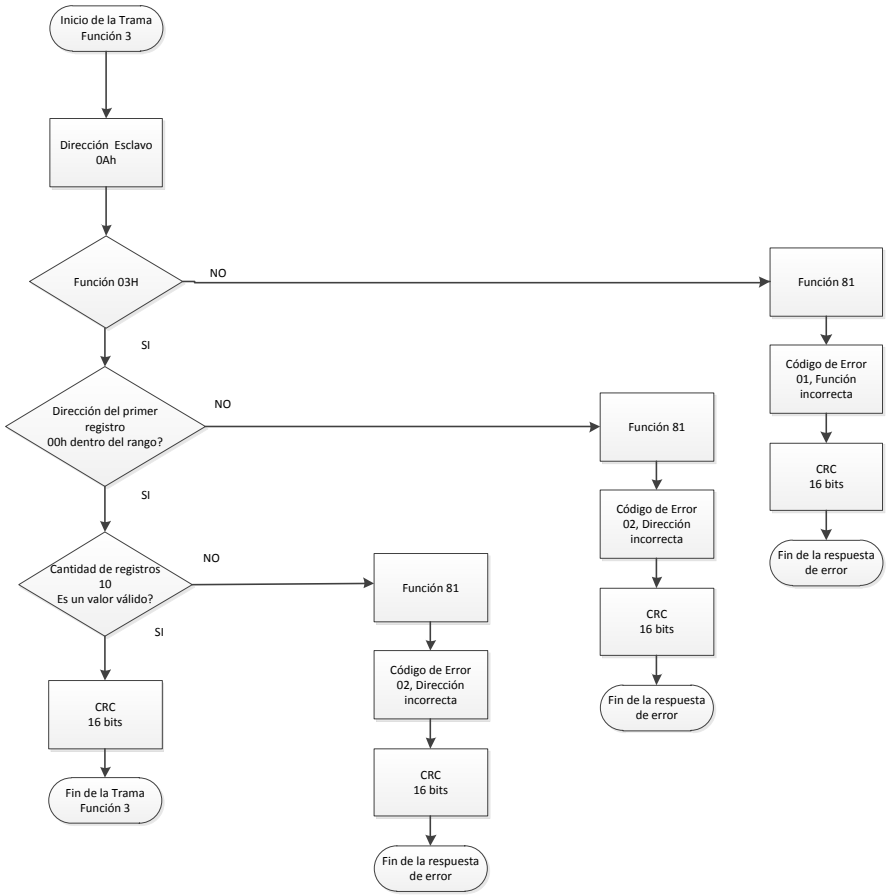


Figura 3.24. Diagrama de bloques solicitud de 10 registros de lectura y escritura a partir de la dirección 00h al esclavo 0Ah

- Respuesta del esclavo

#Esclavo	Función	Número de registros leídos	Primera Palabra	Siguiente palabra hasta la décima	CRC 16 bits
0Ah	03h	0Ah	H L	H L	H L

Tabla 3.3. Trama Modbus función 3 Respuesta esclavo

El esclavo de dirección 10 va a responder los 10 registros a partir de la dirección 00h equivalente a 40001, que el Maestro solicitó.

Figura 3.25. Diagrama de flujo respuesta del esclavo dirección 10. 10 registros a partir de la dirección 00h

Registros de Lectura (Input Registers)

Los registros de lectura del protocolo Modbus, se caracterizan por estar ubicados a partir de la diercción 30001, y se los conoce como Input Registers, son de 16 bits y son leídos desde un equipo Maestro Modbus con la función 4, con lo que se genera una trama como la que se indica a continuación.

- Petición del maestro

#Esclavo	Función	Dirección Primera Palabra	Número de Palabras a leer	CRC 16 bits
0Ah	04h	00h	05h	H L

Tabla 3.4. Trama Modbus función 4 petición del Maestro

Con lo cual el maestro le está solicitando al esclavo de dirección 10 que le envíe los datos correspondientes a 5 registros a partir de la dirección 00h que para el caso del mapa de direcciones correspondería a la dirección 30001.

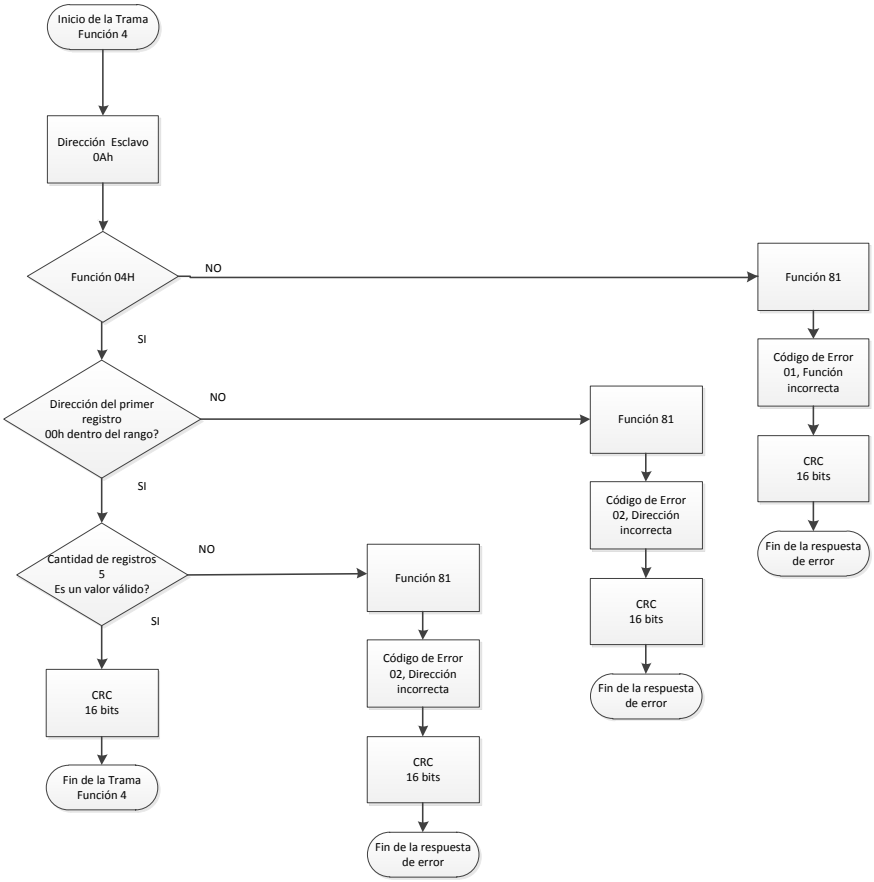


Figura 3.26. Diagrama de bloques solicitud de 10 registros de lectura a partir de la dirección 00h al esclavo 0Ah

- Respuesta del esclavo

#Esclavo	Función	Número de registros leídos	Primera Palabra	Siguiente palabra hasta la décima	CRC 16 bits
0Ah	04h	05h	H L	H L	H L

Tabla 3.5. Trama Modbus función 4 Respuesta esclavo

El esclavo de dirección 10 va a responder los 5 registros a partir de la dirección 00h equivalente a 30001, que el Maestro solicitó.

Figura 3.27. Diagrama de flujo respuesta del esclavo dirección 10. 10 registros a partir de la dirección 00h

A continuación se muestra como están dispuestas las variables dentro del mapa de registros que estamos utilizando en el módulo de adquisición de datos para el microcontrolador que actúa como Esclavo Modbus.

REGISTROS DE ESCRITURA Y LECTURA		
DIRECCIÓN	**VARIABLE**	**VALOR QUE PUEDE TENER**
40001	SALIDA ANÁLOGA 0-10V 0-20mA MAS-VI	0-4095
40002	SALIDA ANÁLOGA 0-10V 0-20mA MAS-VI	0-4095
40003	SALIDA ANÁLOGA 0-10V 0-20mA MAS-VI	0-4095
40004	SALIDA ANÁLOGA 0-10V 0-20mA MAS-VI	0-4095
40005	LIBRE	
40006	LIBRE	
40007	LIBRE	
40008	LIBRE	
40009	LIBRE	
400010	SALIDAS DIGITALES MSD-RELES	0-255
REGISTROS DE LECTURA		
DIRECCIÓN	**VARIABLE**	
30001	ENTRADAS DIGITALES MED1	0-65535
30002	ENTRADAS ANÁLOGAS 0-10V 4-20mA MAE1	0-4095
30003	ENTRADAS ANÁLOGAS 0-10V 4-20mA MAE2	0-4095
30004	ENTRADAS ANÁLOGAS 0-10V 4-20mA MAE3	0-4095
30005	ENTRADAS ANÁLOGAS 0-10V 4-20mA MAE4	0-4095

Tabla 3.6. Mapa de registros Modbus Módulo de Adquisición de datos

Escritura de registros con la función 06H.

Los registros de escritura del protocolo Modbus, se caracterizan por estar ubicados a partir de la dirección 40001, y se los conoce como Holding Registers, son de 16 bits y pueden ser escritos desde un equipo Maestro Modbus con la función 6 o la función 16, en nuestro caso vamos a usar la función 6 para escribir un solo registro a la vez.

Con lo que se genera una trama como la que se indica a continuación.

- Petición del maestro

#Esclavo	Función	Dirección a escribir	Dato que va a ser escrito	CRC 16 bits
0Ah	06h	00h	0FFFh	H L

Tabla 3.7. Trama Modbus función 6 petición del Maestro

Con lo cual el maestro le está escribiendo al esclavo de dirección 10 en el registro 00h que corresponde al 40001 el dato 0FFFh.

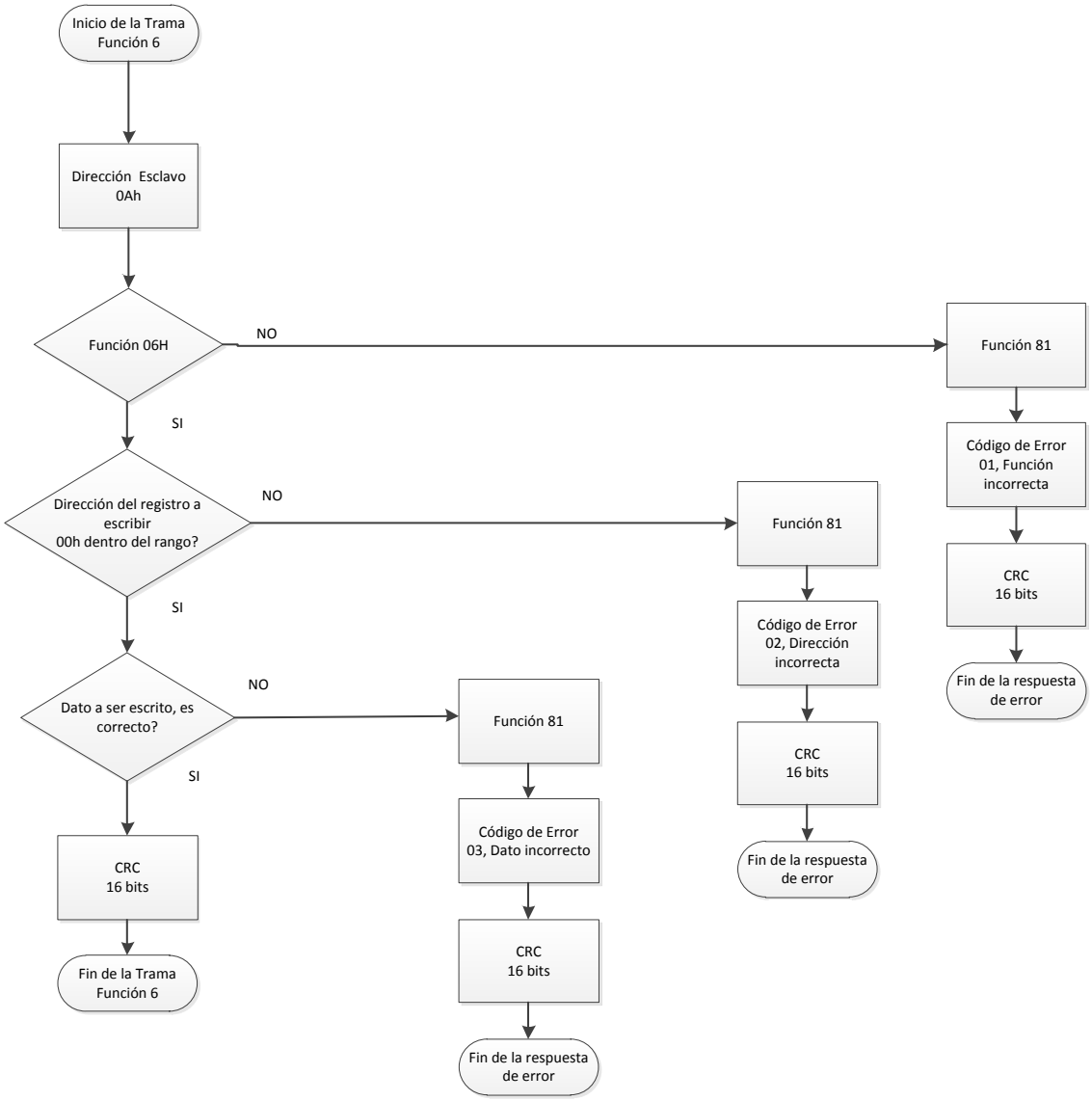


Figura 3.28. Diagrama de bloques escritura del registro 00H con la función 6.

- Respuesta del esclavo

#Esclavo	Función	Dirección que fue escrita	Dato que fue escrito	CRC 16 bits
0Ah	06h	00h	0FFFh	H L

Tabla 3.8. Trama Modbus función 6 respuesta del Esclavo

Con lo cual el esclavo está confirmando que en el registro 00h se escribió el dato 0FFFh.

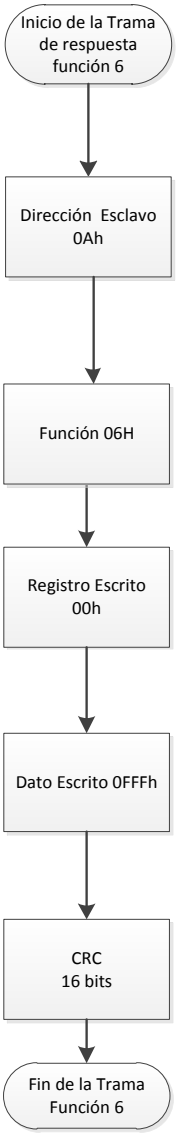


Figura 3.29. Diagrama de bloques respuesta del esclavo dirección 0Ah luego de haber sido escrito su registro 00h con el dato 0FFFh.

Cabe señalar que para las salidas y entradas análogas los valores que van a tener sus registros son de 0 a 4095 ya que tienen resolución de 12 bits.

Las entradas discretas forman una palabra de 16 bits.

Y las salidas discretas son 8 por lo que forman un byte de 8 bits.

3.4.2.2. Configuración Modbus Esclavo con el compilador CCS para el PIC16F887

A continuación se van a presentar las líneas básicas del programa para la configuración Modbus Slave.

Para poder configurar la comunicación Modbus se utilizó la librería Modbus.c que viene con el compilador CCS y posee las subrutinas que ejecutan las diferentes funciones Modbus, con las cuales vamos a trabajar en el programa principal para poder establecer la comunicación para cumplir las funciones que requerimos.

```
//Incluimos las librerías del PIC16F887 el cual vamos a configurar
#include     <16f887.h>
//Configuramos los fusibles de Microcontrolador, Cristal externo, se
//deshabilita el WDT, sin código de protección, reset cuando exista bajo
//voltaje.
#fuses       XT, NOWDT, NOLVP, NOBROWNOUT, NOPROTECT, PUT
//Establecemos que vamos a trabajar con un cristal de 4MHz
#use         delay(clock=4000000)
//Seleccionamos para que el PIC16F887 trabaje en modo Modbus Esclavo.
#define      MODBUS_TYPE MODBUS_TYPE_SLAVE
//Definimos como 64 bytes el tamaño del buffer de recepción.
#define      MODBUS_SERIAL_RX_BUFFER_SIZE    64
//Establecemos como 9600 baudios la velocidad de transmisión de datos
#define      MODBUS_SERIAL_BAUD              9600
//Habilita la interrupción por recepción serial
#define      MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
//Incluye la librería MODBUS.C ubicada en el directorio del Compilador CCS
#include     "MODBUS.c"
//Establecemos que la dirección Modbus estará dada por el puerto A.
#define      MODBUS_ADDRESS PORT_A
```

Una vez definidas las configuraciones básicas de Modbus para el microcontrolador, se establecen las direcciones de los registros dentro de la memoria del microcontrolador y con éstas definidas se puede realizar las subrutinas respectivas correspondientes a cada variable ya sea de entrada o de salida.

```
//Creamos 2 arreglos de 10 registros de 16 bits cada registro y los //llamamos
hold_regs e input_regs, los cuales van a ser nuestros registros //de memoria de
lectura y escritura y solamente de lectura //respectivamente.
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};
//Y de la siguiente manera es como asignamos el valor de cada variable de
//salida y entrada a las direcciones correspondientes para que puedan ser
//leídas y escritas por un dispositivo Modbus Master.
```

```
//Input registers ó registros de dirección 3xxxx
input_regs[0]=make16(DIG_IN0,DIG_IN1);       //Entradas discretas 16 bits
input_regs[1]=make16(AN01,AN02);             //Entrada análoga 0 12 bits
input_regs[2]=make16(AN11,AN12);             //Entrada análoga 1 12 bits
input_regs[3]=make16(AN21,AN22);             //Entrada análoga 2 12 bits
input_regs[4]=make16(AN31,AN32);             //Entrada análoga 3 12 bits
input_regs[5]=make16(00,00);                 //Registro Libre
input_regs[6]=make16(00,00);                 //Registro Libre
input_regs[7]=make16(00,00);                 //Registro Libre
input_regs[8]=make16(00,00);                 //Registro Libre
input_regs[9]=make16(00,00);                 //Registro Libre
//Holding registers ó registros de dirección 4xxxx
hold_regs[0]=make16(an_out02,an_out01);      //Salida análoga 0 12 bits
hold_regs[1]=make16(an_out12,an_out11);      //Salida análoga 1 12 bits
hold_regs[2]=make16(an_out22,an_out21);      //Salida análoga 2 12 bits
hold_regs[3]=make16(an_out32,an_out31);      //Salida análoga 3 12 bits
hold_regs[4]=make16(00,00);                  //Registro Libre
hold_regs[5]=make16(00,00);                  //Registro Libre
hold_regs[6]=make16(00,00);                  //Registro Libre
hold_regs[7]=make16(00,00);                  //Registro Libre
hold_regs[8]=make16(00,00);                  //Registro Libre
hold_regs[9]=make16(out_d,00);               //Salidas discretas 8 bits
```

3.4.2.3. Herramientas para desarrollar el protocolo Modbus

Para poder desarrollar la comunicación Modbus en le Microcontrolador PIC se usaron varias herramientas informáticas que actúan como simuladores Modbus en modo Master (ModbusPoll y ModScan) y para modo esclavo (MOD-RSSIM)

Programa simulador Modbus Poll.

Modbus Poll es un programa muy popular que simula ser un Master Modbus para poder realizar pruebas de funcionamiento y desarrollo de dispositivos esclavos Modbus, soporta Modbus RTU/ASCC y Modbus sobre TCP/IP.

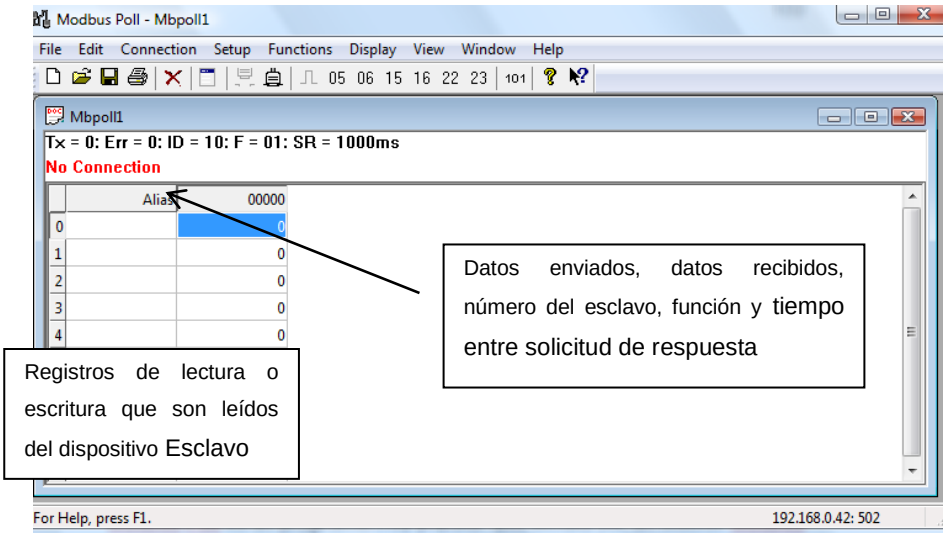


Figura 3.30. Entorno de trabajo Modbus-Poll

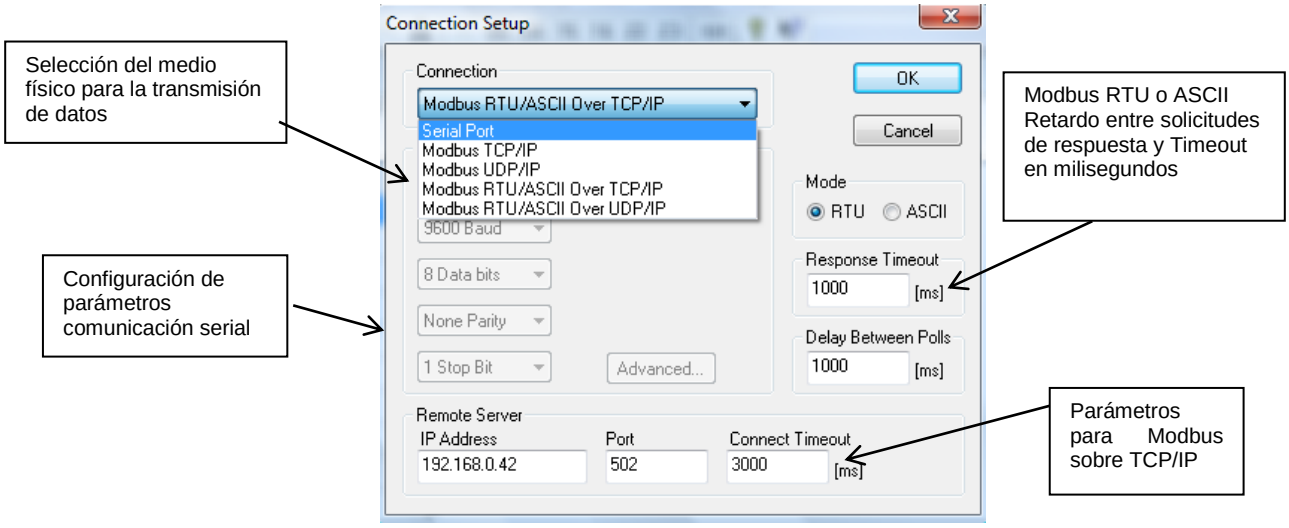


Figura 3.31. Menú para establecer los parámetros de comunicación

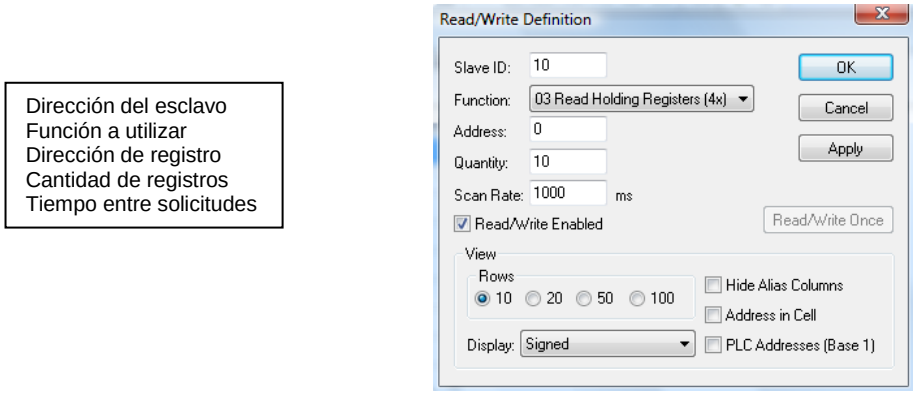


Figura 3.32. Menú para establecer dirección, función y cantidad de registros a leer.

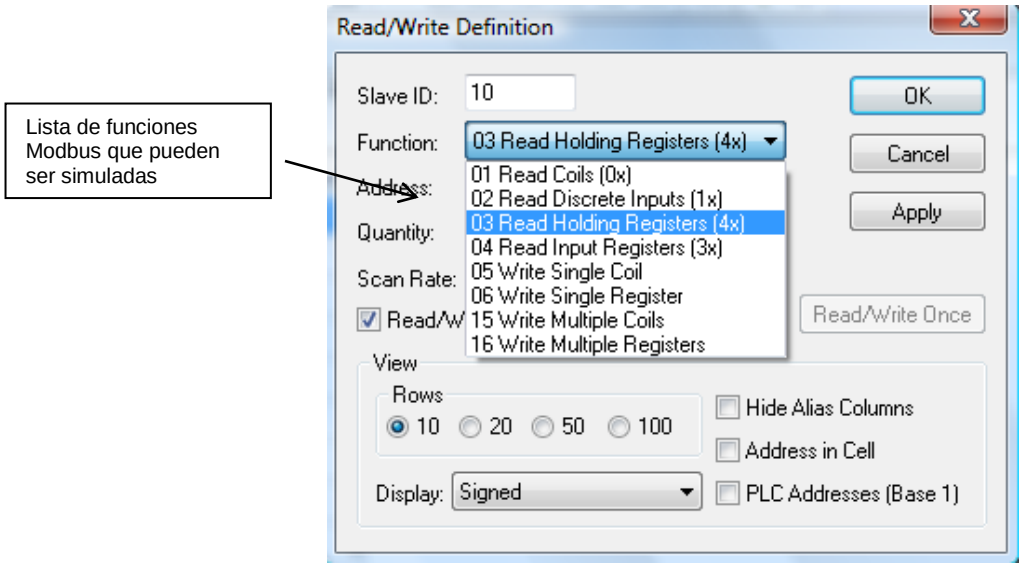


Figura 3.33. Menú para establecer la función Modbus que va a simular el programa

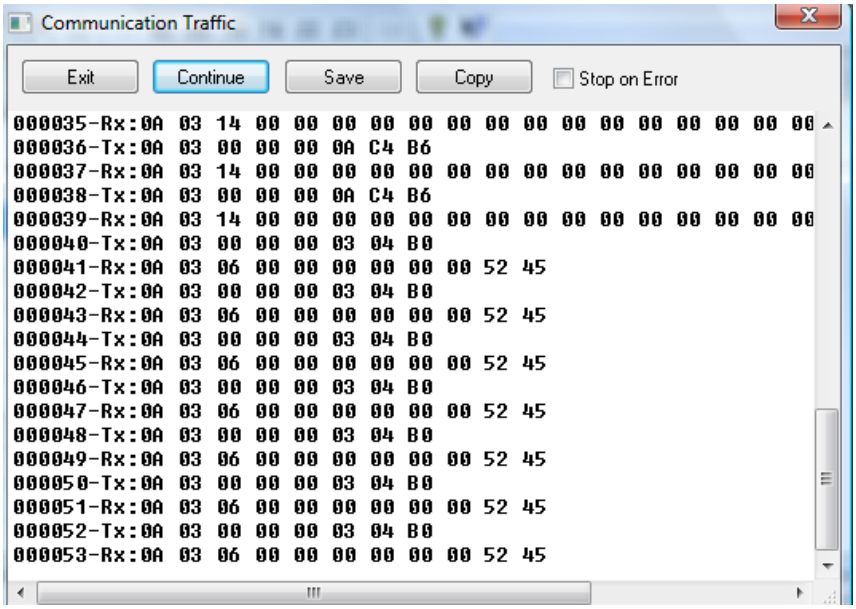


Figura 3.34. Ventana en la que se puede ver el tráfico de las tramas de pregunta y respuesta.

Programa simulador ModScan

ModScan al igual que Modbus Poll es un programa que simula un equipos Master Modbus y de esta manera probar equipos Modbus esclavos como PLC's o en nuestro caso, el módulo de adquisición de datos.

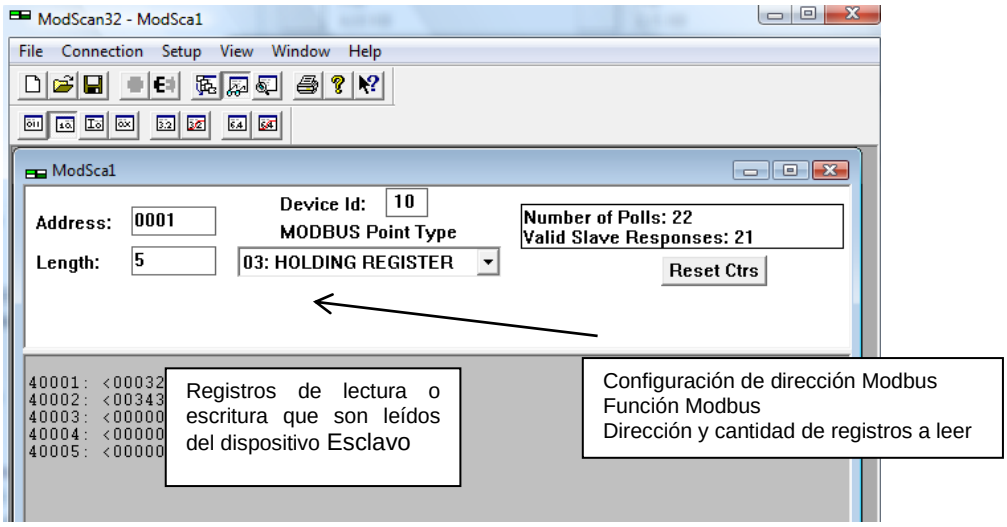


Figura 3.35. Entorno de trabajo ModScan

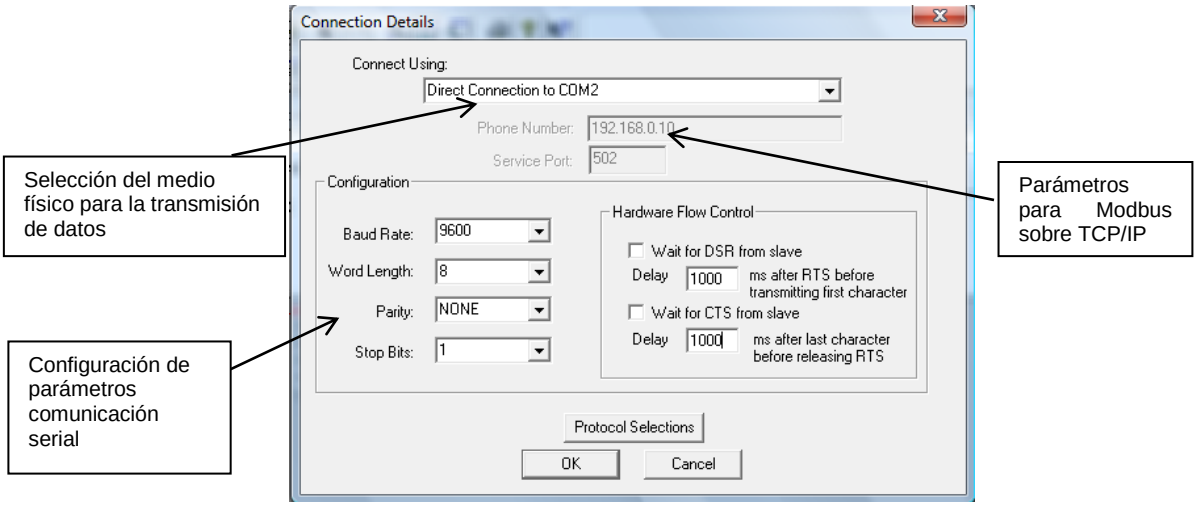


Figura 3.36. Menú para establecer los parámetros de comunicación

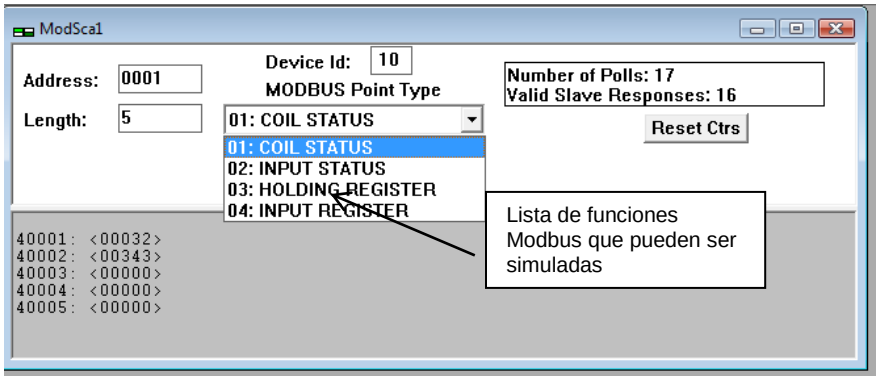


Figura 3.37. Menú para establecer la función Modbus que va a simular el programa

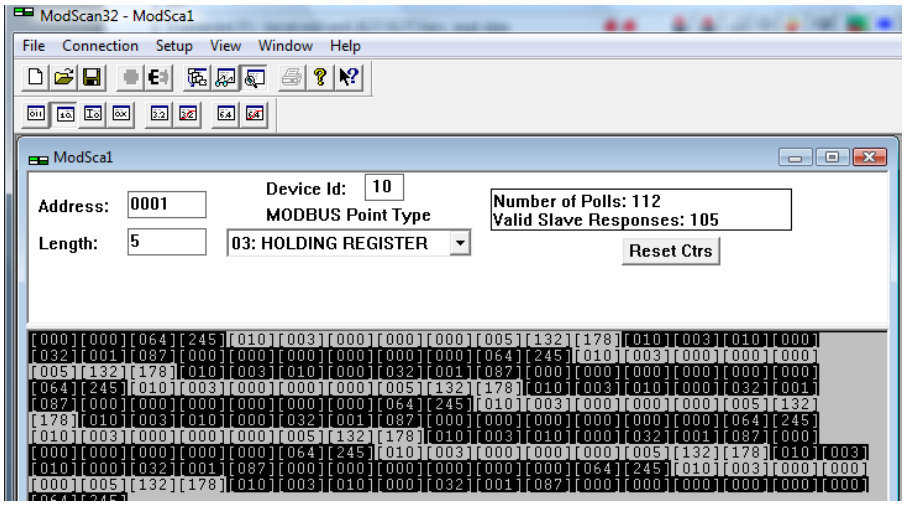


Figura 3.38. Ventana en la que se puede ver el tráfico de las tramas de pregunta y respuesta

Programa simulador MOD RSSIM

Este simulador es de mucha utilidad para probar los equipos Master Modbus ya que puede simular hasta 255 esclavos Modbus, puede simular en modo RS-232 y sobre TCP/IP, posee su ventana principal en la cual se puede ir variando los datos de los registros y mantiene activos todos los esclavos, por lo que a la dirección del esclavo dada entre 00h y FFh, éste simulador responderá sin ningún problema.

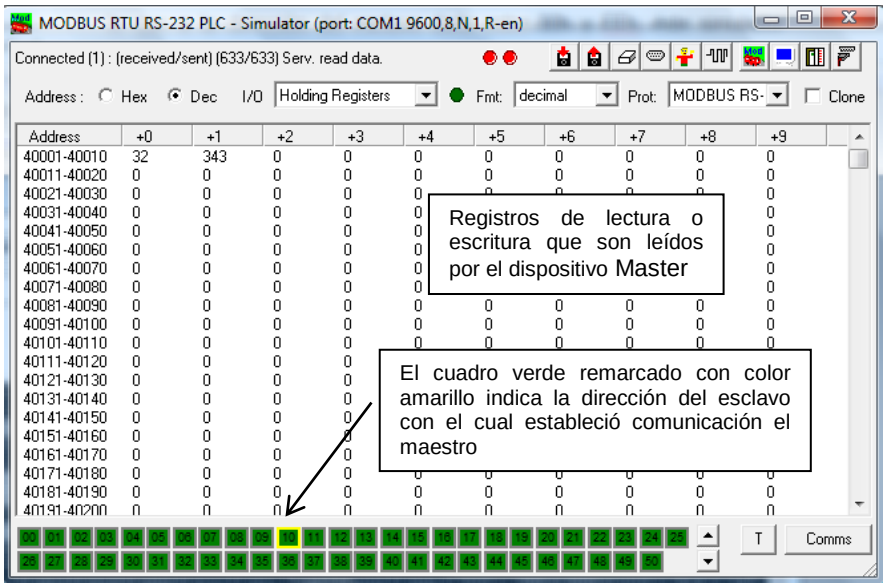


Figura 3.39. Entorno de trabajo Mod RSSIM

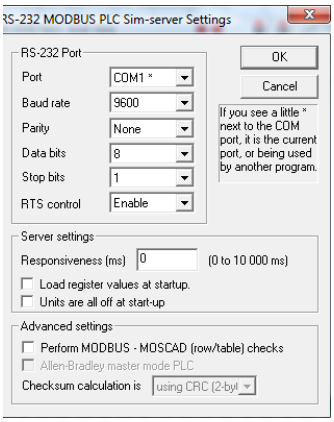


Figura 3.40. Menú para establecer los parámetros de comunicación seriales

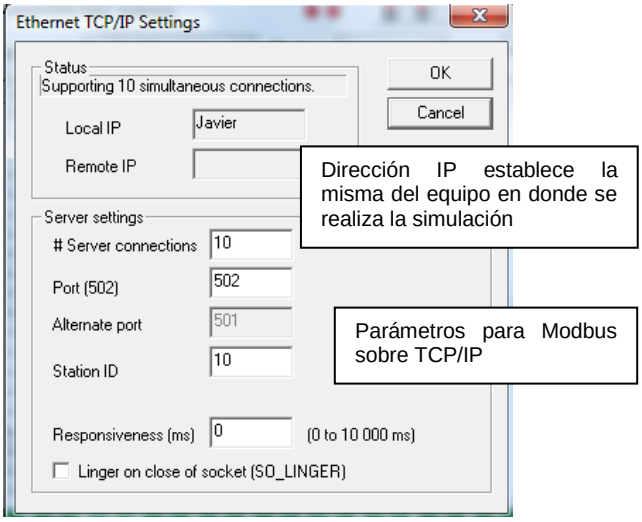


Figura 3.41. Menú para establecer los parámetros de comunicación sobre TCP/IP

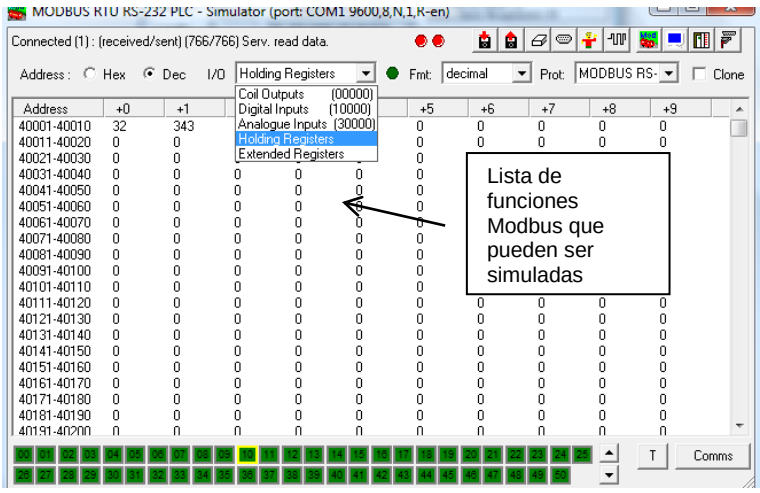


Figura 3.42. Menú para establecer la función Modbus que va a simular el programa

3.4.2.4. 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 o invertir tiempo en entender nuevas aplicaciones. 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. Los Servidores NI OPC se conectan a través del cliente OPC en el Módulo LabVIEW Datalogging and Supervisory Control (DSC) permitiendo desarrollar un sistema HMI/SCADA con PLCs, PACs y sensores inteligentes. Tiene por características principales las siguientes:

- 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.
- Funciona con LabVIEW y cliente OPC en LabVIEW DSC

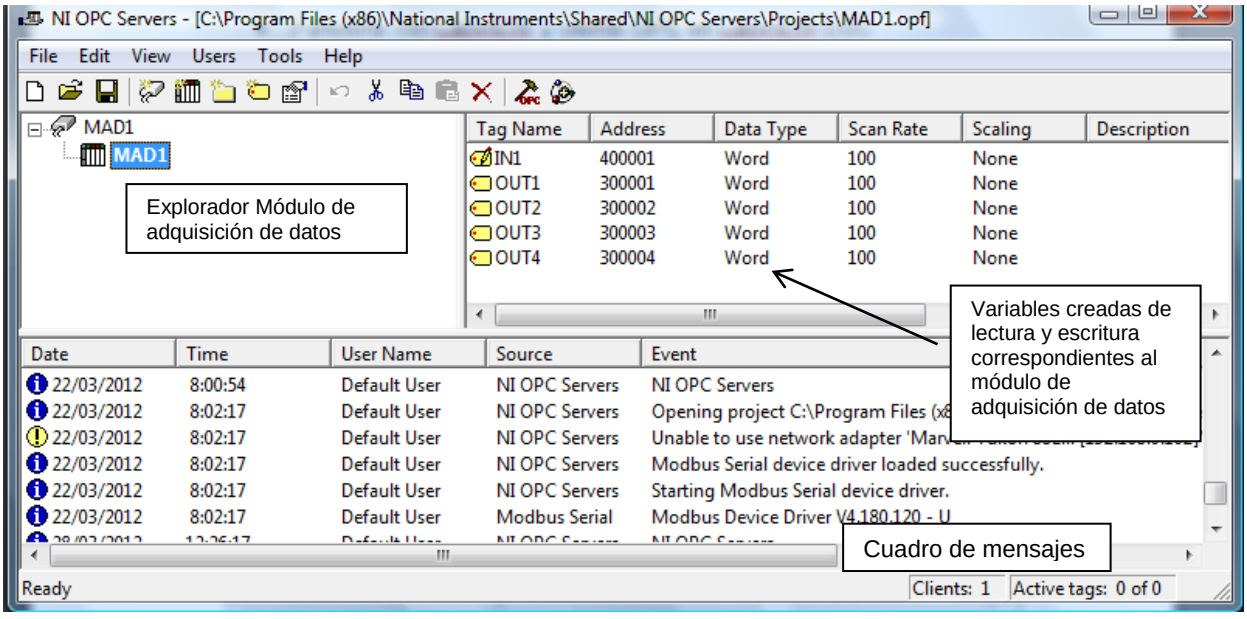


Figura 3.43. Entorno pantalla principal de NI-OPC server

Proceso para la configuración del Módulo 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 venta 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.

Nuestro canal se va a llamar MAD2012, por poner un ejemplo.

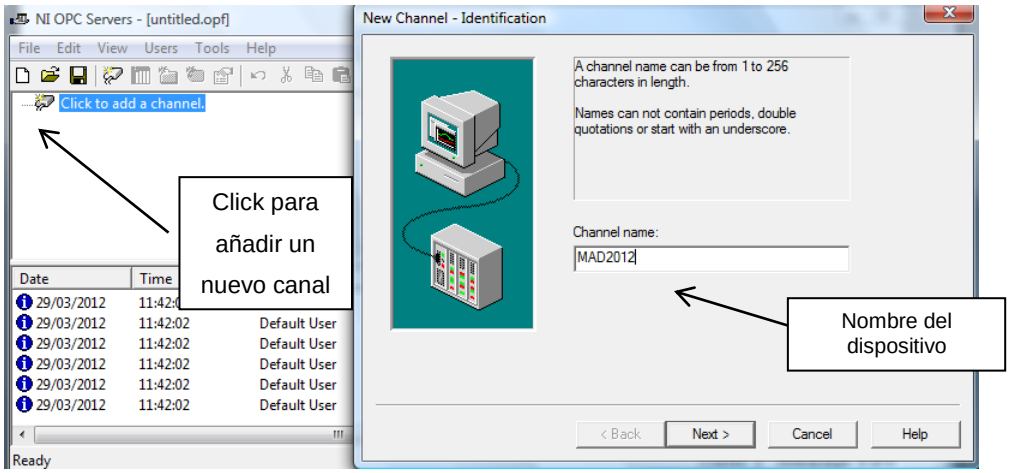


Figura 3.44. Crear y adjuntar un nuevo canal en el OPS Server

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 PLC's, tales como Allen-Bradley, Mitsibishi, Siemens, Cuttler Hammer, Omron, Toshiba.

Para nuestro caso vamos a utilizar el driver de Modbus RTU Serial o Ethernet, 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.

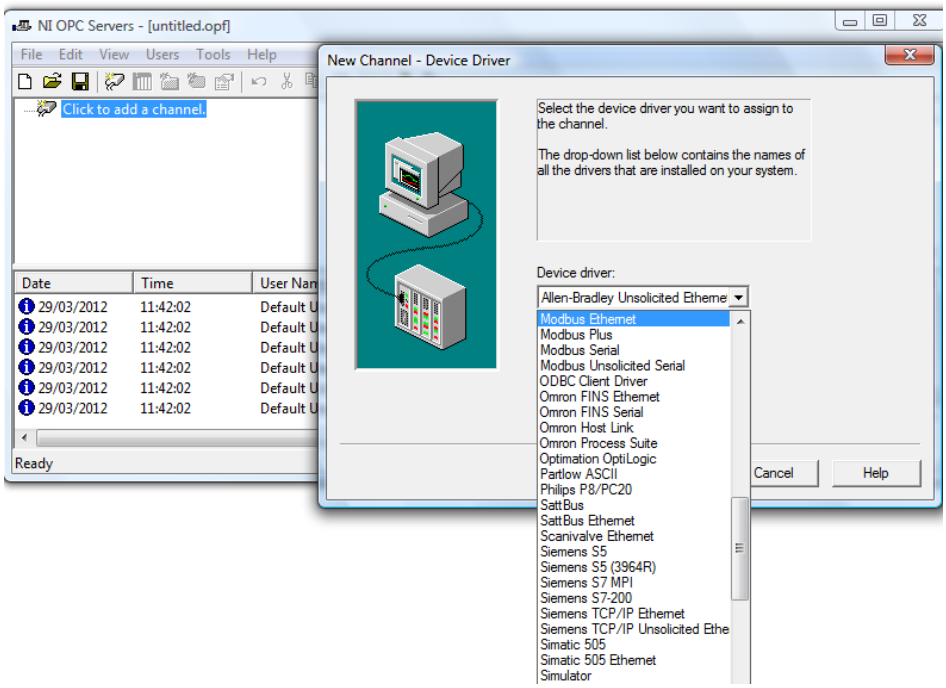


Figura 3.45. Seleccionamos El driver que vamos a asignar al canal

Si seleccionamos Driver Modbus Serial aparece la siguiente ventana.

Donde configuramos la velocidad de transmisión de datos, asignamos el puerto COM, establecemos Paridad y número de bits de datos y cantidad de bits de parada, así como también el control de flujo

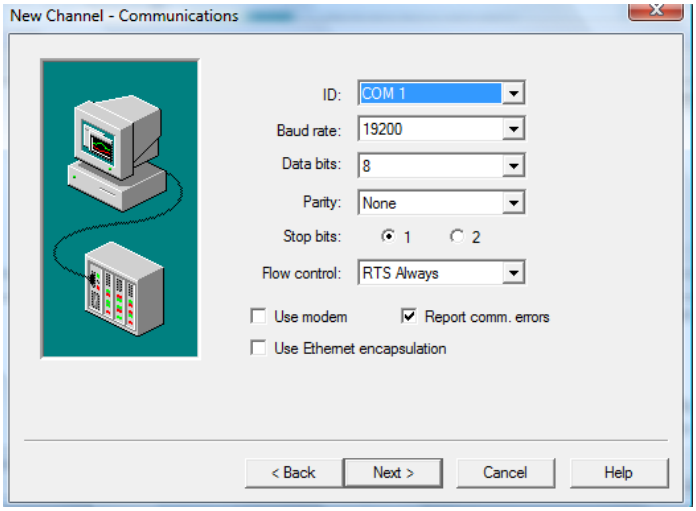


Figura 3.46. Ventana para configurar la comunicación serial

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

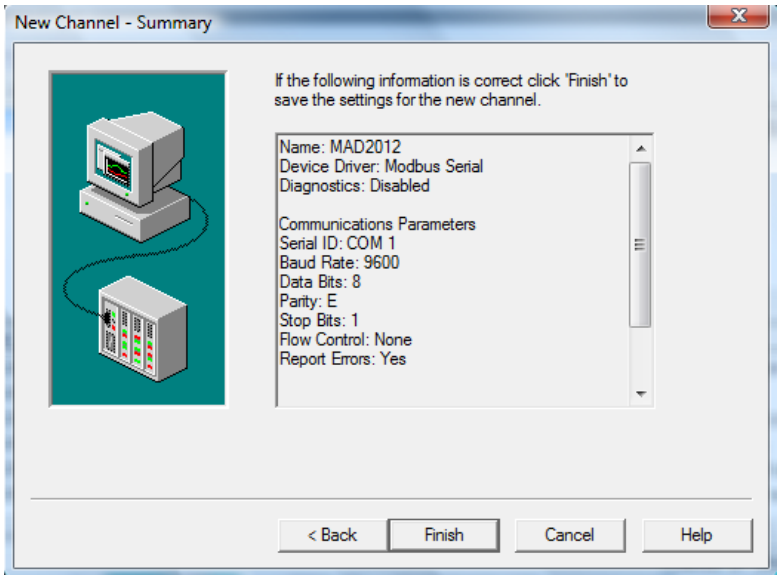


Figura 3.47. Ventana que muestra configuración actual

Ahora en el explorador de dispositivos se ha creado nuestro Módulo MAD2012, el cual muestra una opción que indica crear un nuevo dispositivo.

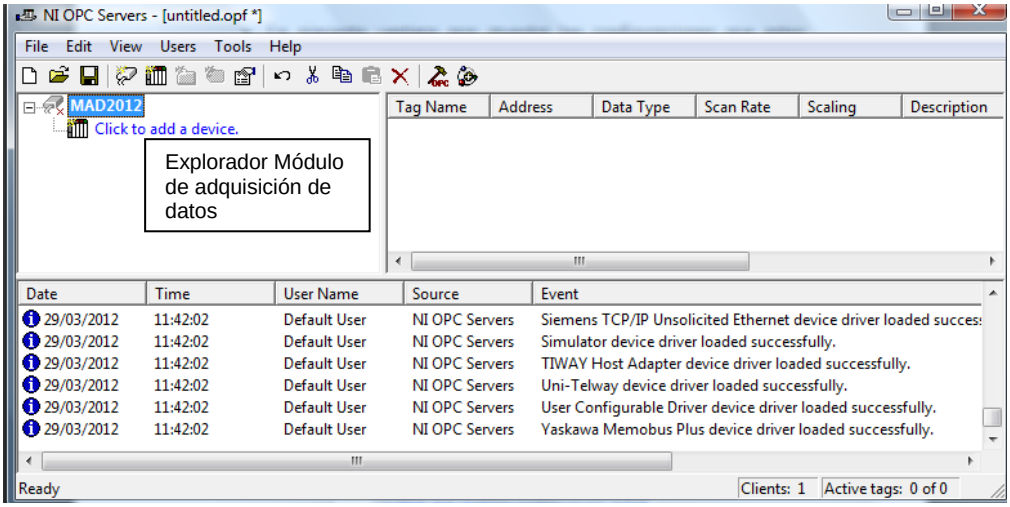


Figura 3.48. Entorno pantalla principal de NI-OPC server, agregar nuevo dispositivo

El dispositivo que vamos a adjuntar se llamará MODBUS_MAD, este será el módulo de adquisición de datos que trabajará como Modbus Esclavo, y luego presionamos Next.

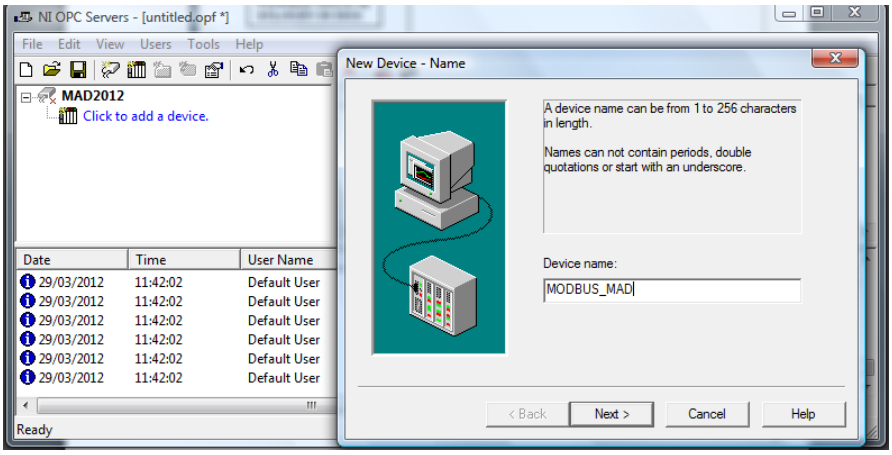


Figura 3.49. Crear y adjuntar un nuevo dispositivo para el canal previamente creado en el OPS Server

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

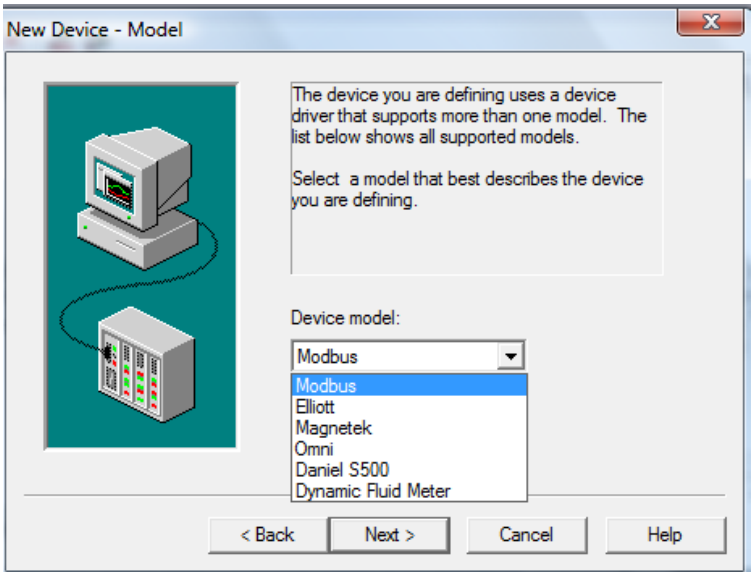


Figura 3.50. Seleccionamos el modelo que describe al dispositivo que estamos usando

Configuramos la dirección Modbus del dispositivo, tomando en cuenta que el módulo de adquisición tendrá su un dipswitch con el cual vamos a asignarle la dirección Modbus.

En este caso le asignamos la dirección 10.

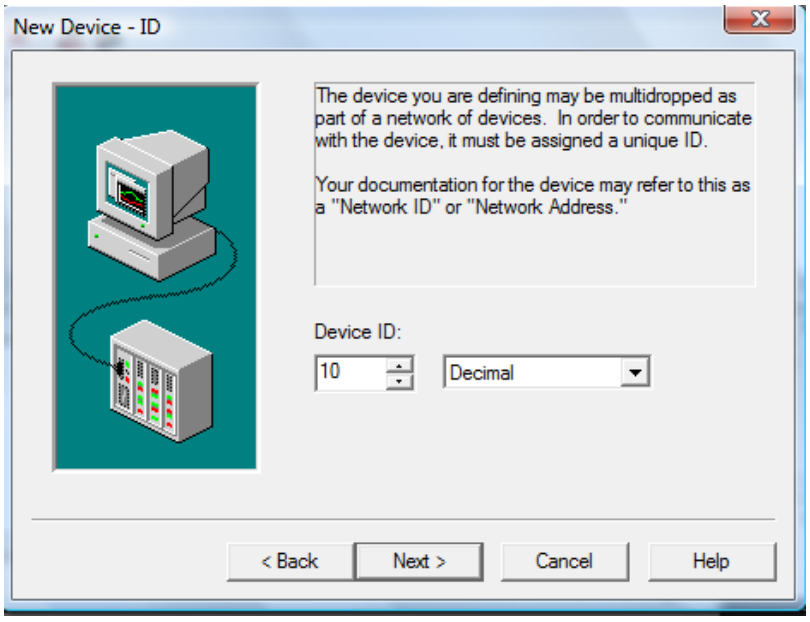


Figura 3.51. Ventana para asignar la dirección del dispositivo Modbus.

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

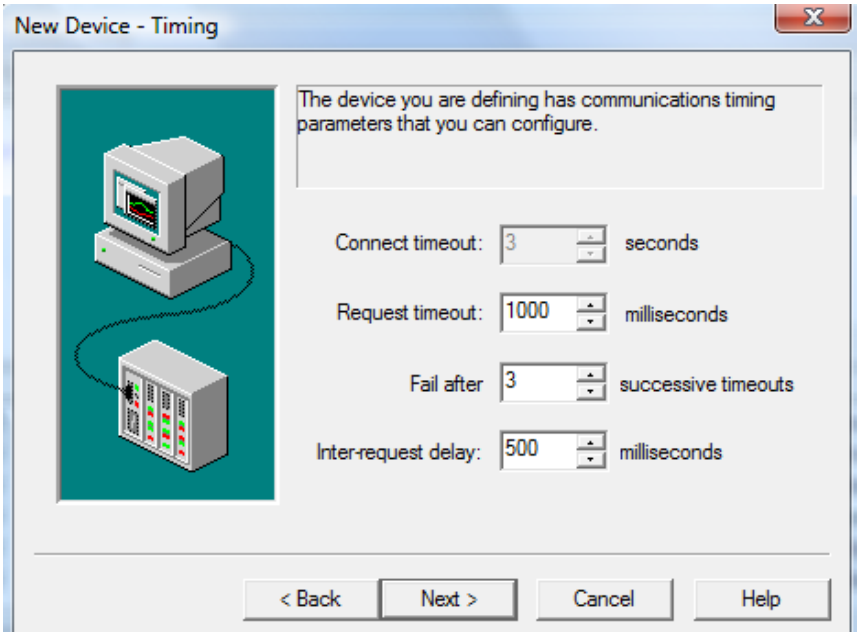


Figura 3.52. Ventana para establecer configuraciones de tiempos para la comunicación Modbus

Ahora definimos la función 6 de Modbus para escribir los registros del Módulo de adquisición de datos.

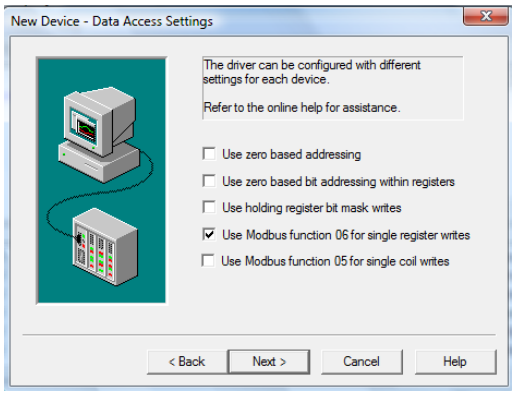


Figura 3.53. Ventana para establecer la funciones para escribir datos vía Modbus.

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

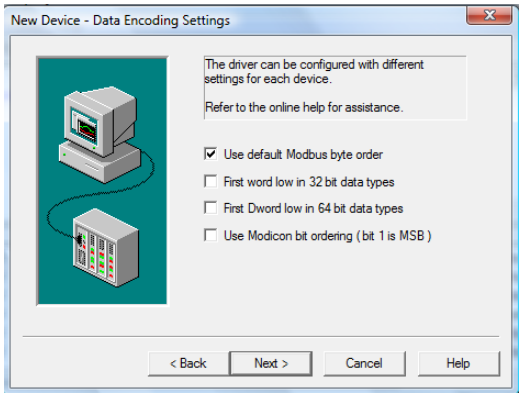


Figura 3.54. Ventana para establecer la codificación de los datos.

A continuación configuramos el tamaño de los registros que vamos a leer, para este caso son 10 de tipo registros internos de lectura y 10 registros de escritura y lectura.

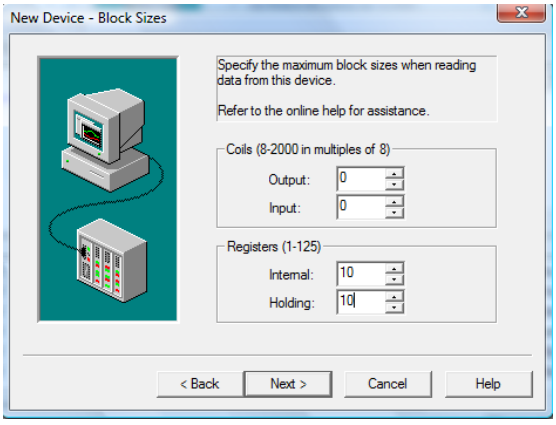


Figura 3.55. Ventana para establecer el tamaño de los bloques de datos.

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

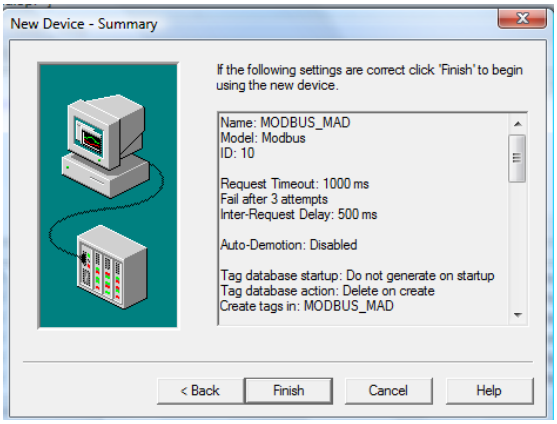


Figura 3.56. Ventana que muestra los datos con los cuales fue configurado el dispositivo Modbus.

En este punto se ha generado el dispositivo 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 nuestro Módulo de adquisición 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.

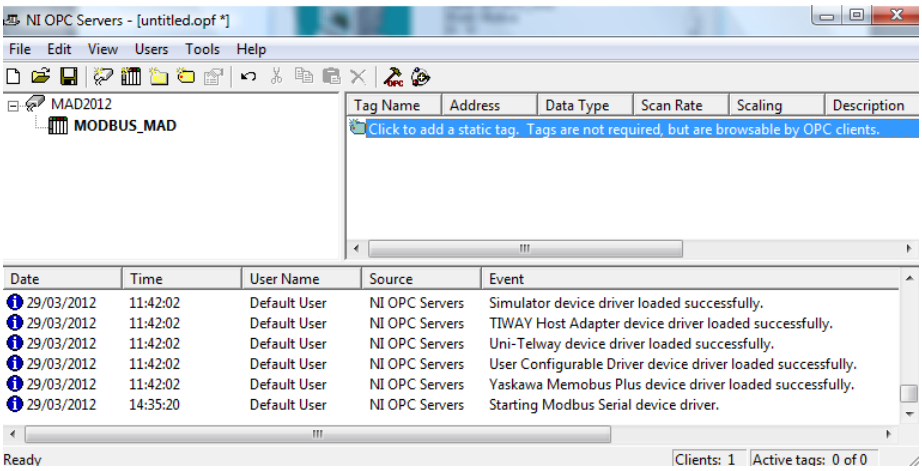


Figura 3.57. Ventana principal muestra el canal y el dispositivo creados y da la opción para crear tags.

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, para el caso del ejemplo le estamos dando el Nombre de DIG_IN, la cual representa el valor decimal del equivalente Binario que corresponde a las 16 entradas digitales de 24 voltios, 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.

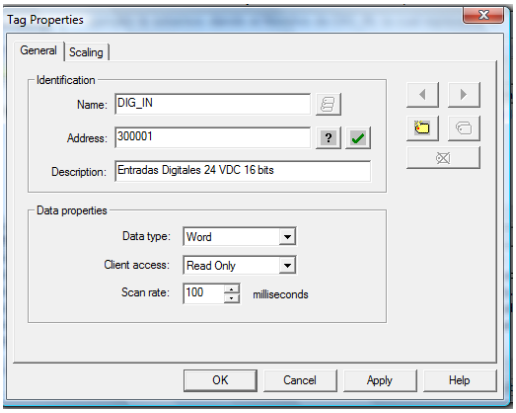


Figura 3.58. Ventana que permite crear tags asignándoles direcciones establecidas en el dispositivo Modbus creado.

Una vez que hayamos creado los Tags correspondientes a las variables a ser leídas y escritas, 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.

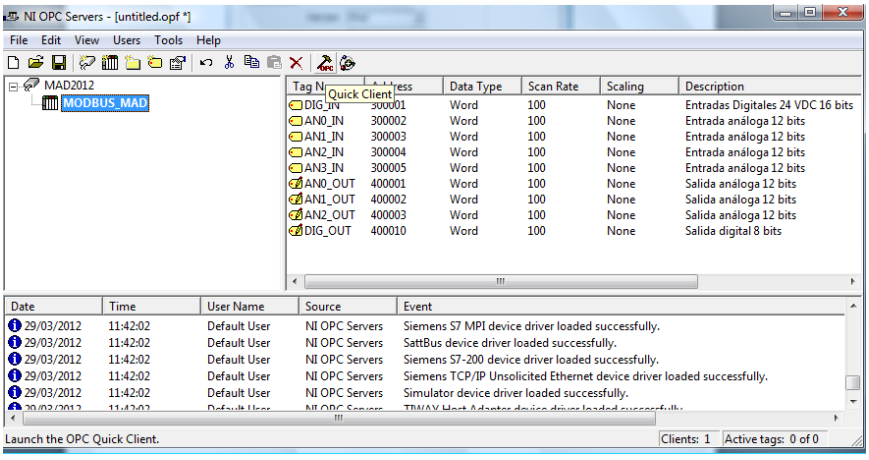


Figura 3.59. Ventana principal, con tags creados

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

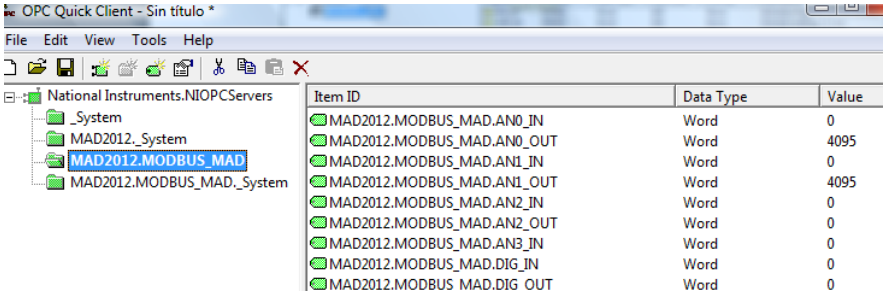


Figura 3.60. Ventana que muestra el OPC cliente obteniendo los valores del dispositivo Modbus esclavo.

Para poder realizar la escritura en los registros es necesario hacer clic derecho sobre el datos del registro que se desea escribir y seleccionamos la opción escritura sincrónica, con la cual nos parece la siguiente ventana en la cual ingresamos el dato que deseamos escribir.

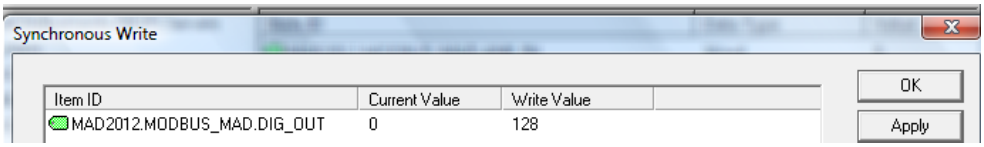


Figura 3.61. Ventana que permite escribir en los registros desde el OPC Server

Ahora podemos decir que hemos terminado de configurar el OPC server de National Instruments para establecer la comunicación Modbus con el dispositivo esclavo que en este caso es el Módulo de adquisición de datos, teniendo las variables creadas en el OPC Server ahora podemos desde Labview tener acceso a estos tags, de la siguiente manera.

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.

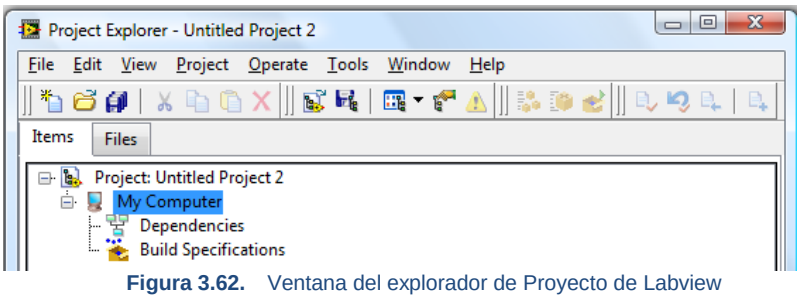


Figura 3.62. Ventana del explorador de Proyecto de Labview

En My Computer hacemos clic derecho y seleccionamos la opción; nuevo I/O Server

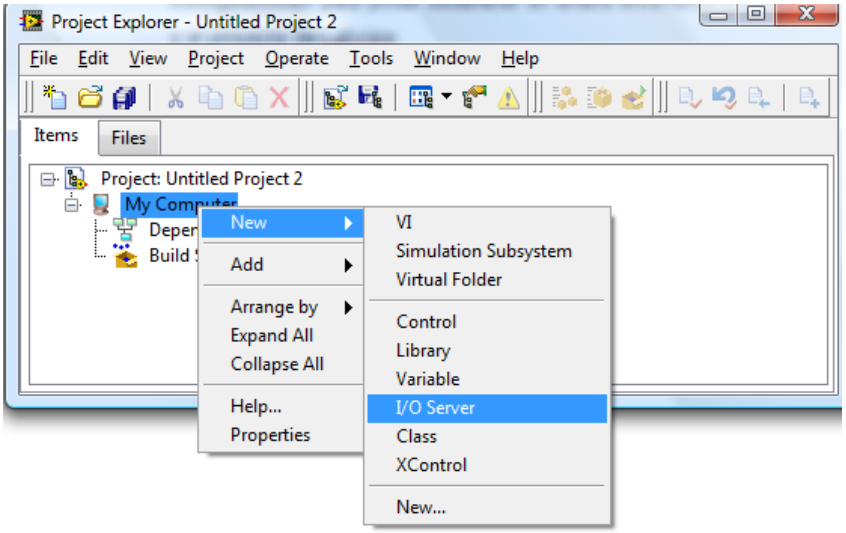


Figura 3.63. Ventana de explorador proyecto nuevo Labview

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, por lo que también podemos establecer la comunicación directa sin tener que usar en OPC. Pero en este caso vamos a utilizar el OPC que creamos ya que no solo nos servirá para NI sino que puede ser utilizado en otros programas de diseño de HMI’s

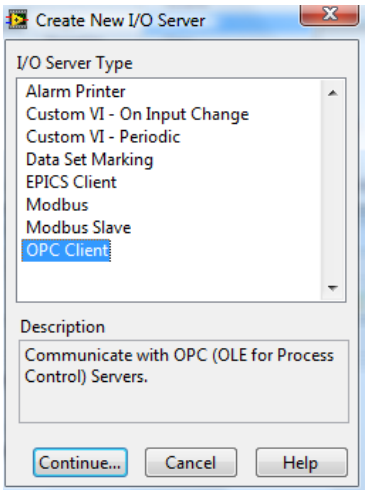


Figura 3.64. Ventana para creación de I/O Server

Luego se obtiene la siguiente ventana, donde indicamos que tipo de OPC vamos a usar.

Aquí elegimos NI OPC SERVER ya que es donde configuramos la conexión con el módulo de adquisición 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.

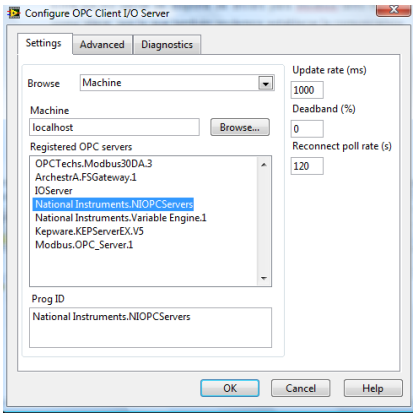


Figura 3.65. Seleccionamos el tipo de OPC Server.

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

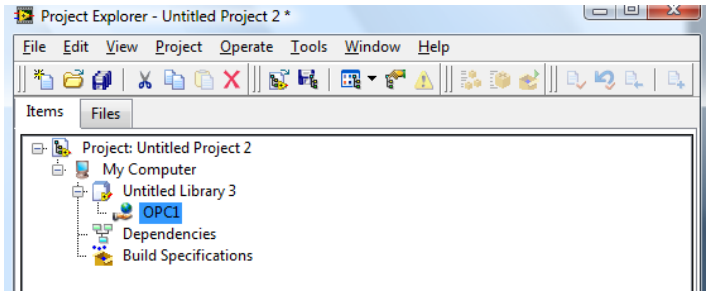


Figura 3.66. Explorador de Proyectos de Labview con e OPC creado

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

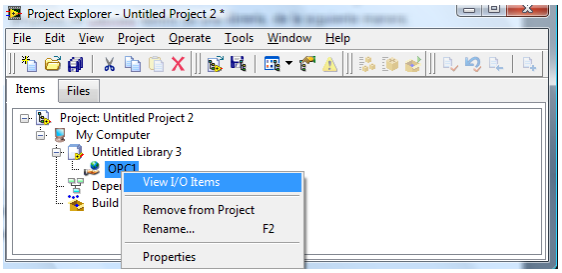


Figura 3.67. OPC dentro del proyecto de Labview

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.

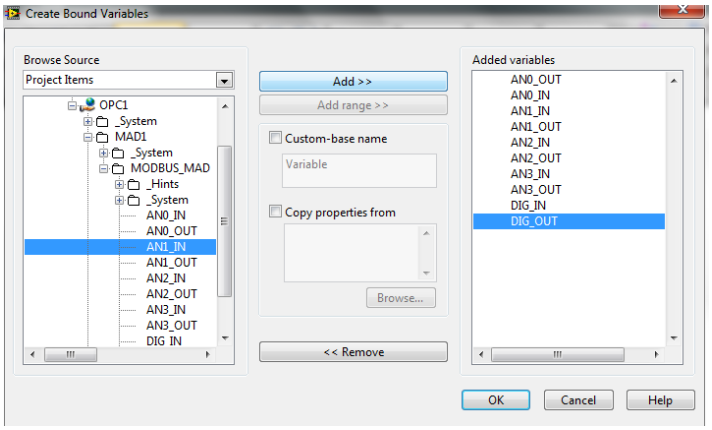


Figura 3.68. Variables del OPC server son cargadas en el proyecto de Labview

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.

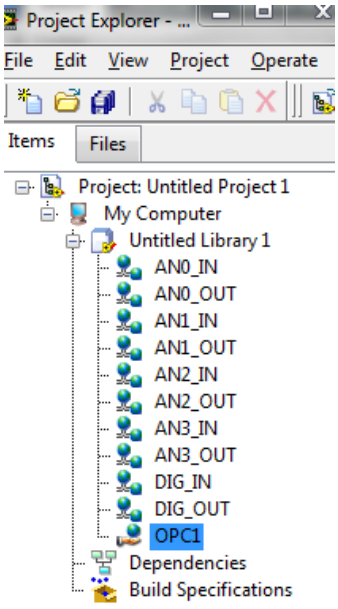
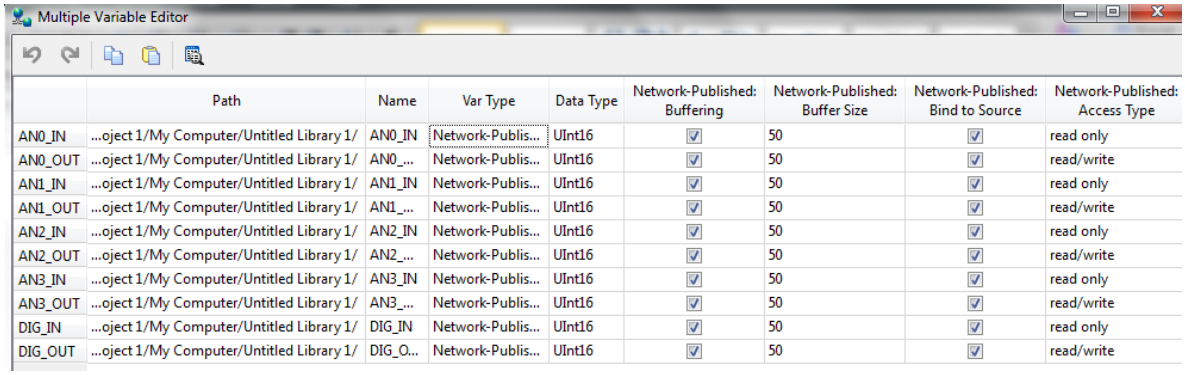


Figura 3.69. Variables creadas se agregan en el explorador del proyecto



	Path	Name	Var Type	Data Type	Network-Published: Buffering	Network-Published: Buffer Size	Network-Published: Bind to Source	Network-Published: Access Type
AN0_IN	...oject 1/My Computer/Untitled Library 1/	AN0_IN	Network-Publis...	UInt16	☑	50	☑	read only
AN0_OUT	...oject 1/My Computer/Untitled Library 1/	AN0_...	Network-Publis...	UInt16	☑	50	☑	read/write
AN1_IN	...oject 1/My Computer/Untitled Library 1/	AN1_IN	Network-Publis...	UInt16	☑	50	☑	read only
AN1_OUT	...oject 1/My Computer/Untitled Library 1/	AN1_...	Network-Publis...	UInt16	☑	50	☑	read/write
AN2_IN	...oject 1/My Computer/Untitled Library 1/	AN2_IN	Network-Publis...	UInt16	☑	50	☑	read only
AN2_OUT	...oject 1/My Computer/Untitled Library 1/	AN2_...	Network-Publis...	UInt16	☑	50	☑	read/write
AN3_IN	...oject 1/My Computer/Untitled Library 1/	AN3_IN	Network-Publis...	UInt16	☑	50	☑	read only
AN3_OUT	...oject 1/My Computer/Untitled Library 1/	AN3_...	Network-Publis...	UInt16	☑	50	☑	read/write
DIG_IN	...oject 1/My Computer/Untitled Library 1/	DIG_IN	Network-Publis...	UInt16	☑	50	☑	read only
DIG_OUT	...oject 1/My Computer/Untitled Library 1/	DIG_O...	Network-Publis...	UInt16	☑	50	☑	read/write

Figura 3.70. 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.

Ahora en el explorador de proyecto creamos un nuevo VI, y arrastramos cada variable del proyecto hasta el panel frontal del editor de Labview.

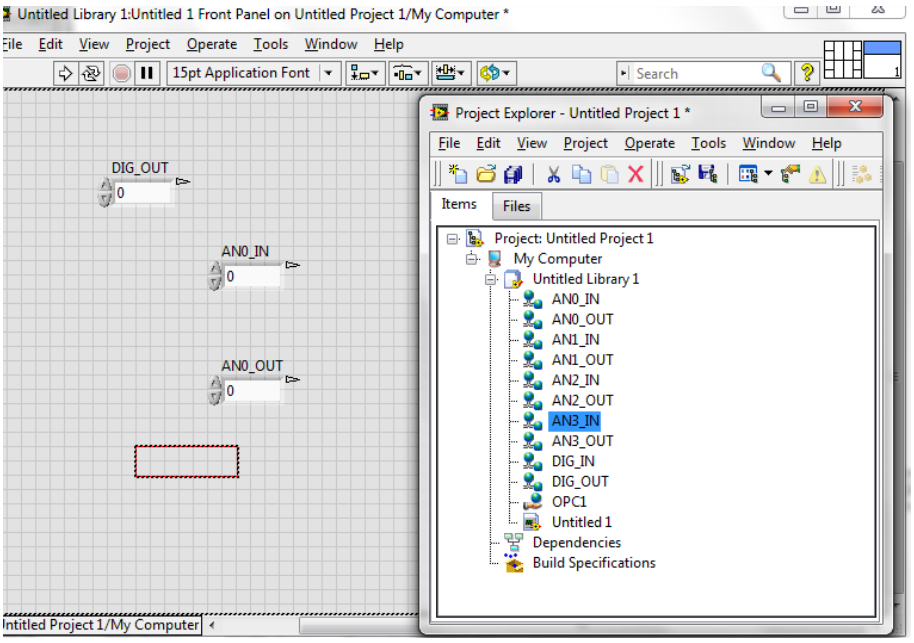


Figura 3.71. Agregando las variables al panel frontal del editor del nuevo VI

Una vez que se tiene las variables obtenidas del módulo de control y monitoreo dentro del diagrama de bloques podemos utilizarlas como mejor nos convenga.

3.4.2.5. Configuración I2C Master con el compilador CCS para el PIC16F887

A continuación se va a detallar las configuraciones que debe tener el microcontrolador PIC16F887 para poder actuar como Master I2C y se indicará las subrutinas para escritura y lectura.

```
//Configuramos el microcontrolador como dispositivo I2C Master el cual va //a
usar los pines C4 y C3 como SDA y SCL respectivamente, además se lo //configura
para que trabaje en modo I2C por software, con lo cual usa las //funciones
propias del módulo MSSP del microcontrolador.
#use   i2c(Master,sda=PIN_C4,scl=PIN_C3,FORCE_SW)
```

Las siguientes líneas de programa van a corresponder a las subrutinas de lectura y escritura mediante el protocolo I2C.

Escritura I2C.

La escritura I2C en nuestro caso la vamos a realizar solamente para enviar el dato de salidas discretas el PIC16F876A que actúa como esclavo I2C y activa los 8 relés correspondientes a las salidas discretas.

La trama enviada desde el Maestro será la siguiente:

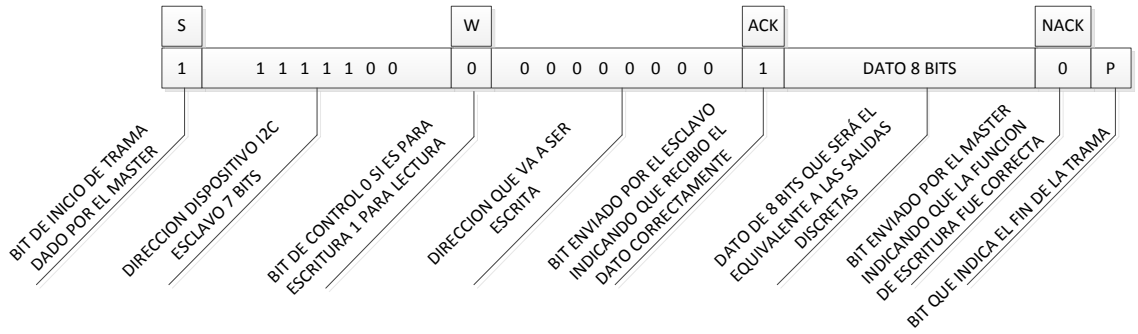


Figura 3.72. Trama que envía el PIC Master I2C hacia el esclavo I2C para poder escribir las salidas discretas.

Este es el código de programación correspondiente a la subrutina de escritura I2C. Tomando en cuenta que el microcontrolador PIC16F876A tiene por dirección el número binario 1111100, el cual se complementa con el bit de escritura que en esta caso es 0, y la dirección interna de este microcontrolador es la 00h, donde se almacenará el dato que a su vez será el equivalente a los 8 relés que son el elementos actuadores.

```
//como se puede ver la subrutina se llama write_ext_i2c, y el //microcontrolador
la hace efectiva cada vez que recibe un dato vía Modbus //en la dirección 09h en
los registros de escritura y lectura.
//void write_ext_i2c(long int address, BYTE data)
{
//define la variable status como short para poder usarla como bandera
   short int status;
//envía el bit de inicio de transmisión
   i2c_start();
//envía la dirección del esclavo incluyendo el bit que indica escritura.
   i2c_write(248);
//envía el dato a ser escrito en la dirección 00h del PIC16F876A
// el dato a ser escrito en el orden de la trama Modbus es el 4
   i2c_write(modbus_rx.data[3]);
//Envía el bit de fin de la transmisión
   i2c_stop();
//Para confirmar que la transmisión concluyó con éxito y para liberar al
//esclavo enviamos un bit de único de trama y esperamos que el esclavo
//responda el bit de ACK para confirmar que la transmisión fue exitosa
   i2c_start();
   status=i2c_write(248);
```

```
//Mientras el bit de ACK no cambie de estado de 1 a 0 se encierra en esta //lazo
while para confirmar que el esclavo fue liberado y poder continuar //con el
programa.
  while(status==1)
   {
      i2c_start();
      status=i2c_write(248);
  }
}
```

Lectura I2C.

La lectura I2C en nuestro caso la vamos a realizar solamente para recibir el dato correspondiente a las entradas discretas conectadas al PIC16F876A que actúa como esclavo I2C y recibe las señales optoacopladas de 24 voltios DC.

La trama enviada desde el Maestro para solicitar los datos de entradas será la siguiente:

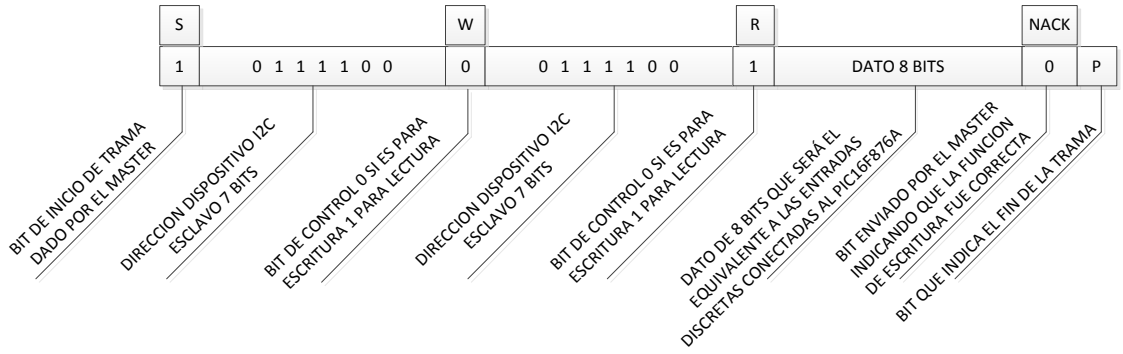


Figura 3.73. Trama que envía el PIC Master I2C hacia el esclavo I2C para poder leer las entradas discretas

Este es el código de programación correspondiente a la subrutina de lectura I2C. Tomando en cuenta que el microcontrolador PIC16F876A tiene por dirección el número binario 0111100, el cual se complementa con el bit de lectura que en esta caso es 1, y la dirección interna de este microcontrolador es la 00h, donde se almacenará el dato que a su vez será el equivalente a las entradas discretas optoacopladas conectadas a este microcontrolador.

```
//como se puede ver la subrutina se llama read_ext_i2c, y el //microcontrolador
la hace efectiva cada vez que recibe una solicitud de //lectura vía Modbus con
la función 4 en las direcciones 00h hasta la 04h //en los registros de lectura.
BYTE read_ext_i2c(long int address) {
//envía el bit de inicio de transmisión
   i2c_start();
//envía la dirección del esclavo incluyendo el bit que indica escritura.
```

```
   i2c_write(120);
//envía el bit de inicio de transmisión
i2c_start();
//envía la dirección del esclavo incluyendo el bit que indica lectura.
   i2c_write(121);
//guarda el dato leído en la variable data
   data=i2c_read(0);
//Envía el bit de fin de la transmisión
   i2c_stop();
//retorna el valor guardado en la variable para posteriormente trasmitirlo //vía
Modbus.
   return(data);
```

Configuración SPI Master con el compilador CCS para el PIC16F887

A continuación se va a detallar las configuraciones que debe tener el microcontrolador PIC16F887 para poder actuar como Master SPI y se indicará las subrutinas para escritura y lectura.

Hay que tener en cuenta que por la forma que trabaja el protocolo SPI se han asignado pines de habilitación del Microcontrolador para poder habilitar el integrado con el cual se va a establecer la comunicación y deshabilitar los otros integrados que no intervienen en el intercambio de datos.

Hay que señalar que se van a utilizar 4 integrados como esclavos SPI estos son 2 MCP4822 (conversor Digital-Análogo) y 2 MCP3202 (Conversor Análogo-Digital)

```
//Configuramos el microcontrolador como dispositivo SPI Master el cual va //a
usar los pines C1 como Clock, C5 como datos de salida, C2 como datos //de
entrada además se configura para que trabaje con grupos de 8 bits de //lectura y
escritura, con una velocidad de 10kbps, ordenando al bit mas //significativo
primero y se va a llamar DA_spi cada vez que se realice //una lectura o una
escritura.
#use spi(MASTER,CLK=PIN_C1, DO=PIN_C5, DI=PIN_C2, BITS=8, baud = 10000, MODE=2,
MSB_FIRST, stream=DA_spi)
```

Las siguientes líneas de programa van a corresponder a las subrutinas de lectura y escritura mediante el protocolo SPI.

Escritura SPI.

La escritura SPI en nuestro caso la vamos a realizar solamente para enviar el dato de salidas análogas a los integrados MCP4822 que actúan como esclavo SPI y convierten el dato digital en una señal análoga de voltaje de 12 bits de resolución.

Este es el código de programación correspondiente a la subrutina de escritura SPI. Tomando en cuenta que el integrado MCP4822 debe ser seleccionado por medio de su pin Chip Select, poniéndolo en estado bajo para que el integrado seleccionado sepa que los datos que se la van a entregar son correspondientes a el dato que requiere ser convertido en señal análoga de voltaje.

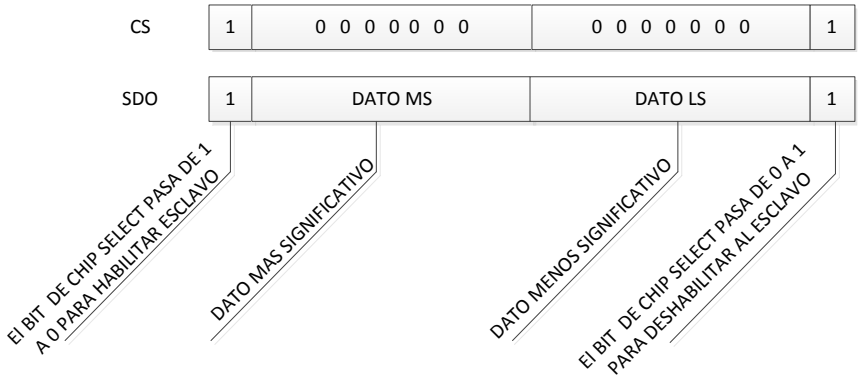


Figura 3.74. Trama que envía el PIC Master SPI para escribir a un esclavo

```
//La subrutina se llama write_SPI0
void write_SPI0()
{
//ponemos en 0 el pin D1 con el cual seleccionamos el integrado con el //cual
vamos a establecer la comunicación.
      output_low(PIN_D1);
//Retardo de 10 milisegundos
      delay_ms(10);
//Escribimos el dato contenido en la variable anout1 que corresponde al //byte
más significativo de la palabra que estamos enviando.
      spi_xfer(DA_spi,anout1);
//Escribimos el dato contenido en la variable anout2 que corresponde al //byte
menos significativo de la palabra que estamos enviando.
      spi_xfer(DA_spi,anout2);
//Retardo de 10 milisegundos
      delay_ms(10);
//Colocamos en 1 el integrado ya que hemos terminado la transmisión y de //esta
manera le estamos liberando
      output_high(PIN_D1);
}
```

Lectura SPI.

La lectura SPI en nuestro caso la vamos a realizar solamente para recibir el dato correspondiente a las entradas análogas conectadas a los integrados MCP3202 que actúa como esclavo SPI y reciben las señales de voltaje de 0 a 5 voltios y lo convierten en dato digital de 12 bits de resolución.

La trama enviada desde el Maestro para solicitar los datos de entradas será la siguiente:

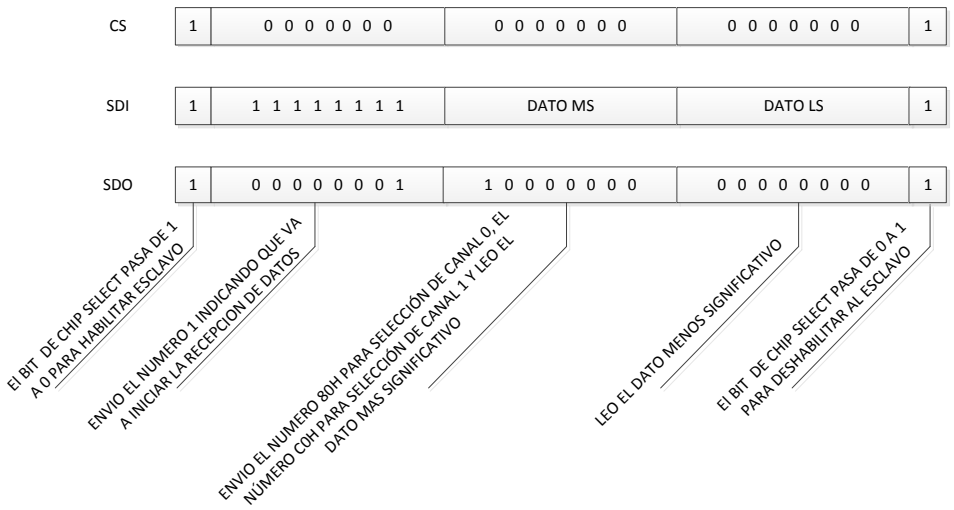


Figura 3.75. Trama que envía el PIC Master SPI para leer a un esclavo

Este es el código de programación correspondiente a la subrutina de lectura SPI.

```
//La subrutina se llama read_SPI0()
void read_SPI0()
{
//Ponemos en bajo el pin de selección que corresponde al integrado en esta
//caso el PIN D4
      output_low(PIN_D4);
//Retardo de 20 milisegundos
      delay_ms(20);
//Enviamos el bit de inicio de transmisión, con lo cual alertamos al //integrado
que estamos listos para recibir los datos.
      spi_xfer(DA_spi,1);
//enviamos el 80h para seleccionar el canal 1 y a la vez recibimos el dato //más
significativo y lo guardamos en la variable dato1
      dato1 = spi_xfer(DA_spi,0x80);
//recibimos el dato menos significativo y lo guardamos en la variable //dato2
      dato2 = spi_xfer(DA_spi,0);
//enmascaro la variable dato1 para eliminar los 4 bits más significativos //ya
que solo nos interesan los 4 bits menos significativos por ser de 12 //bits de
resolución y el resultado lo guardamos en dato3
      dato3 = dato1&15;
//retardo de 20 milisegundos
      delay_ms(20);
//liberamos al integrado poniendo en 1 el pin de selección.
      output_high(PIN_D4);
```

3.4.3. Configuración USB y Modbus Master para el PIC18F2550 en CCS.

Como lo habíamos indicado en anteriores apartados dentro de este mismo documento, el PIC18F2550 actúa como in dispositivo Gateway entre la comunicación NI VISA-USB y Modbus, por lo que se lo ha configurado para que trabaje como Modbus Master, utilizando la misma librería Modbus.c que se utilizó para el Microcontrolador configurado como Modbus esclavo, y además posee la capacidad de conectarse a un computador utilizando el driver de NI VISA de National Instruments, a continuación se va a detallar las configuraciones que fueron utilizadas para lograr que mencionado microcontrolador trabaje de manera eficaz, usando ambas comunicaciones a la vez.

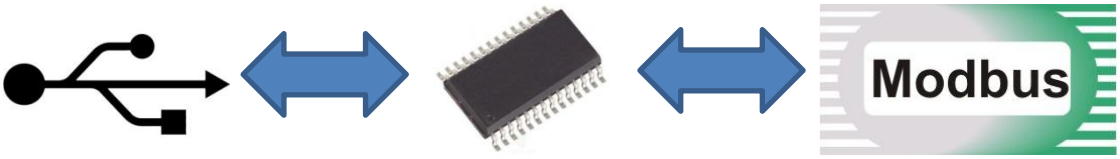


Figura 3.76. Diagrama básico de enlace entre PIC18F2550 vía USB y Modbus

Vamos a presentar funciones básicas de configuración de los puertos tanto USB como USART trabajando con Modbus del microcontrolador y las subrutinas necesaria para lograr la comunicación.

3.4.3.1. Configuración Modbus Master PIC18F2550

Dentro del desarrollo de la comunicación Modbus modo Master, para poder realizar un monitoreo de la señales de entrada tanto análogas como digitales y poder realizar un control de sobre las señales de salida discretas y análogas, que las vamos a adquirir desde el Microcontrolador Modbus Esclavo, hemos utilizado las siguientes funciones Modbus.

- Función 3: Read Holding Registers (Registros de Lectura y escritura), Función que vamos a utilizar para leer los registros de escritura del dispositivo esclavo

- Función 4: Read Input Registers (Registros solo de lectura), Función que vamos a utilizar para leer los registros de lectura del dispositivo esclavo.
- Función 6: Write Single Register (Escribir un solo registro), Función que vamos a utilizar para escribir un registro a la vez del Modbus esclavo previamente seleccionando la dirección del esclavo y del registro que vamos a escribir.

Las funciones de escritura y lectura trabajan tal cual se mencionó en la parte de la configuración de dispositivo esclavo, los diagramas de flujo son prácticamente los mismos, ya que establecen la comunicación Master-Slave y usan las mismas tramas, funciones y códigos de error.

A continuación se van a presentar las líneas básicas del programa para la configuración Modbus Master.

Como se dijo antes, para poder configurar la comunicación Modbus se utilizó la librería Modbus.c que viene con el compilador CCS y posee las subrutinas que ejecutan las diferentes funciones Modbus, con las cuales vamos a trabajar en el programa principal para poder establecer la comunicación para cumplir las funciones que requerimos.

```
//Incluimos las librerías del PIC18F2550 el cual vamos a configurar
#include <18F2550.h>
//Configuramos los fusibles de Microcontrolador, Cristal externo de alta
//velocidad, se deshabilita el WDT, sin código de protección, reset cuando
//exista bajo voltaje, se deshabilita programación con bajo voltaje, se
//configura en modo USB para trabajar con cristal de 20Mhz.
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
//Establecemos que vamos a trabajar con un cristal de 20MHz
#use         delay(clock=20000000)
//Seleccionamos para que el PIC18F2550 trabaje en modo Modbus Master.
#define      MODBUS_TYPE MODBUS_TYPE_MASTER
//Definimos como 64 bytes el tamaño del buffer de recepción.
#define      MODBUS_SERIAL_RX_BUFFER_SIZE    64
//Establecemos como 9600 baudios la velocidad de transmisión de datos
#define      MODBUS_SERIAL_BAUD              9600
//Habilita la interrupción por recepción serial
#define      MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
//Incluye la librería MODBUS.C ubicada en el directorio del Compilador CCS
#include     "MODBUS.c"
//Establecemos que la dirección Modbus estará dada por un dato de //dirección
enviado vía USB llamado MDB_SLV.
#define      MODBUS_ADDRESS MDB_SLV
```

Una vez definidas las configuraciones básicas de Modbus para el microcontrolador, se establecen las direcciones de los registros dentro de la memoria del microcontrolador y con éstas definidas se puede realizar las subrutinas respectivas correspondientes a cada variable ya sea de entrada o de salida.

3.4.3.2. Configuración USB NI-VISA para el PIC18F2550

Básicamente la gestión de datos se la va a realizar dentro del lazo principal del programa el cual es controlado por la comunicación USB por lo que a continuación se va a detallar las diferentes configuraciones para lograr la comunicación USB con el driver de NI VISA.

El siguiente diagrama de flujo muestra el proceso de gestión de datos desde de USB a Modbus.

El PIC18F2550 recibe 6 bytes vía USB desde Labview, el primer dato en recibir Dato[0], puede tener 2 valores, 0 ó 1.

Si Dato[0] toma el valor de 1 Labview está solicitando el PIC18F2550 los datos correspondientes a las entradas discretas y análogas divididos en 10 bytes según la siguiente tabla:

Identificador USB	Dato que corresponde a:
oBuff[0]	Byte más significativo Entradas discretas
oBuff[1]	Byte menos significativo Entradas discretas
oBuff[2]	Byte más significativo canal análogo 1
oBuff[3]	Byte menos significativo canal análogo 1
oBuff[4]	Byte más significativo canal análogo 2
oBuff[5]	Byte menos significativo canal análogo 2
oBuff[6]	Byte más significativo canal análogo 3
oBuff[7]	Byte menos significativo canal análogo 3
oBuff[8]	Byte más significativo canal análogo 4
oBuff[9]	Byte menos significativo canal análogo 4

Tabla 3.9. Lista de datos que ingresan a la aplicación VI utilizando VISA.

- Si Dato[0] toma el valor de 0 Labview se abra la opción para leer o escribir las registros Modbus del dispositivo esclavo, así que ahora trabajamos con la variable de entrada USB Dato[1], la cual puede tener dos valores, A o B.

- Si Dato[1] toma el valor de A Labview, solicita que se realice una lectura de los registros de entradas del dispositivo esclavo y los almacene en registros temporales, acción mediante la cual se obtienen los datos que se detallaron en la tabla anterior y que serán enviados vía USB cuando Dato[0] tome el valor de 1.
- Si Dato[1] toma el valor de B Labview, va a escribir sobre los registros de escritura del dispositivo esclavo, teniendo en cuenta que
- Dato[5] indicará la dirección del Esclavo Modbus,
- Dato[2] indica la dirección del registro que vamos a escribir y
- Dato[4] y Dato[3] son el dato de 16 bits que será escrito en el esclavo Modbus.

La siguiente tabla resume lo antes expuesto:

Datos que intervienen en la comunicación Modbus			
Nombre	Rango	Operación	
Dato[0]	0,1	Si es = 0 habilita opciones para leer o escribir El Modbus esclavo	Si es = 1 envía los datos de entrada recibidos
Dato[1]	A,B	Si es = A, Lee los registros internos de entrada del esclavo Modbus	Si es = B, Habilita opción escribir un registro del esclavo Modbus
Dato[2]	0-255	Dirección Registro interno	
Dato[3]	0-255	Byte menos significativo dato a escribir	
Dato[4]	0-255	Byte más significativo dato a escribir	
Dato[5]	0-255	Dirección Esclavo Modbus	

Tabla 3.10. Datos que interviene en la comunicación Modbus

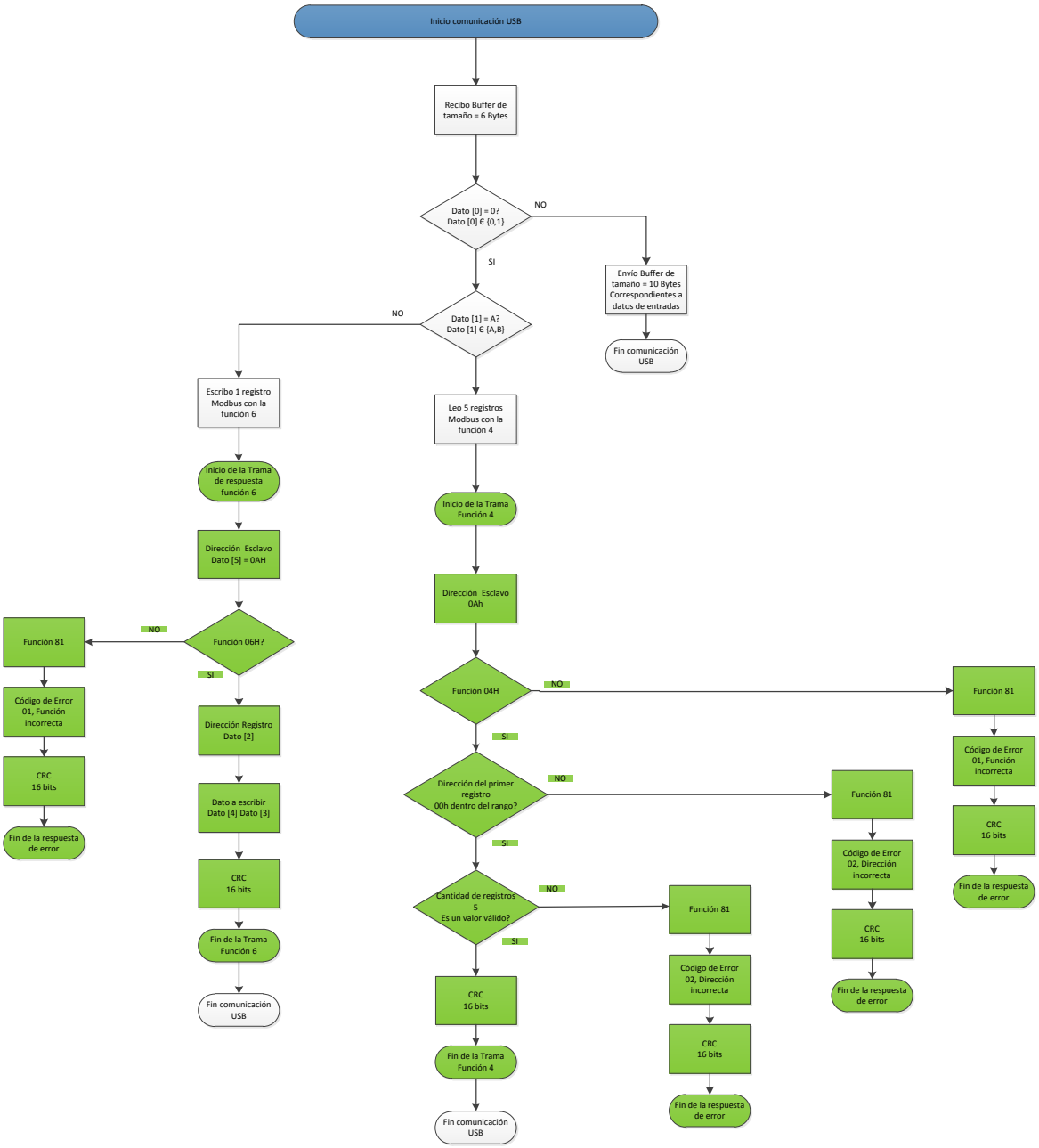


Figura 3.77. Diagrama de flujo de las funciones que cumple el PIC18F2550 como Modbus Master

Lo que a continuación se va a presentar son las configuraciones básicas acerca de la comunicación USB con el Microcontrolador PIC18F2550, vamos a detallar todo lo que tiene que ver con la configuración ya que anteriormente ya se ha mencionado acerca de la creación del Driver e instalación.

Unas de las cosas importantes que tenemos que tener en cuenta al programar el Microcontrolador son los fusibles, en los que entre otras cosas se define la

velocidad de trabajo del PIC. Una característica que hay que tener en cuenta cuando usamos un PIC con interfaz USB, es que para que dicho módulo funcione la frecuencia de reloj en la entrada de dicho módulo debe de ser de 48 MHz, para conseguir dicha frecuencia se dispone de un multiplicador con pre-escaler y post-escaler. A la entrada del multiplicador tenemos que tener una frecuencia fija de 4 MHz. En la figura de abajo se muestra como configurar el pre-escaler del PLL cuando en nuestro circuito tenemos un cristal de 20 MHz.

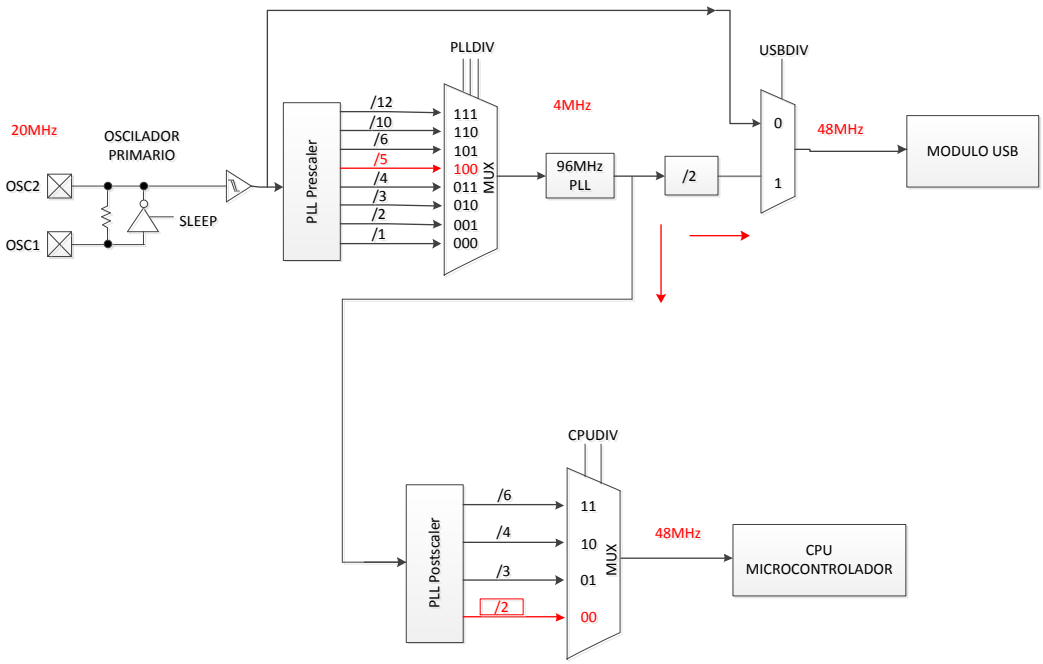


Figura 3.78. Diagrama configuración de Oscilador para el Módulo USB y para el CPU del Microcontrolador PIC28F2550

Como se ve en la figura, en este caso el pre-escaler divide los 20 MHz por 5 para obtener los 4MHz requeridos a la entrada del PLL, este a su vez produce 96 MHz en su salida los cuales se distribuyen por un lado al módulo USB, dividiendo previamente la frecuencia por 2 para obtener los 48 MHz a la entrada del módulo por el otro lado alimenta el post-divisor del PLL, para que podamos elegir la frecuencia de trabajo del núcleo del micro, en este caso divide por dos por lo que tendremos 48 MHz para alimentar al "CPU" del PIC.

Debido a la complejidad de la comunicación USB, lo que ha hecho tanto Microchip como otros desarrolladores de compiladores como CCS es proporcionar unas

librerías que comúnmente se les llaman "Stacks" o pila de software cuya finalidad es facilitar la tarea al programador de dispositivos.
En nuestro caso, las funciones que se han utilizado para enviar y recibir datos por el bus USB son:

- **usb_kbhit(1).** Es una función que retorna el valor booleano TRUE si hay uno o más caracteres esperando en el buffer de recepción.
- **usb_get_packet(1, iBuff, 6).** Obtiene el carácter recibido en el Buffer de recepción.
- **usb_put_packet(1, oBuff, 10).** Coloca el carácter que recibe como parámetro en el buffer de transmisión para ser enviado. Hay más funciones disponibles para su uso directo, la descripción de cada una de ellas las podéis encontrar en la cabecera del archivo pic18_usb.h.
- **usb_init(),** La comunicación USB ha de ser inicializada, eso se consigue llamando a la función: al principio de la función main() .

De las librerías que nos proporciona CCS para la comunicación USB podemos editar y modificar los descriptores pertenecientes al VID, PID, consumo del dispositivo y versión del firmware.

```
char const USB_DEVICE_DESC[] ={
USB_MAX_EP0_PACKET_LENGTH,
//El VID es un número de 16 bits que significa Vendor Identification o //código que identifica al fabricante del hardware a conectar.  (0x04D8 is Microchip)
   0xD8,0x04,
//El PID es un número de 16 bits que significa Product Identification o //código que identifica al dispositivo en concreto hardware a conectar //(PicUsbVISA 0x0BB0).
   0xB0,0x0B,
//Estos dos numeros VID&PID nos va a servir para conectar con el Driver de //Windows
```

Si modificamos el VID y el PID tendremos que modificarlo también en el archivo .INF, creado con el driver Wizard de National Intstruments.

Otra cosa que se puede modificar en este archivo son los "Strings" o cadenas de caracteres que sirven para que Windows muestre información referente al fabricante del dispositivo, como por ejemplo nombre de la compañía y nombre del producto. Estas cadenas están colocadas al final del archivo descriptor.h y hay que colocarlas en un formato determinado según se muestra

```
#define USB_STRING_DESC_COUNT sizeof(USB_STRING_DESC_OFFSET)

// La tabla USB_STRING_DESC contiene la descripción del dispositivo
//detectado por el Driver de Windows y que nos va a mostrar en la
//correspondiente entrada en la lista del Hardware Instalado en el
//Sistema.

char const USB_STRING_DESC[]={

//string 1 --> la compañia del producto ???

USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
   '@',0,
   'A',0,
   '@',0,
//string 2 --> nombre del dispositivo

USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING)
   'P',0,
   'i',0,
   'c',0,
   'U',0,
   's',0,
   'b',0,
   'V',0,
   'I',0,
   'S',0,
   'A',0
};
```

Estos Strings también pueden ser ingresados al momento de crear el driver en el Driver Wizard de National Instruments.

Un vez que se conecta el dispositivo en el puerto USB de un computador, el nombre que aparece cuando es identificador es el que se configura en el comando USB_DESC_STRING_TYPE

3.4.4. Programa para la tarjeta de entradas discretas.

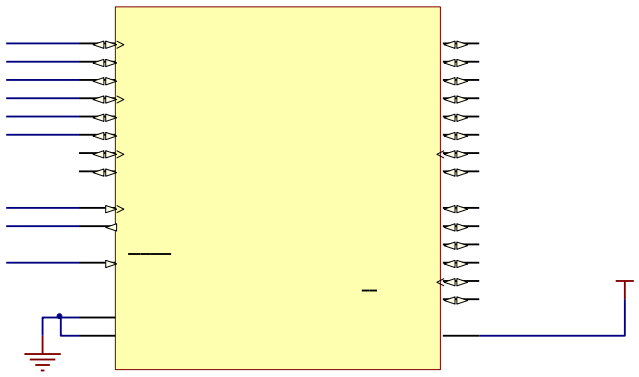


Figura 3.79. Distribución de pines PIC16F876A para entradas discretas

Este módulo posee 2 microcontroladores PIC16F876A, en los cuales están conectadas 8 entradas de 24 voltios optoacopladas en su pines configurados como entradas, cada PIC tiene una dirección I2C esclava por medio de la cual vamos a poder leer el dato de 8 bits correspondiente a sus entradas, una vez que los datos han sido leídos por el Microcontrolador I2C Master, estos datos forman una palabra de 16 bits, la cual será almacenada en la dirección 00 de los registros internos.

A continuación se muestra la distribución de los pines donde fueron conectadas las entradas discretas del PIC16F876A.

PUERTO B	FUNCIÓN	PUERTO C	FUNCIÓN
PIN_B0.0	Entrada digital 1	PIN_C0.0	Entrada digital 7
PIN_B0.1	Entrada digital 2	PIN_C0.1	Entrada digital 8
PIN_B0.2	Entrada digital 3	PIN_C0.2	NC
PIN_B0.3	Entrada digital 4	PIN_C0.3	SCL
PIN_B0.4	Entrada digital 5	PIN_C0.4	SDA
PIN_B0.5	Entrada digital 6	PIN_C0.5	Led Indicador
PIN_B0.6	PIN PGC	PIN_C0.6	NC
PIN_B0.7	PIN PGD	PIN_C0.7	NC

Tabla 3.11. Lista de pines para entradas discretas

Los datos de entradas se almacenan de bit en bit en un registro de salida, el cual es enviado al Master cuando este lo solicita.

Y la configuración es la siguiente:

```
//Configuramos el microcontrolador como dispositivo I2C Esclavo el cual va //a usar los pines C4 y C3 como SDA y SCL respectivamente, además se lo //configura para que trabaje en modo I2C por software, con lo cual usa las //funciones propias del módulo MSSP del microcontrolador.
#use   i2c(Slave,sda=PIN_C4,scl=PIN_C3,FORCE_SW)
```

La lectura I2C en nuestro caso la vamos a realizar solamente para recibir el dato correspondiente a las entradas discretas conectadas al PIC16F876A que actúa como esclavo I2C y recibe las señales optoacopladas de 24 voltios DC.

La trama enviada desde el Maestro para solicitar los datos de entradas será la siguiente:

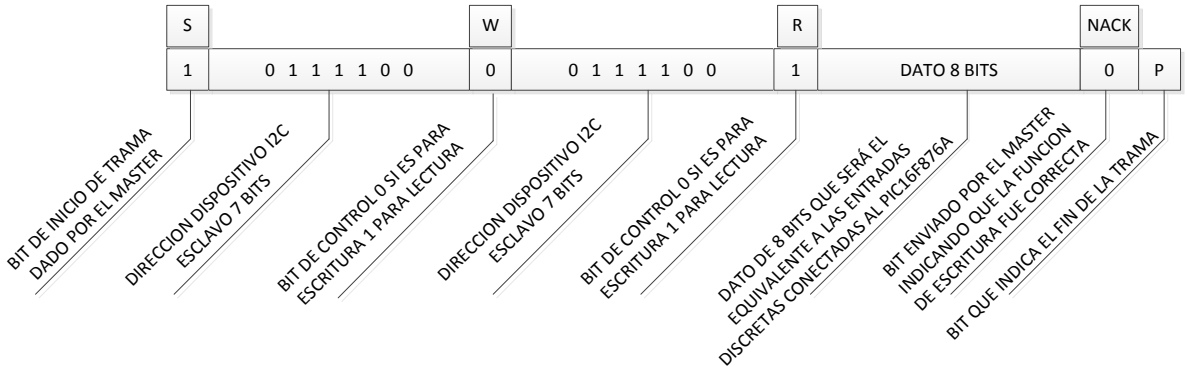


Figura 3.80. Trama que envía el PIC Master I2C hacia el esclavo I2C para poder leer las entradas discretas

Este es el código de programación correspondiente a la subrutina de lectura I2C. Tomando en cuenta que el microcontrolador PIC16F876A tiene por dirección el número binario 0111100, el cual se complementa con el bit de lectura que en esta caso es 1, y la dirección interna de este microcontrolador es la 00h, donde se almacenará el dato que a su vez será el equivalente a las entradas discretas optoacopladas conectadas a este microcontrolador.

```
//como se puede ver la subrutina se llama read_ext_i2c, y el //microcontrolador la hace efectiva cada vez que recibe una solicitud de //lectura vía Modbus con la función 4 en las direcciones 00h hasta la 04h //en los registros de lectura.
BYTE read_ext_i2c(long int address) {
//envía el bit de inicio de transmisión
i2c_start();
//envía la dirección del esclavo incluyendo el bit que indica escritura.
```

```
   i2c_write(120);
//envía el bit de inicio de transmisión
i2c_start();
//envía la dirección del esclavo incluyendo el bit que indica lectura.
   i2c_write(121);
//guarda el dato leído en la variable data
   data=i2c_read(0);
//Envía el bit de fin de la transmisión
   i2c_stop();
//retorna el valor guardado en la variable para posteriormente trasmitirlo //vía
Modbus.
   return(data);
}
```

3.4.5. Programa para la tarjeta de salidas discretas.

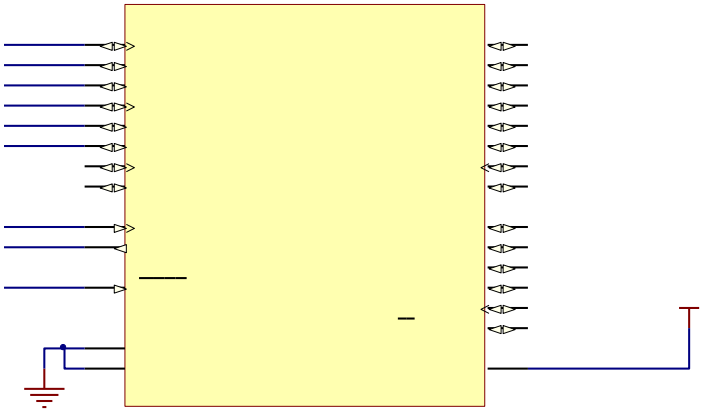


Figura 3.81. Distribución de pines PIC16F876A para salidas discretas

Este módulo posee 1 microcontrolador PIC16F876A, en el cual están conectado 8 relés con bobina de 12 voltios, el PIC tiene una dirección I2C esclava por medio de la cual vamos a poder escribir un dato de 8 bits correspondiente a sus salidas, una vez que los datos han sido enviados por el Microcontrolador I2C Master, estos datos forman una byte, la cual será almacenado en la dirección 00 de los registros internos.

A continuación se muestra la distribución de los pines donde fueron conectadas las salidas discretas del PIC16F876A.

PUERTO B	FUNCIÓN	PUERTO C	FUNCIÓN

PIN_B0.0	Salida digital 1	PIN_C0.0	Salida digital 7
PIN_B0.1	Salida digital 2	PIN_C0.1	Salida digital 8
PIN_B0.2	Salida digital 3	PIN_C0.2	NC
PIN_B0.3	Salida digital 4	PIN_C0.3	SCL
PIN_B0.4	Salida digital 5	PIN_C0.4	SDA
PIN_B0.5	Salida digital 6	PIN_C0.5	Led Indicador
PIN_B0.6	PIN PGC	PIN_C0.6	NC
PIN_B0.7	PIN PGD	PIN_C0.7	NC

Tabla 3.12. Lista de pines para salidas discretas

Los datos de salidas se almacenan de bit en bit en un registro de entrada, el cual es recibido desde Master.

Y la configuración es la siguiente:

```
//Configuramos el microcontrolador como dispositivo I2C Esclavo el cual va //a usar los pines C4 y C3 como SDA y SCL respectivamente, además se lo //configura para que trabaje en modo I2C por software, con lo cual usa las //funciones propias del módulo MSSP del microcontrolador.
#use   i2c(Slave,sda=PIN_C4,scl=PIN_C3,FORCE_SW)
```

La escritura I2C en nuestro caso la vamos a realizar solamente para enviar el dato de salidas discretas el PIC16F876A que actúa como esclavo I2C y activa los 8 relés correspondientes a las salidas discretas.

La trama enviada desde el Maestro será la siguiente:

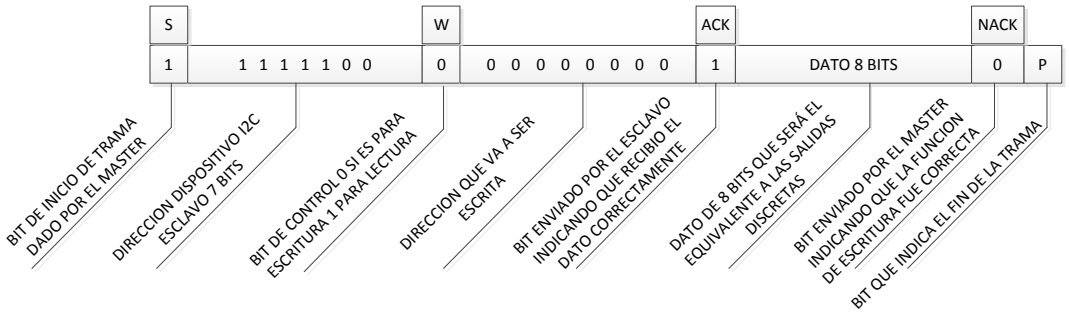


Figura 3.82. Trama que envía el PIC Master I2C hacia el esclavo I2C para poder escribir las salidas discretas.

Este es el código de programación correspondiente a la subrutina de escritura I2C. Tomando en cuenta que el microcontrolador PIC16F876A tiene por dirección el número binario 1111100, el cual se complementa con el bit de escritura que en esta caso es 0, y la dirección interna de este microcontrolador es la 00h, donde se

almacenará el dato que a su vez será el equivalente a los 8 relés que son el elementos actuadores.

```
//como se puede ver la subrutina se llama write_ext_i2c, y el //microcontrolador
la hace efectiva cada vez que recibe un dato vía Modbus //en la dirección 09h en
los registros de escritura y lectura.
//void write_ext_i2c(long int address, BYTE data)
{
//define la variable status como short para poder usarla como bandera
   short int status;
//envía el bit de inicio de transmisión
   i2c_start();
//envía la dirección del esclavo incluyendo el bit que indica escritura.
   i2c_write(248);
//envía el dato a ser escrito en la dirección 00h del PIC16F876A
// el dato a ser escrito en el orden de la trama Modbus es el 4
   i2c_write(modbus_rx.data[3]);
//Envía el bit de fin de la transmisión
   i2c_stop();
//Para confirmar que la transmisión concluyó con éxito y para liberar al
//esclavo enviamos un bit de único de trama y esperamos que el esclavo
//responda el bit de ACK para confirmar que la transmisión fue exitosa
   i2c_start();
   status=i2c_write(248);
  //Mientras el bit de ACK no cambie de estado de 1 a 0 se encierra en esta
//lazo while para confirmar que el esclavo fue liberado y poder continuar //con
el programa.
  while(status==1)
   {
      i2c_start();
      status=i2c_write(248);
   }
}
```

3.4.6. Configuración Módulo de Entradas Análogas.

EL módulo de entradas análogas posee como elemento principal 2 circuitos integrados MCP3202, los cuales son fabricados por Microchip, tiene interfaz SPI, cuenta con 2 canales de entrada analógica de 12 bits de resolución, los datos se almacenan en sus registros internos y por medio de la trama enviada desde el dispositivo SPI Master podemos leer los datos correspondientes a cada canal.

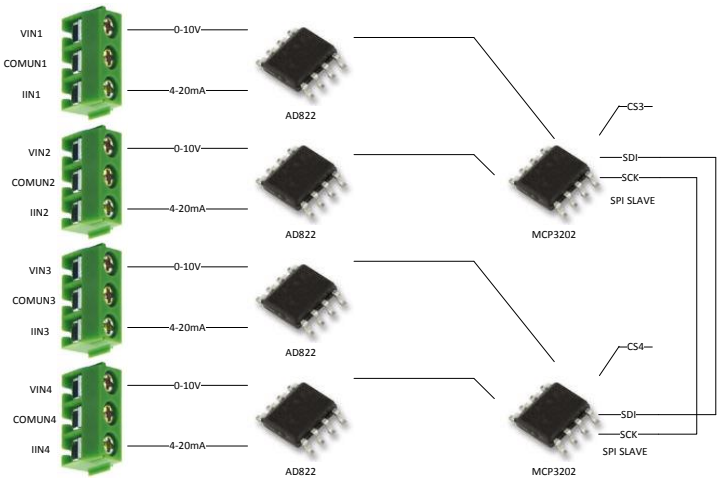


Figura 3.83. Diagrama general Módulo de entradas análogas

3.4.6.1. Comunicación SPI con el MCP3202

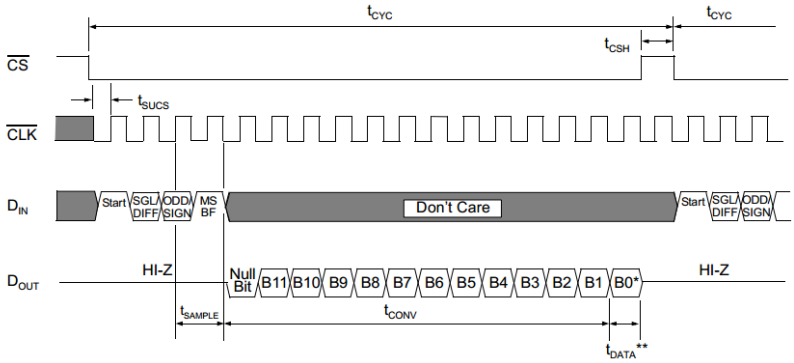


Figura 3.84. Comunicación SPI con MCP3202

Para poder leer el MCP3202 es necesario que el pin de habilitación CS pase de estado 1L a 0L y una vez que se ha terminado la transmisión el pin CS deberá volver a 1L. Mientras esto ocurre, el Pin de Clock empieza a oscilar para poder sincronizar la transmisión de datos desde el Maestro hacia el Esclavo SPI.

El dato se escribe tomando en cuenta los siguientes aspectos.

START	C0	C1	MSBF	X	X	X	X	X	X	X	X	X	X	X	X

Tabla 3.13. Trama que envía el Master para leer el MCP3202

X	X	X	NULL	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 3.14. Trama que recibe el Master del MCP3202

El microcontrolador SPI Maestro transmite en forma serial 1 byte hacia el esclavo, el cual tiene un propósito que se detalla a continuación bit a bit.

START. Es el bit que le indica al MCP3202 esclavo que va a iniciar la trasmisión de datos .

C0-C1. Selecciona el canal que va a ser leído desde el maestro.
C0=1 y C1=0 el canal seleccionado es el CH0
C0=1 y C1=1 el canal seleccionado es el CH1

MSBF. Bit que indica el orden que usará el MCP3202 para transmitir el dato requerido, es decir cuando MSBF = 0 se transmitirá primero el más significativo, si MSBF = 1 se transmitirá primero el menos significativo.

NULL. Se lee como 0L.

D11-D0. Es el dato de 12 bits que se trasmite desde el esclavo hacia el maestro SPI y corresponde al equivalente digital del valor medido en cualquiera de sus dos canales analógicos.

Selección canal CH0

START	C0	C1	MSBF	NULL	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 3.15. Trama para indicarle al MCP3202 que se solicita el dato del Canal 0

Para iniciar la transmisión y recepción de datos con el MCP3202 es necesario primero enviar un bit de inicio, a continuación para leer un dato en el canal 0 es necesario enviar el número hexadecimal 80hex el cual nos garantiza seleccionar el canal 0, y establecer que el primer dato a recibir será el más significativo.

Selección canal CH1

START	C0	C1	MSBF	NULL	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 3.16. Trama para indicarle al MCP3202 que se solicita el dato del Canal 1

Para leer un dato en el canal 1 es necesario enviar el número hexadecimal C0hex el cual nos garantiza seleccionar el canal 1, y establecer que el primer dato a recibir será el más significativo.

A la entrada de cada canal con la ayuda de un amplificador operacional se obtiene el voltaje, correspondiente a la señal de corriente de 4 a 20 mA. Y un potenciómetro en modo de divisor de tensión para la entrada de 0 a 10 voltios

3.4.7. Configuración Módulo de Salidas Análogas.

El módulo de salidas análogas tiene como elemento primario en su funcionamiento el integrado MCP4822, el cual es un convertidor digital análogo de 12 bits de resolución que posee comunicación SPI mediante la cual el microcontrolador PIC16F887 actúa como Master SPI.

EL MCP4822 es un circuito integrado creado por Microchip, el cual tiene 8 pines, posee 2 canales análogos VOUTA y VOUTB, es un convertidor Digital-Análogo de 12 bits de resolución y posee un puerto SPI, por medio del cual recibe los datos y comandos necesarios para poder obtener a la salida de cada canal un valor analógico de 0 a 4.095 voltios.

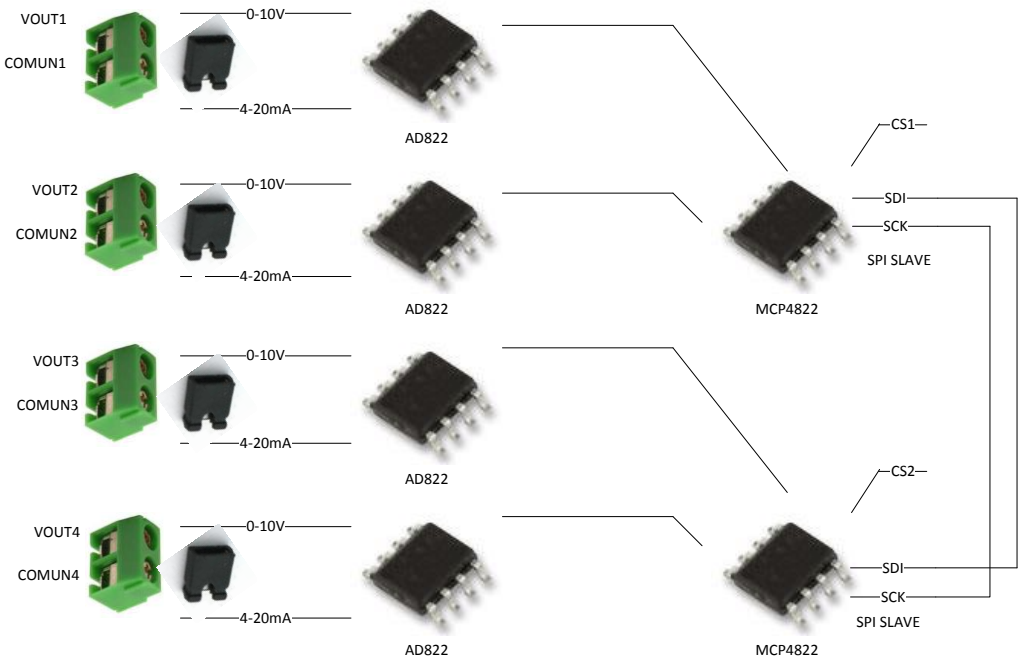


Figura 3.85. Esquema general Módulo de salidas análogas

3.4.7.1. Comunicación SPI con un MCP4822

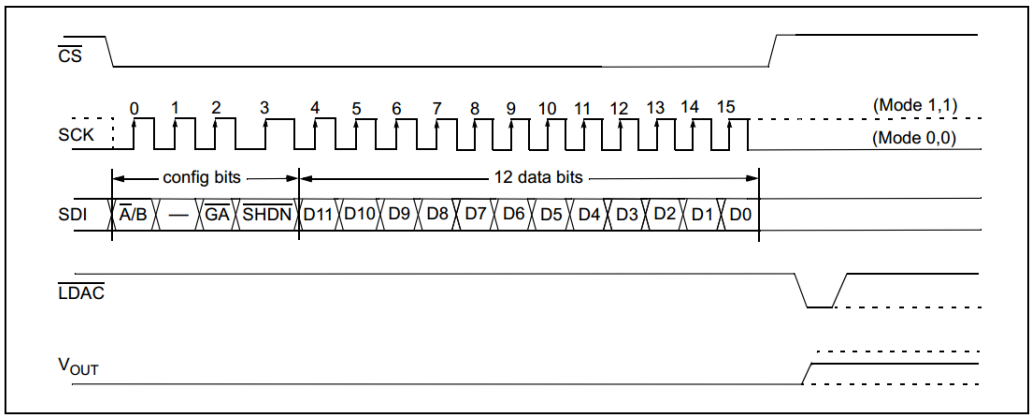


Figura 3.86. Comunicación SPI con MCP4822

Para poder escribir sobre el MCP4822 es necesario que el pin de habilitación CS pase de estado 1L a 0L y una vez que se ha terminado la transmisión el pin CS deberá volver a 1L. Mientras esto ocurre, el Pin de Clock empieza a oscilar para poder sincronizar la transmisión de datos desde el Maestro hacia el Esclavo SPI.
El pin LDAC sirve para trasladar el dato del latch de entrada hacia el dato de latch de salida, en nuestro caso el pin LDAC siempre estará en modo bajo para que una vez terminada la conversión el dato análogo se muestre automáticamente.
El dato se escribe tomando en cuenta los siguientes aspectos.

A/B	----	GA	SDN	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 3.17. Trama para escribir un dato en el MCP4822

El microcontrolador SPI Maestro transmite en forma serial 2 bytes hacia el esclavo, estas palabras tienen un propósito que se detalla a continuación bit a bit.

A/B (16). Es el bit que selecciona el canal de salida del MCP4822 si es igual a 0 el canal es el A, si es igual a 1 el canal es el B.

---- (15). No interviene se escribe como 0L.

GA (14). Bit que controla la ganancia, cuando es 0L la ganancia es 1X, si el 1L la ganancia será de 2X.

SDN(13). Encender o apagar el módulo de conversión, si es 1L el módulo se activa, si está en 0L la salida del canal se desactiva.

D11-D0. Es el dato de 12 bits que se trasmite desde el master el cual será convertido a señal analógica por la salida del canal A o del canal B.

Selección canal A

A/B	----	GA	SDN	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 3.18. Trama para seleccionar el Canal A para la escritura

Para escribir un dato en el canal A es necesario enmascarar el byte más significativo con el número 16, lo cual nos garantiza seleccionar el canal A, mantener encendido el conversor análogo digital con ganancia 1x.

Selección canal B

A/B	----	GA	SDN	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Tabla 3.19. Trama para seleccionar el Canal B para la escritura

Para escribir un dato en el canal B es necesario enmascarar el byte más significativo con el número 144, lo cual nos garantiza seleccionar el canal B, mantener encendido el conversor análogo digital con ganancia 1x.

A la salida de cada canal con la ayuda de un amplificador operacional se amplifica el voltaje para tener una salida de 0 a 10 voltios DC y con la ayuda de jumpers se selecciona salida de corriente de 0 a 20mA.

3.4.8. Modulo de control Master de Procesos

Este módulo cuenta con 4 entradas digitales de 24 voltios opto acopladas, 4 salidas a transistor, y mediante el protocolo I2C y SPI establece comunicación con integrados tales como el MCP3202, MCP4822, MCP5335 y el microcontrolador PIC16F819 para poder manejar salidas análogas, entradas análogas una salida tipo PWM para controlar un servomotor.

Además este módulo va a ejercer control sobre un motor de corriente continua el cual puede variar su velocidad y giro, también controla un motor de pasos, también cuenta con contadores que serán utilizados como encoder de los motores DC y de pasos.

Realiza también control de iluminación de una luminaria tipo dicroico, y se acoplará una tarjeta en la cual existirán circuitos integrados con comunicación SPI en los cuales se conectan los sensores de temperatura teles como RTD, termocupla y una celda de carga.

Para tener un control y monitoreo de las variables que en este módulo intervienen, se lo realiza mediante el protocolo Modbus vía RS232, RS485, TCP/IP, y mediante VISA USB utilizando el microcontrolador PIC18F2550 del módulo master de comunicaciones el cual trabaja como Modbus Master y transmite los datos vía USB a Labview, utilizando el protocolo VISA.

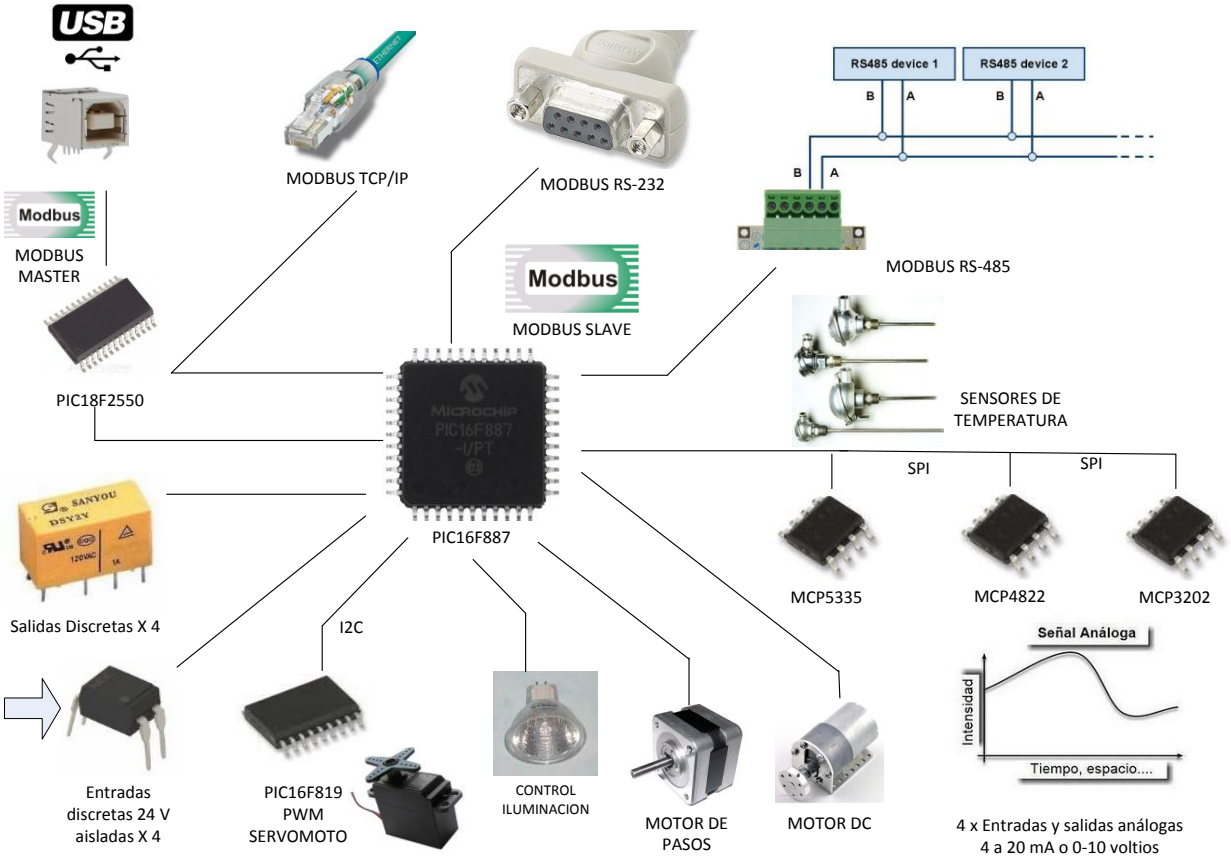


Figura 3.87. Arquitectura Módulo Master de Procesos

Este módulo se va a encargar de tareas puntuales tales como control de velocidad y giro de un motor DC y para monitorearlo se dispondrá de un contador que actuará como encoder del motor, va a controlar también un motor de pasos el cual

al igual que el motor DC tendrá un contador de pasos en modo de encoder, también actuará sobre un servomotor de posición.

A este módulo también van a ingresar las señales correspondientes a sensores de temperatura tales como RTD tipo PT100, termocupla tipo K y también tendrá una entrada para conectar una celda de carga.

También este módulo realizará el control de iluminación con un dicroico.

De manera adicional va a contar con 4 entradas discretas de 24 voltios y 4 salidas discretas de transistor.

Disponen también de 2 entradas análogas 1 de 0 a 10 voltios y otra de 0 a 20 mA y 2 entradas análogas de 0 a 10 voltios y de 0 a 20mA.

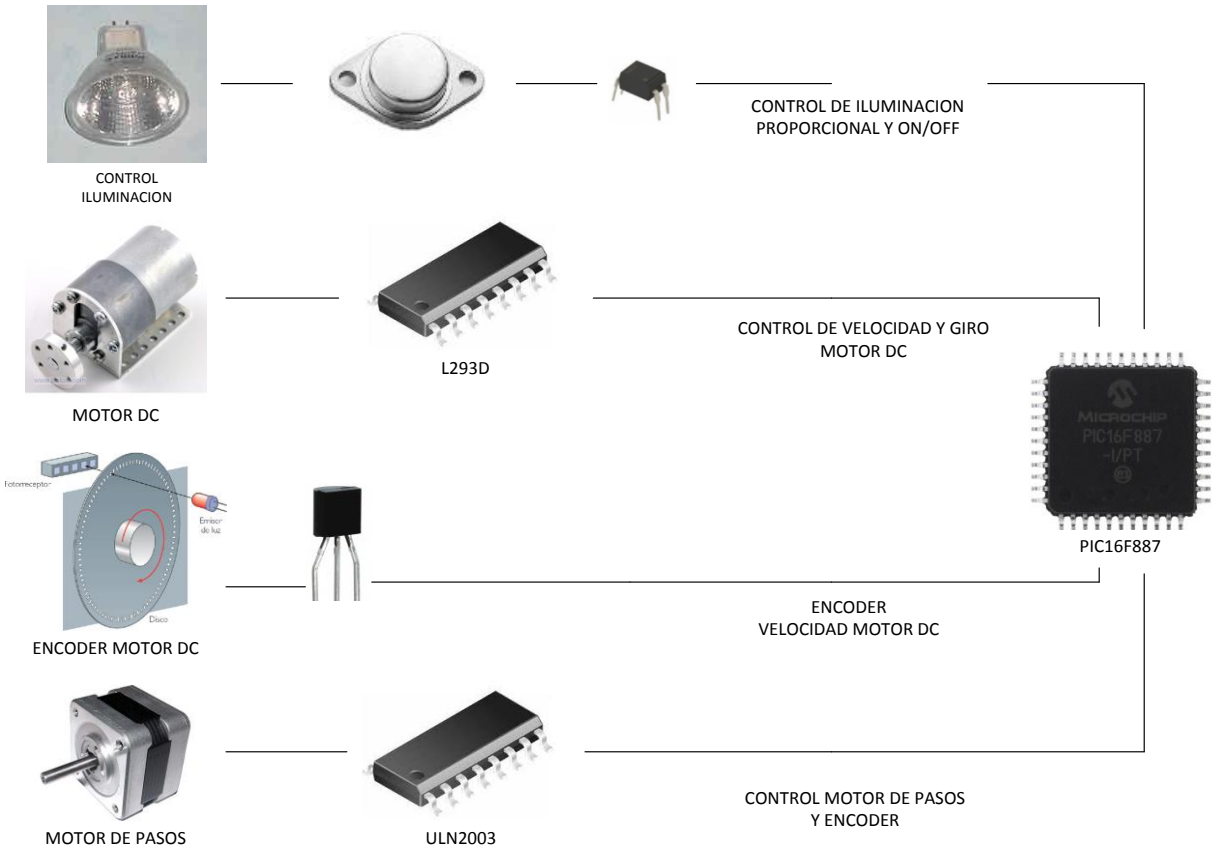


Figura 3.88. Elementos que son controlados desde el Microcontrolador Master de Procesos

A continuación se detalla el mapa de registros que será utilizado para este módulo

REGISTROS DE ESCRITURA Y LECTURA		
DIRECCIÓN	**VARIABLE**	**VALOR QUE PUEDE TENER**
40001	SALIDA ANÁLOGA 0-10V_	0-4095
40002	SALIDA ANÁLOGA 0-20mA_	0-4095
40003	SALIDAS DIGITALES A TRANSISTOR_	0-15
40004	CONTROL MOTOR DE PASOS_	0-200
40005	CONTROL MOTOR DC_	0-100
40006	CONTROL SERVOMOTOR_	0-180
40007	CONTROL DE ILUMINACIÓN_	0-100
40008	LIBRE	
40009	LIBRE	
40010	LIBRE	
REGISTROS DE LECTURA		
DIRECCIÓN	**VARIABLE**	
30001	ENTRADAS DIGITALES_	0-15
30002	DATO RTD_	0-65535
30003	DATO TERMOCUPLA	0-65535
30004	DATO CELDA DE CARGA_	0-65535
30005	ENTRADAS ANÁLOGAS 0-10V_	0-4095
30006	ENTRADAS ANÁLOGAS 0-20mA_	0-4095
30007	CONTADOR 0	0-255
30008	CONTADOR 1	0-65535
30009	TEMPERATURA	0-100
30010	LIBRE	

Tabla 3.20. Mapa de registros Modbus para el Módulo de Procesos

A continuación se va describir el código de programa para cada elemento que conforma este módulo.

3.4.8.1. Salidas análogas módulo Master de Procesos.

Al igual que las salidas análogas detalladas anteriormente, estas son realizadas con el integrado de Microchip MCP4822 el cual posee comunicación SPI y va a trabajar como esclavo del microcontrolador PIC16F887.

El proceso de escritura sobre este tipo de integrados ya se lo describió en el apartado al que se hace referencia, lo único que varía es que el canal A va a actuar como salida de voltaje de 0 a 10 voltios y el canal B será salida de corriente de 0 a 20mA.

Y se debe seguir el mapa de registros Modbus para poder escribir sobre estas salidas.

3.4.8.2. Entradas análogas módulo Master de Procesos.

Al igual que las entradas análogas detalladas anteriormente, estas son realizadas con el integrado de Microchip MCP3202 el cual posee comunicación SPI y va a trabajar como esclavo del microcontrolador PIC16F887.

El proceso de lectura de este tipo de integrados ya se lo describió en el apartado al que se hace referencia, lo único que varía es que el canal 0 (CH0) va a actuar como entrada de voltaje de 0 a 10 voltios y el canal 1 (CH1) será entrada de corriente de 0 a 20mA.

Se debe seguir el mapa de registros Modbus para poder leer la dirección correcta de estas entradas.

3.4.8.3. Entradas y Salidas discretas del Módulo Master de Procesos.

Al microcontrolador se han conectado las entradas y salidas discretas siguiendo la siguiente tabla referencial.

PUERTO A	FUNCIÓN	PUERTO D	FUNCIÓN
PIN_A0	Entrada digital 1	PIN_D0	Salida digital 1
PIN_A1	Entrada digital 2	PIN_D1	Salida digital 2
PIN_A2	Entrada digital 3	PIN_D2	Salida digital 3
PIN_A3	Entrada digital 4	PIN_D3	Salida digital 4

Tabla 3.21. Pines utilizados para Salidas discretas

Es decir tanto las entradas como salidas discretas están formadas por 4 bits.

Para leer los 4 bits menos significativos del puerto A se realizó el siguiente código de programa.

```
//Puerto A configurado como entradas
set_tris_a(0b11111111);
//guardo lo que contiene el Puerto A en la variable in_dig y //no tomo en cuenta
los bits más significativos.
byte in_dig=input_a()&15;
```

```
//invierto el estado de los pines de entrada con una función //lógica XOR, ya que por defecto el puerto esta puesto a 1 //lógico por medio de una resistencia pullup.
DIG_IN=in_dig^15;
```

El dato que se obtiene lo coloco en el registro correspondiente de registros internos Modbus para cuando requiera ser leído por transmitirlo.

Para escribir los 4 bits menos significativos del puerto D correspondiente a las salidas discretas, se procedió con el siguiente código de programa.

```
//Puerto D configurado como salidas
set_tris_d(0b00000000);
//cargo el dato correspondiente al equivalente decimal de las salidas //discretas en uno de los registros Modbus para poder indicar que fue //escrito con éxito.
out_q = modbus_rx.data[3];
//Escribo el Puerto D enmascarando la salida para respetar los bits de //dicho Puerto antes utilizados.
output_d (out_q&255);
```

3.4.8.4. Control de un motor de Pasos.

Un motor de pasos posee 4 bobinas las cuales deben ser energizadas cumpliendo una secuencia para poder hacer girar el motor.

La lógica aplicada será la siguiente.

- El dato que se enviará vía Modbus va a comprender entre 0 y 200.
- Los valores comprendidos entre 0 y 99 indicarán que el motor va a girar en sentido anti horario y el tiempo entre pasos estará dado en milisegundos de 0 a 99.
- Los valores comprendidos entre 100 y 199 indicarán que el motor va a girar en sentido horario y el tiempo entre pasos estará dado en milisegundos de 0 a 99.

Los pines del microcontrolador usados para este propósito están detallados a continuación.

PUERTO D	FUNCIÓN
PIN_D4	Bobina 1
PIN_D5	Bobina 2
PIN_D6	Bobina 3

PIN_D7	Bobina 4

Tabla 3.22. Pines utilizados para control del Motor de Pasos

El código del programa será el siguiente:

```
//Se define el byte donde se va a almacenar el dato que ingresaría vía //Modbus
Byte pasos, pasos1;
//Lazo if que realiza el giro anti horario para todo valor de pasos menor a
//100 y el valor de tiempo en milisegundos es el equivalente al dato de la
//variable pasos
If (pasos < 100)
{
   output_low (D7);
   output_high (D4);
   delay_ms(pasos);
   output_low (D4);
   output_high (D5);
   delay_ms(pasos);
   output_low (D5);
   output_high (D6);
   delay_ms(pasos);
   output_low (D6);
   output_high (D7);
   delay_ms(pasos);
}
//Lazo if que realiza el giro horario para todo valor de pasos mayor a //100 y
el valor de tiempo en milisegundos es el equivalente al dato de la //variable
pasos restada 100 y guardado en la variable pasos1.
If (pasos > 100)
{
   pasos1=pasos – 100;
   output_low (D4);
   output_high (D7);
   delay_ms(pasos1);
   output_low (D7);
   output_high (D6);
   delay_ms(pasos1);
   output_low (D6);
   output_high (D5);
   delay_ms(pasos1);
   output_low (D5);
   output_high (D4);
   delay_ms(pasos1);
}
```

Con lo cual se va a obtener el control de velocidad y sentido de giro del motor de pasos.

3.4.8.5. Control Motor de corriente continua.

El motor DC que vamos a controlar es de 12 voltios, y se le podrá variar la velocidad y el sentido de giro con la ayuda de una de las salidas de modulación de ancho de pulso PWM, del microcontrolador.

Esta salida tipo PWM va a ingresar a un Driver L293D el cual posee la circuitería necesaria para poder realizar la inversión de giro y variación velocidad del motor DC.

A continuación se indica la configuración de la salida PWM a través del modulo CCP del microcontrolador.

```
//Configuramos el Timer 2 por medio del cual se establecen las //configuraciones necesarias para el uso de los módulos CCP1 y CCP2.
//indica que está configurado como prescaler 16, el periodo es 1 y el //postcaler es 1.
SETUP_TIMER_2(T2_DIV_BY_16,1,1);
//Configuramos el modulo CCP1 como salida PWM.
SETUP_CCP1(CCP_PWM);
//Configuramos el modulo CCP2 como salida PWM.
SETUP_CCP2(CCP_PWM);
//y de la siguiente manera se escribe en el modulo PWM1 el valor que será //el correspondiente a la modulación de ancho de pulso de 0 a 255.
set_pwm1_duty(value);
```

3.4.8.6. Control de un servomotor.

El microcontrolador PIC16F887 posee dos canales de salida tipo PWM, por lo que para poder controlar el servomotor, el cual también trabaja con una señal de modulación de ancho de pulso fue necesario implementar un microcontrolador PIC16F819, el cual está configurado como esclavo I2C y será el encargado de recibir el dato correspondiente a la modulación de ancho de pulso vía I2C desde el Master y este le enviará esta señal al servomotor para que pueda realizar el control de posición.

Cabe señalar que un servomotor es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición, está conformado por un motor, una caja reductora y un circuito de control.

La corriente que requiere depende del tamaño del servo. La corriente depende principalmente del torque, y puede exceder un amperio si el servo está enclavado, pero no es muy alto si el servo está libre moviéndose todo el tiempo.

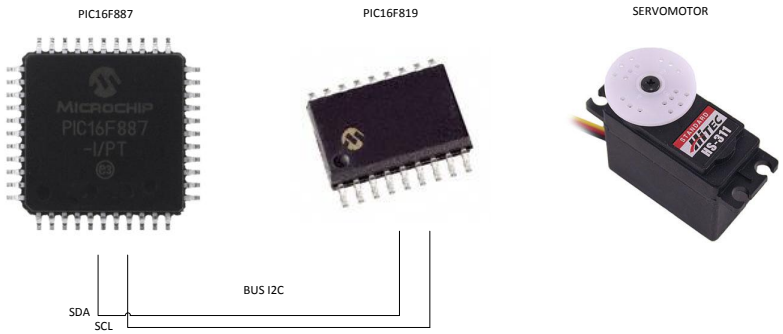


Figura 3.89. Diagrama básico de conexión entre PIC16F819 y PIC16F887 para control de un servomotor

El código de programa es similar al de escritura en las salidas discretas del Módulo Master de comunicaciones, es decir mediante comandos I2C envía el dato equivalente a la modulación de ancho de pulso, el microcontrolador esclavo lo recibe, lo almacena y finalmente lo aplica a la salida del módulo CCP1 configurado como salida PWM.

3.4.8.7. Control de iluminación.

El Modulo CCP2 se lo va a configurar como salida PWM para poder realizar el control de iluminación, esta señal de modulación de ancho de pulso va a controlar la iluminación de un dicroico de 50 watios de energía Alterna, por trabajar con voltaje AC, es necesario aislar las parte de control de la parte de Potencia, por lo que se usa un optoacoplador PC817, y a la salida de este se va a colocar un transistor de potencia el cual realizará el trabajo de elemento actuador sobre el dicroico

A continuación se indica la configuración de la salida PWM a través del módulo CCP del microcontrolador.

```
//Configuramos el Timer 2 por medio del cual se establecen las //configuraciones
necesarias para el uso de los módulos CCP1 y CCP2.
//indica que está configurado como prescaler 16, el periodo es 1 y el
//postcaler es 1.
SETUP_TIMER_2(T2_DIV_BY_16,1,1);
//Configuramos el modulo CCP1 como salida PWM.
```

```
SETUP_CCP1(CCP_PWM);
//Configuramos el modulo CCP2 como salida PWM.
SETUP_CCP2(CCP_PWM);
//y de la siguiente manera se escribe en el módulo PWM2 el valor que será //el
correspondiente a la modulación de ancho de pulso de 0 a 255.
set_pwm2_duty(value1);
```

3.4.8.8. Sensor de temperatura RTD tipo PT-100.

Para obtener el dato correspondiente a la lectura de temperatura con un sensor RTD se utilizó un circuito integrado llamado MCP3551, el cual es un convertidor análogo digital de tipo Delta-Sigma de 22 bits de resolución.

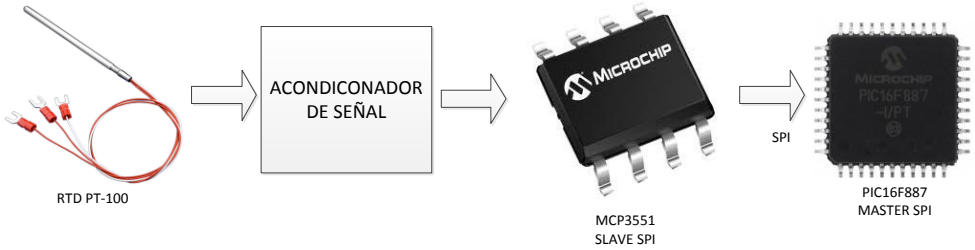


Figura 3.90. Diagrama básico de conexión entre MCP3551 y PIC16F887

Comunicación SPI con el MCP3551

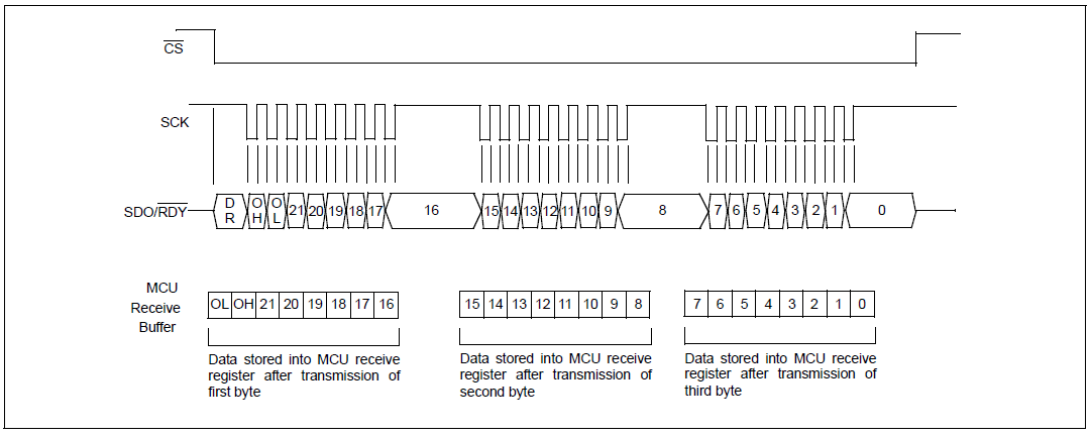


Figura 3.91. Comunicación SPI con MCP3551

Para poder leer el MCP3551 es necesario que el pin de habilitación CS pase de estado 1L a 0L y una vez que se ha terminado la transmisión el pin CS deberá volver a 1L. Mientras esto ocurre, el Pin de Clock empieza a oscilar para poder sincronizar la transmisión de datos desde el Maestro hacia el Esclavo SPI.

El dato se escribe tomando en cuenta los siguientes aspectos.

TRAMA QUE RECIBE EL MASTER

OL	OH	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabla 3.23. Trama para leer el MCP3551

Una vez que se ha establecido la comunicación con el MCP3551, este envía una trama de datos dividida en 3 bytes en la cual está codificado el valor equivalente al dato análogo que ingresa en el circuito integrado.

Esta codificación la detallamos a continuación.

- El bit 23 OL (Overflow Low), se pone en 1 lógico cuando en la entrada se ha detectado un voltaje negativo o menor al voltaje de referencia negativa.
- El bit 22 OH (Overflow High), se pone en 1 lógico cuando en la entrada se ha detectado un voltaje muy superior al voltaje de referencia positivo.
- No existe una condición en la cual se pongan a 1 ambos bits OL y OH, y la única razón por la que puedan ponerse en 1 ambos es por una falla en la comunicación SPI.
- Los 22 bits restantes corresponden al dato que ingresa en la entrada análoga.

A continuación se detalla el código de programa para la lectura del dato que proviene del circuito acondicionador del RTD.

La subrutina utilizada será la siguiente:

```
//La subrutina se llama read_SPI_RTD
void read_SPI_RTD()
{
//El PIN_B1 es el selector del integrado se pone en 0 para iniciar la
//recepción de datos
  output_low(PIN_B1);
//Recibe los 3 datos que corresponden a la lectura de los 24 bits
  datoRTD1 = spi_xfer(DA_spi,0);
  datoRTD2 = spi_xfer(DA_spi,0);
  datoRTD3 = spi_xfer(DA_spi,0);
  output_high(PIN_B1);
}
```

Debido a que los datos que se va a mostrar son de 16 bits en los registros Modbus, se ha decidido eliminar los más significativos. Y solo trabajar con los 2 bytes menos significativos y el byte más significativo utilizarlo para verificar el signo de la señal que ingresa.

3.4.8.9. Sensor de Peso, tipo celda de carga.

Para obtener el dato correspondiente a la lectura de peso con un sensor tipo celda de carga se utilizó un circuito integrado llamado MCP3551, el cual es un convertidor análogo digital de tipo Delta-Sigma de 22 bits de resolución.

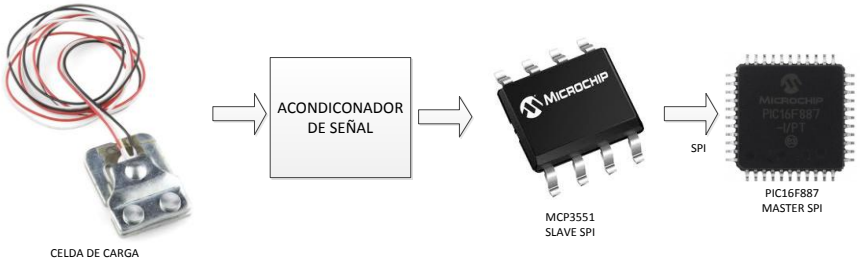


Figura 3.92. Diagrama básico de conexión entre MCP3551 y PIC16F887

Comunicación SPI con el MCP3551

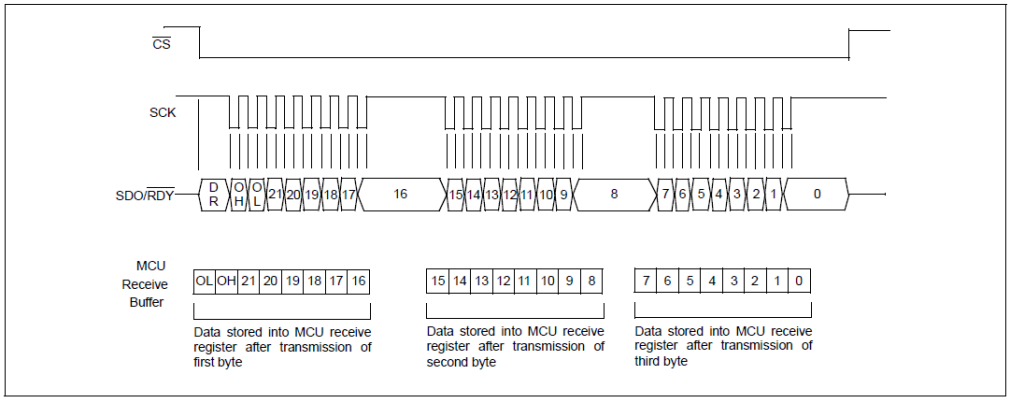


Figura 3.93. Comunicación SPI con MCP3551

Para poder leer el MCP3551 es necesario que el pin de habilitación CS pase de estado 1L a 0L y una vez que se ha terminado la transmisión el pin CS deberá volver a 1L. Mientras esto ocurre, el Pin de Clock empieza a oscilar para poder sincronizar la transmisión de datos desde el Maestro hacia el Esclavo SPI.

El dato se escribe tomando en cuenta los siguientes aspectos.

TRAMA QUE RECIBE EL MASTER

OL	OH	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabla 3.24. Trama para leer el MCP3551

Una vez que se ha establecido la comunicación con el MCP3551, este envía una trama de datos dividida en 3 bytes en la cual está codificado el valor equivalente al dato análogo que ingresa en el circuito integrado.
Esta codificación la detallamos a continuación.

- El bit 23 OL (Overflow Low), se pone en 1 lógico cuando en la entrada se ha detectado un voltaje negativo o menor al voltaje de referencia negativa.
- El bit 22 OH (Overflow High), se pone en 1 lógico cuando en la entrada se ha detectado un voltaje muy superior al voltaje de referencia positivo.
- No existe una condición en la cual se pongan a 1 ambos bits OL y OH, y la única razón por la que puedan ponerse en 1 ambos es por una falla en la comunicación SPI.
- Los 22 bits restantes corresponden al dato que ingresa en la entrada análoga.

A continuación se detalla el código de programa para la lectura del dato que proviene del circuito acondicionador de la celda de carga, la subrutina utilizada será la siguiente:

```
//La subrutina se llama read_SPI_LC
void read_SPI_LC()
{
//El PIN_B2 es el selector del integrado se pone en 0 para iniciar la
//recepción de datos
  output_low(PIN_B2);
//Recibe los 3 datos que corresponden a la lectura de los 24 bits
  datoRTD1 = spi_xfer(DA_spi,0);
  datoRTD2 = spi_xfer(DA_spi,0);
  datoRTD3 = spi_xfer(DA_spi,0);
  output_high(PIN_B2);
}
```

Debido a que los datos que se va a mostrar son de 16 bits en los registros Modbus, se ha decidido eliminar los más significativos. Y solo trabajar con los 2 bytes menos significativos y el byte más significativo utilizarlo para verificar el signo de la señal que ingresa.

3.4.8.10. Sensor de Temperatura Termocupla tipo K.

Para obtener el dato correspondiente a la lectura de temperatura con una Termocupla se utilizó un circuito integrado llamado MCP3421, el cual es un

convertidor análogo digital de tipo Delta-Sigma de 18 bits de resolución y cuenta con comunicación I2C.

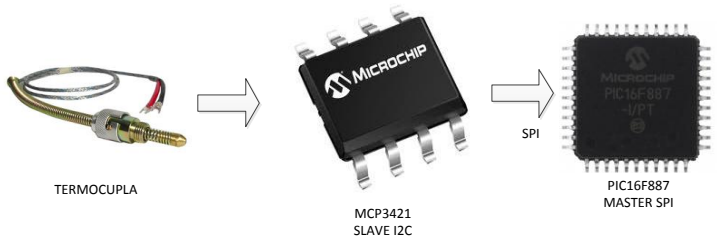


Figura 3.94. Diagrama básico de conexión entre MCP3421 y PIC16F887

Comunicación SPI con el MCP3421

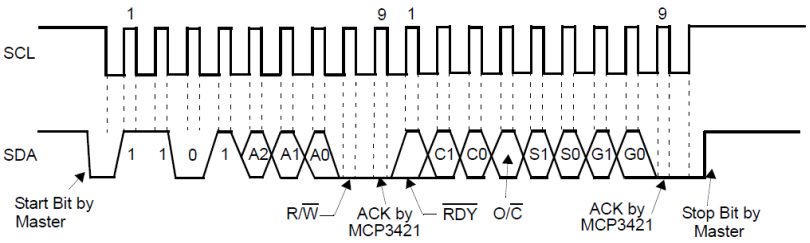


Figura 3.95. Comunicación SPI con MCP3421

Para poder leer el MCP3421 es necesario primero configurarlo, para esto, dispone de un registro de configuración el cual será detallado a continuación.

RDY	C1	C0	O/C	S1	S0	G1	G0

Tabla 3.25. Registro de configuración MCP3421

El registro de configuración es un byte el cual tiene el siguiente información:

- BIT RDY, es utilizado para indicar que una conversión fue concluida pero solamente en el caso que el MCP3421 funcione en modo One Shot, para nuestro caso la conversión será continua por lo que no nos va a interesar el estado de este BIT de configuración.
- C1 y C0, son bits para selección del canal de lectura, los cuales no vamos a utilizar porque el MCP3421 posee un solo canal de entrada.
- BIT O/C, Por defecto se lee 1 lógico, y sirve para seleccionar modo de conversión Continua si es 1 o One Shot si es 0.
- S0 y S1, selecciona la resolución de bits que va a utilizar la conversión.
  00 = (12 bits),

01 = (14 bits),

10 = (16 bits),

11 = (18 bits)

- G1 y G0, selecciona la ganancia que va a tener la señal de entrada.
  00 = 1 V/V,
  01 = 2 V/V,
  10 = 4 V/V,
  11 = 8 V/V

Para nuestro propósito vamos a utilizar conversión continua, con ganancia 2, 12 bits de resolución.

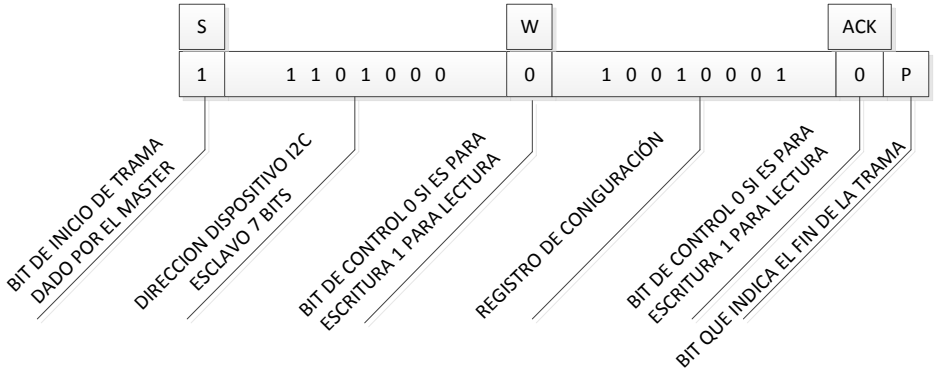


Figura 3.96. Trama de configuración del MCP3421

Una vez que se configurado el MCP3421 para poder adquirir el equivalente digital de la señal análoga que ingresa al convertidor, es necesario leer estos datos.
El dato que va a leer el Microcontrolador Master I2C va a estar dividido en 3 bytes, por lo que la resolución es de 12 bits.

La trama que recibe el microcontrolador es la siguiente:

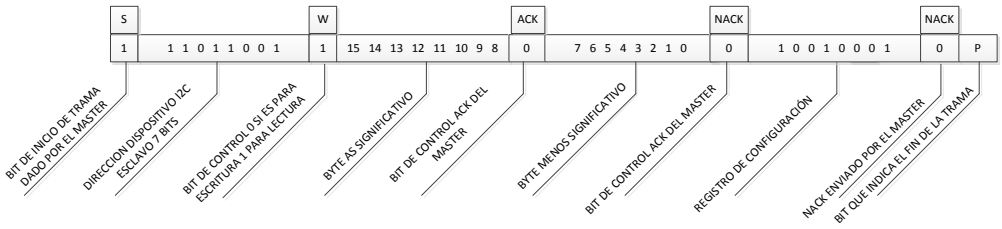


Figura 3.97. Trama de lectura del MCP3421

El MCP3421 envía 2 bytes que corresponden al dato del convertidor análogo digital, y el tercer dato es el byte de configuración, el cual es leído el cual también es enviado para confirmar que está configurado según la necesidad.

A continuación se detalla el código de programa para la lectura del dato que proviene del MCP3421.

La Subrutina para la escritura será la siguiente:

```
//La subrutina se llama write_ext_ADC
void write_ext_ADC(long int address, BYTE data)
{
//Se inicia la transmisión
   i2c_start();
//(dirección 68h + 0 para escritura)
   i2c_write(208);
//dato de configuración de registro
   i2c_write(145);
//Se termina la transmisión
   i2c_stop();
}
```

La subrutina utilizada para la lectura será la siguiente:

```
//La subrutina se llama read_ext_ADC
BYTE read_ext_ADC(long int address) {
//Inicio la transmisión
   i2c_start();
//(dirección 68h + 1 para lectura)
   i2c_write(209);
//leemos 3 bytes los cuales corresponden al dato del convertidor análogo
//digital, y el tercer dato es el registro de configuración del MCP3421.
   dato1_I2C=i2c_read(0);
   dato2_I2C=i2c_read(0);
   dato3_I2C=i2c_read(0);
//Finalización de la transmisión.
   i2c_stop();
}
```

Ya que se obtiene 16 bits del dato equivalente, se van a utilizar solo 12 para determinar el dato decimal con un valor máximo de 4095, por esta razón de los 4 bits más significativos, vamos a usar los 2 más altos para determinar los signos del valor que ingresa, de esta manera:

- El bit 15 OL (Overflow Low), se pone en 1 lógico cuando en la entrada se ha detectado un voltaje negativo o menor al voltaje de referencia negativa.
- El bit 14 OH (Overflow High), se pone en 1 lógico cuando en la entrada se ha detectado un voltaje muy superior al voltaje de referencia positivo.

3.4.8.11. Encoder Motor de pasos

Para este propósito se utilizó el Timer 0 del microcontrolador PIC16F887, el cual tiene como fuente de incremento del contador el Pin RA4.

Reseña del Timer 0

El Timer 0 es un temporizador/contador ascendente de 8 bits, cuando trabaja con el reloj del PIC se le suele llama temporizador y cundo los pulsos los recibe de una fuente externa a través de la patilla RA4/TOCKI se le llama contador.

Para nuestro caso el TIMER 0 está configurado como contador y se lo hace con la siguiente línea de programa

```
setup_timer_0(RTCC_EXT_L_TO_H|RTCC_DIV_1);
```

Con lo cual configuramos el Timer 0 como contador, que va a incrementar cada cuenta en cada flanco ascendente, y por cada pulso que reciba la cuenta será incrementada en 1.

Existen dos funciones básicas dentro del código del programa las cuales serán de mucha utilidad para leer y setear el valor de Timer 0.

Con esta función seteamos el Timer 0 con el valor ubicado dentro del paréntesis.

```
set_timer0(0);
```

Con esta función guardamos el dato del timer 0 en la variable llamada CONTADOR0, la cual posteriormente será enviada vía Modbus cada vez que el Master Modbus lo solicite.

```
CONTADOR0=GET_TIMER0();
```

3.4.8.12. Encoder Motor de Corriente Continua

Para este propósito se utilizó el Timer 1 del microcontrolador PIC16F887, el cual tiene como fuente de incremento del contador el Pin C0.

Reseña del Timer 1

El Timer1 es un temporizador/contador ascendente parecido al TMR0, pero con algunas peculiaridades que lo hacen muy interesante a la hora de incluir temporizaciones en nuestros programas.

La primera de ellas, es que se trata de un contador de 16 bits cuyo valor se almacena en dos registros de 8 bits el TMR1H y el TMR1L, ambos registros se pueden leer y escribir su valor durante la ejecución del programa.

Cuando el Timer1 está habilitado, el valor de esos registros se incrementan desde 0000h a FFFFh y una vez que llega a su máximo valor empieza otra vez desde 0 avisándonos de ello por medio de la bandera TMR1F.

Si está activa la interrupción por desbordamiento del Timer 1 al desbordarse el contador, el programa entra en la función de tratamiento a la interrupción por desbordamiento del Timer1.

Para nuestro caso el TIMER 1 está configurado como contador y se lo hace con la siguiente línea de programa

```
setup_timer_1(T1_EXTERNAL_SYNC|T1_DIV_BY_1
```

Con lo cual configuramos el Timer 1 como contador sincronizado a la entrada externa del reloj, y su prescaler es de 1

Existen dos funciones básicas dentro del código del programa las cuales serán de mucha utilidad para leer y setear el valor de Timer 1.

Con esta función seteamos el Timer 1 con el valor ubicado dentro del paréntesis, este valor es de 16 bits

```
set_timer1(0);
```

Con esta función guardamos el dato del timer 1 en la variable llamada CONTADOR1, la cual posteriormente será enviada vía Modbus cada vez que el Master Modbus lo solicite, este dato es de 16 bits

```
CONTADOR1=GET_TIMER1();
```

CAPITULO IV

PROTOCOLO DE PRUEBAS

Introducción

Este capítulo ha sido dedicado al protocolo de pruebas a todo el módulo de adquisición de datos en cada una de sus tarjetas como son: Fuente de alimentación y circuitos de procesos, Master de comunicaciones. Master de procesos, Entradas digitales, Entradas análogas, Salidas digitales, Salidas análogas.

Estas pruebas se las realizara utilizando señales estándar para las que han sido diseñadas cada una de las tarjetas como 24 VDC, 0 – 10 VDC, 0 – 20 mA, etc. Y también se realizaran las pruebas de comunicaciones respectivas de la tarjeta master de comunicaciones con estándares de comunicaciones y protocolos como: Modbus RTU sobre RS – 232, Modbus RTU sobre RS – 485, USB, Modbus RTU sobre TCP/IP.

Se han adjuntado las fotografías de las pruebas de funcionamiento realizadas al módulo en el Anexo VI.

En el presente capítulo también se detallarán los precios para la construcción de cada una de las tarjetas y al final se adjunta una reseña con el análisis de precios de la mano de obra.

4.1. Pruebas de funcionamiento de la tarjeta de Fuente de alimentación y circuitos de procesos.

En esta tarjeta se encuentran montadas las fuentes de alimentación y algunos de los circuitos integrados que se encargaran del control de motores y Dimmer.

Figura 4.1. Tarjeta fuente de alimentación y procesos.

4.1.1. Pruebas de fuente de alimentación.

Se realizaran las pruebas de funcionamiento de los diferentes niveles de tensión de los que consta el módulo. Estos niveles de tensión serán utilizados para consumo propio del módulo y para alimentación de dispositivos externos hasta 5A.

PRUEBAS DE LA FUENTE DE ALIMENTACIÓN

N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Fuente de alimentación de 5 VDC.	Obtener 5 VDC en las borneras correspondientes a la salida de dicho nivel de tensión, se realizaran medidas con un multímetro.	X	
2	Fuente de alimentación de 12 VDC.	Obtener 12 VDC en las borneras correspondientes a la salida de dicho nivel de tensión, se realizaran medidas con un multímetro.	X	
3	Fuente de alimentación de 24 VDC	Obtener 24 VDC en las borneras correspondientes a la salida de dicho nivel de tensión, se realizaran medidas con un multímetro.	X	

Tabla 4.1. Tabla de pruebas de la fuente de alimentación.

4.1.2. Pruebas de circuitos de procesos.

En esta etapa se van a realizar pruebas de funcionamiento de los circuitos de control de los siguientes circuitos:

- Control de luminosidad de una luminario (dicroico) de 50 W a 120 VAC.
- Control de un motor DC de 12 VDC con su respectivo encoder.
- Control de un motor de pasos (PAP) de 4 polos a 5 VDC.

PRUEBAS DE LOS CIRCUITOS DE PROCESOS				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Pruebas de funcionamiento del Dimmer.	Variar la intensidad luminosa de un foco de tipo dicroico de 50W, esto se realizara median un troceado de la señal alterna entrante mediante el PWM emitido por uno de los microcontroladores. Se ha comprobado el troceado de la señal con un osciloscopio.	X	
2	Pruebas de funcionamiento del circuito de control del motor DC.	Para el control del motor DC no hemos ayudado del CI L293D, ingresando 2 señales de tipo PWM para lograr el incremento y decremento de velocidad y la inversión de giro del motor. Los valores de PWM serán escritos a través de comunicación Modbus o USB.	X	

3	Pruebas de funcionamiento del circuito de control del motor de pasos (PAP) y contador de pasos.	Se controlará una un motor de pasos de 5 VDC mediante un CI ULN2003, este deberá girar 360°, este valor deberá ser cargado a través de comunicación Modbus o USB. Este tendrá un lazo que verificara el si se han cumplido los pasos realizados por el motor PAP.	X	

Tabla 4.2. Tabla de pruebas de los circuitos de procesos

4.1.3. Resumen costos Fuente de alimentación y circuitos de fuerza

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	60	1	30
Borneras	Borneras 5mm macho y hembra	1	10	10
Capacitores	Cerámicos SMD	0,28	4	1,12
Capacitores	Electrolíticos	0,07	11	0,77
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
Puente de diodos	Puente de diodos 8 amp. 200 voltios	1,5	3	4,5
Diodos	Diodo Rectificador 1N4007	0,07	4	0,28
Diodo Led	Leds verde 3mm	0,10	1	0,10
Opto acopladores	PC817	0,35	3	1,05
Potenciómetros	Potenciómetro de precisión 10k Ohm	0,8	1	0,8
Transistor	2N3904	0,1	2	0,2
Transistor	TIP122	1,25	1	1,25
Resistencias	Resistencias SMD1206	0,03	16	0,48
Disipador	Disipador de calor tipo TO03	2	1	2
Driver	L293DD SMD	2,88	1	2,88
Regulador	LM350K ajustable TO-03	7,92	1	7,92
Regulador	LM7805 TO220	0,55	1	0,55
Regulador	LM7812 TO220	0,55	1	0,55
ULN2003	Arreglo darlington	0,34	1	0,34
Transformador	120/24VAC 2amperios	6	1	6
Ventiladores	Ventiladores 12 voltios	1	2	2
TOTAL				83,44

Tabla 4.3. Costos de la tarjeta de la fuente de alimentación y circuitos de fuerza.

4.2. Pruebas de funcionamiento de la tarjeta master de comunicaciones.

La tarjeta trae montados en si dispositivos transeivers para comunicar el módulo con diferentes dispositivos externos, las comunicaciones que encontraremos en la tarjeta son: RS – 232, RS – 485, USB y ETHRNET.

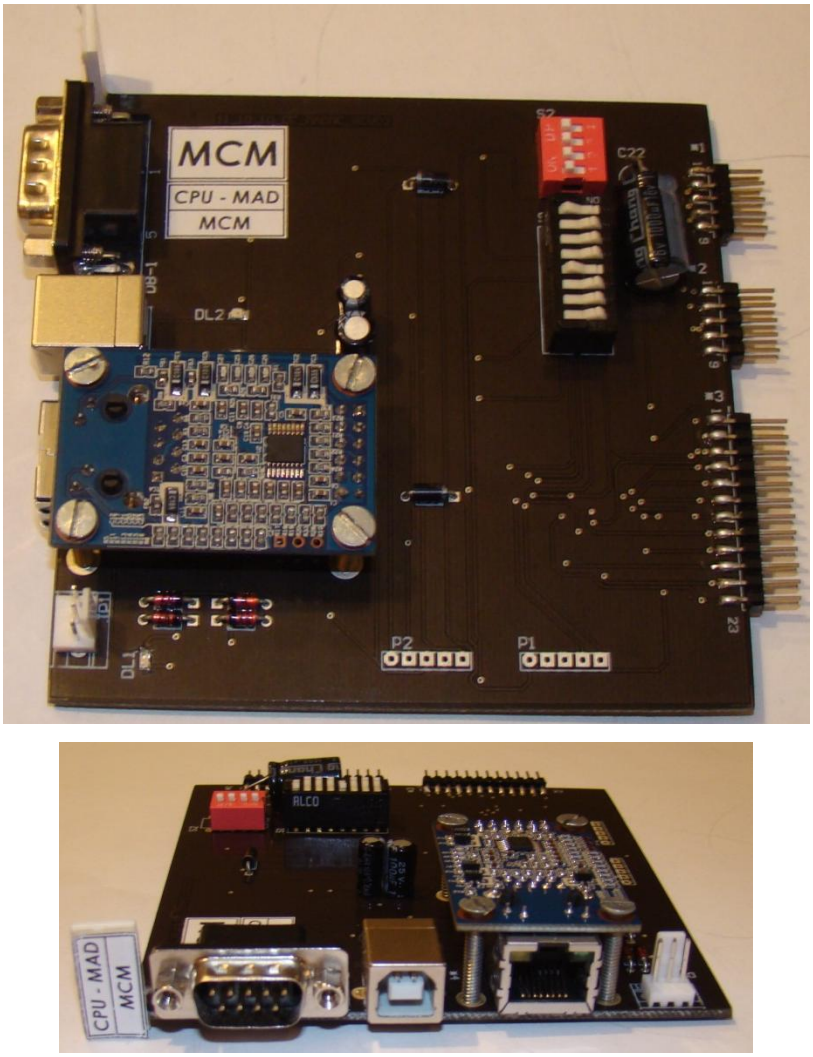


Figura 4.2. Tarjeta master de comunicaciones.

4.2.1. Pruebas de comunicación Modbus sobre RS232.

Se realizaron pruebas de comunicación sobre el estándar RS – 232 los cuales se realizaron con una conexión punto a punto utilizando protocolo Modbus que se encuentra cargado en el microprocesador.

COMUNICACIÓN RS - 232.				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Conexión mediante cable cruzado Modulo – Computador	Conectar la tarjeta a un ordenador desde su puerto RS - 232 mediante un cable cruzado con terminales DB9 hembra - hembra.	X	
2	Pruebas de comunicación Modbus sobre RS – 232	Asignar una dirección de esclavo a la tarjeta mediante los dip - swith antes descritos y comunicar con el gestor OPC server de National Instruments.	X	
3	Pruebas de comunicación Modbus punto a punto con un PLC Allen Bradley RSLogix 1100.	Asignar una dirección de esclavo a la tarjeta mediante los dip – switch antes descritos y comunicar con el PLC master realizando pruebas de lectura/escritura de estados del módulo.	X	
4	Monitoreo de la trama de protocolo Modbus.	Conectar el modulo al ordenador asignando una dirección de esclavo y monitorear la tramas en las diferentes funciones utilizadas para la lectura/escritura para esto nos ayudaremos del software Modbus Tools y del servidor OPC Client de Natonal Instruments.	X	

Tabla 4.4. Tabla de pruebas de la tarjeta master de comunicaciones RS - 232.

4.2.2. Pruebas de comunicación Modbus sobre RS485.

Estas pruebas se las realizaron ayudándonos del mismo PLC Allen Bradley RSLogix 1100, un módulo de conversión de estándares de RS – 232 a RS - 485 AIC+ (Advanced Interface Converter), un modulo de adquisición de datos de la

marca Red Lion serie DLC (Dual Loop Controler) y tarjetas adicionales fabricadas con propósito de pruebas.

Se construyó una Red RS485 en la cual estuvieron involucrados los equipos antes mencionados y por medio de la cual se pudo comprobar que tanto la parte de Hardware y Software trabajan de manera adecuada en cuanto tiene que ver al módulo de adquisición de datos ya que recibe y envía tramas de datos vía Modbus sin ningún inconveniente.

COMUNICACIÓN RS - 485				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Conectar e red el módulo de adquisición de datos con el PLC Allen Bradley RSLogix 1100	Conectar a través de protocolo Modbus sobre RS - 485 hacia el PLC usando diferentes direcciones y funciones del protocolo Modbus. Las conexiones se las realizaron punto a punto con el cable que corresponde al plc.	X	
2	Ingresar a la red más dispositivos con diferentes direcciones.	Montar en la red más dispositivos (un dispositivo master y 3 esclavos) que tengan comunicación Modbus RS - 485 para probar el correcto direccionamiento de los datos.	X	
3	Prueba de pequeño sistema SCADA de la red.	Monitorear el estado de los diferentes dispositivos subiéndolos al PC mediante el OPC Server proporcionado por National Instruments.	X	

Tabla 4.5. Tabla de pruebas de la tarjeta master de comunicaciones RS - 485.

4.2.3. Pruebas de comunicación USB.

Si realizaran pruebas de conectividad USB median el paquete Visa – USB de National Instruments, se generaran lo respectivos drivers que se han descrito anteriormente para que el computador pueda reconocer el dispositivo como un dispositivo USB – RAW (Read and Write) y adquirir los datos igualmente antes descritos.

COMUNICACIÓN USB - RAW

N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Crear driver para reconocimiento automático del módulo.	Generar el driver correspondiente al VID y PID del módulo y verificar la detección automática del mismo en el computador en el paquete Measurement & Automation de National Instruments.	X	
2	Pruebas de control y adquisición de datos del módulo.	Adquirir los datos del módulo mediante USB con las herramientas de LabVIEW, específicamente los bloques que Instruments I/O Assistant.	X	

Tabla 4.6. Tabla de pruebas de la tarjeta master de comunicaciones USB- RAW.

4.2.4. Pruebas de comunicación Ethernet.

Se alimentará el módulo y se procederá al direccionamiento IP mediante el software provisto por los proveedores del módulo conversor Ethernet / RS – 485 WIZ107SR luego de lo cual se procederá a las pruebas de comunicaciones a través de protocolo Modbus sobre TCP/IP.

COMUNICACIÓN ETHERNET				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Cargar dirección IP sobre el modulo conversor WIZ107SR.	Conectar a la red el módulo y hacer que el software provisto por la marca detecte automáticamente le dispositivo para así tener acceso a la configuración de direcciones IP, Mascara de Subred, Gateway, etc.	X	
2	Configuración de la red en el computador.	Se configurara los parámetros de la red que trae el computador ingresando en las configuraciones de red del mismo	X	
3	Pruebas de comunicaciones.	Se realizarán las pruebas de comunicaciones del módulo hacia el PC valiéndonos de herramientas tales como el Modbus Poll, OPC server de National Instruments.	X	
4	Prueba de pequeño sistema SCADA de la red.	Monitorear el estado de los diferentes dispositivos subiéndolos al PC mediante el OPC Server proporcionado por National Instruments. Esto se lo realizara enviando datos a los diferentes registros asignados para cada uno de los periféricos del módulo.	X	

Tabla 4.7. Tabla de pruebas de la tarjeta master de comunicaciones Ethernet.

4.2.5. Resumen costos Tarjeta Master de Comunicaciones.

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
Microcontrolador	PIC16F887	7	1	7
Microcontrolador	PIC18F2550	8,7	1	8,7
WIZ107SR	Gateway Ethernet – RS-232	45	1	45
Conector USB	Conector USB Tipo B Para placa	0,7	1	0,7
Conector DB-9	Conector DB-9 Macho para placa	0,7	1	0,7
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Capacitores	Cerámicos SMD	0,28	21	5,88
Capacitores	Electrolíticos	0,07	3	0,21
Diodos	Diodo 1N4148	0,09	6	0,54
Diodo Led	Leds SMD	0,71	3	2,13
Resistencias	Resistencias SMD1206	0,03	25	0,75
Dip Switch	10 pines	0,4	1	0,4
Dip Switch	16 pines	0,5	1	0,5
MAX232	Circuito integrado MAX232	2,05	2	4,1
MAX485	Circuito integrado MAX485	1,33	1	1,33
LM1117	Regulador 3,3 voltios SMD	0,62	1	0,62
Cristal	Cristal de 20 Mhz SMD	0,60	2	1,2
TOTAL				120,56

Tabla 4.8. Costos de la tarjeta master de comunicaciones.

4.3. Pruebas de funcionamiento de la tarjeta de entradas digitales.

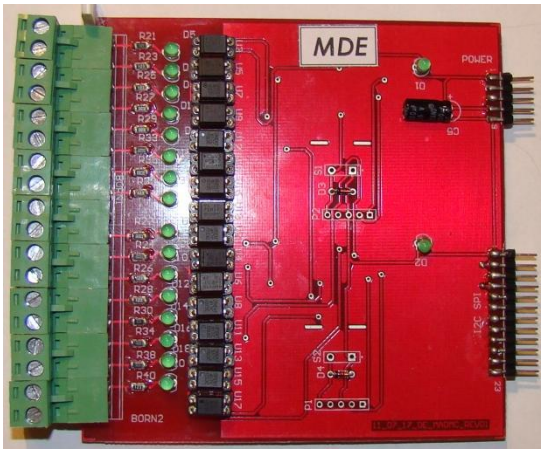


Figura 4.3. Tarjeta de entradas digitales.

En el presente capítulo se realizaran las pruebas de funcionamiento de las entradas digitales sometiéndolas a diferentes niveles de tensión entre 12 y 24 VDC, cabe recalcar que las entradas fueron diseñadas para que funcionen de preferencia a 24 VDC.

ENTRADAS DIGITALES				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Prueba visual del accionamiento de las entradas digitales.	Se aplicarán diferentes niveles de tensión sobre las entradas digitales y se observara que los led indican que se han accionado.	X	
2	Probar el funcionamiento de todas y cada una de las entradas digitales mediante software.	Se realizara una pequeña aplicación en el software LabVIEW valiéndonos de las comunicaciones antes descritas para al adquisición de datos de las entradas digitales. Se aplicara en todas las entradas un voltaje de 24 VDC.	X	
3	Variar niveles de Tensión en las entradas digitales	Variar los niveles de tensión que se están aplicando en las entradas digitales y comprobar niveles máximos y mínimos de accionamiento de las entradas	X	

Tabla 4.9. Tabla de pruebas de la tarjeta de entradas digitales.

4.3.1. Resumen costos tarjeta de Entradas Digitales 24VDC

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
Microcontrolador	PIC16F876A	5	2	10
Borneras	Borneras 5mm macho y hembra	1	18	18
Capacitores	Cerámicos SMD	0,28	7	1,96
Capacitores	Electrolíticos	0,07	1	0,07
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
Diodo Led	Leds verde 3mm	0,10	18	1,8
Diodos	Diodo 1N4148	0,09	2	0,18
Resistencias	Resistencias SMD1206	0,03	40	1,2
Opto acopladores	PC817	0,35	16	5,6
Cristal	Cristal de 20 Mhz SMD	0,60	2	1,2
TOTAL				71,21

Tabla 4.10. Costos de la tarjeta de entradas digitales.

4.4. Pruebas de funcionamiento de la tarjeta de salidas de relé.

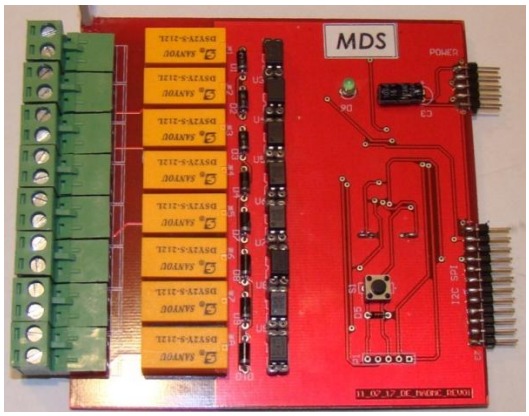


Figura 4.4. Tarjeta de salidas de relé.

Se probaran todas y cada una de las salidas de relé de las que dispone la tarjeta (8 salidas de relé) accionándolas con cargas de no mayores a 1 A.

SALIDAS DE RELÉ				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Pruebas de accionamiento de los relés mediante software.	Se accionaran todas y cada uno de los relés que corresponden a las salidas y se constatara con un multímetro midiendo continuidad en los puntos de bornera que corresponden a las salidas. Los accionamientos se los realizaran a través de una pequeña aplicación realizada en LabVIEW.	X	

Tabla 4.11. Tabla de pruebas de la tarjeta de salidas de relé.

4.4.1. Resumen costos tarjeta de Salidas a Relé

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
Microcontrolador	PIC16F876A	5	1	5
Relé	8 pines 12 voltios, 2 contactos	2	8	16
Borneras	Borneras 5mm macho y hembra	1	16	16
Capacitores	Cerámicos SMD	0,28	5	1,4
Capacitores	Electrolíticos	0,07	1	0,07
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
Diodos	Diodo Rectificador 1N4007	0,07	8	0,56
Diodos	Diodo 1N4148	0,09	1	0,09
Diodo Led	Leds verde 3mm	0,10	1	0,10
Resistencias	Resistencias SMD1206	0,03	12	0,36
Opto acopladores	PC817	0,35	8	2,8
Cristal	Cristal de 20 Mhz SMD	0,60	1	0,6
TOTAL				74,18

Tabla 4.12. Costos de la tarjeta de salidas de relé.

4.5. Pruebas de funcionamiento de la tarjeta de entradas análogas.

Figura 4.5. Tarjeta de entradas análogas.

Para realizar las pruebas de funcionamiento de la tarjeta de entradas análogas se realizó las medidas con un osciloscopio para constatar que la salida de tensión se de forma continua y un multímetro un multímetro para constatar los valores de salida con más exactitud.

ENTRADAS ANÁLOGAS				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Medición de valores del lazo de corriente.	Colocar los jumpers en las posiciones adecuadas para empezar la emisión de la señal del lazo de corriente y enviar dados desde el software LabVIEW hacia el módulo para obtener diferentes valores.	X	
2	Medición de niveles de tensión generados.	Colocar los jumpers en las posiciones adecuadas para empezar la emisión de señales de tipo voltaje y enviar dados desde el software LabVIEW hacia el módulo para obtener diferentes valores.	X	

Tabla 4.13. Tabla de pruebas de la tarjeta de entradas análogas.

4.6.5. Resumen costos Tarjeta de entradas análogas.

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
MCP3202	ADC SPI 12 bits	3,28	2	7,06
Potenciómetros	Potenciómetro de precisión 10k Ohm	0,8	4	3,2
Borneras	Borneras 5mm macho y hembra	1	12	12
Capacitores	Cerámicos SMD	0,28	6	1,68
Capacitores	Electrolíticos	0,07	1	0,07
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
AD822	Amplificador Operacional Rail to Rail	6,73	4	26,92
Resistencias	Resistencias SMD1206	0,03	31	0,93
TOTAL				83,06

Tabla 4.14. Costos de la tarjeta de entradas análogas.

4.6. Pruebas de funcionamiento de la tarjeta de salidas análogas.

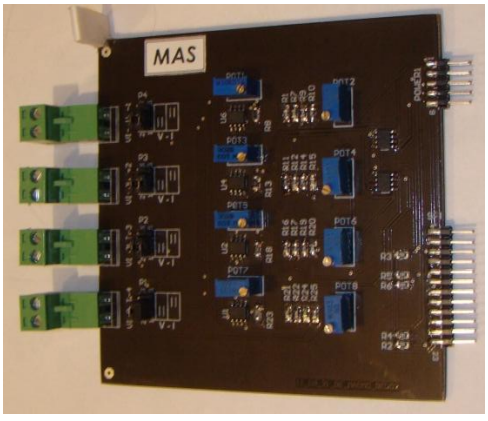


Figura 4.6. Tarjeta de salidas análogas.

Para realizar las pruebas de funcionamiento de la tarjeta de entradas análogas se debió suministrar dos tipos de señales (corriente y voltaje). Estos niveles de tensión los hemos obtenido de la tarjeta de salidas análogas ya que esta puede generar estos dos tipos de señales.

SALIDAS ANÁLOGAS				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Medición de valores del lazo de corriente en formato de 12 bit.	Colocar los jumpers en las posiciones adecuadas para realizar las respectivas mediciones del lazo de corriente y empezar la lectura a de los valores a través de una aplicación básica de LabVIEW.	X	
2	Medición de valores de niveles de tensión en formato de 12 bit.	Colocar los jumpers en las posiciones adecuadas para realizar las respectivas mediciones de niveles de tensión y empezar la lectura a de los valores a través de una aplicación básica de LabVIEW.	X	

Tabla 4.15. Tabla de pruebas de la tarjeta de salidas análogas.

4.6.1. Resumen costos Tarjeta de Salidas análogas

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
MCP4822	DAC SPI 12 bits	3,41	2	6,82
Potenciómetros	Potenciómetro de precisión 10k Ohm	0,8	8	6,4
Borneras	Borneras 5mm macho y hembra	1	8	8
Capacitores	Cerámicos SMD	0,28	6	1,68
Capacitores	Electrolíticos	0,07	1	0,07
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
AD822	Amplificador Operacional Rail to Rail	6,73	4	26,92
Resistencias	Resistencias SMD1206	0,03	35	0,75
TOTAL				81,84

Tabla 4.16. Costos de la tarjeta de salidas análogas.

4.7. Pruebas de funcionamiento de la tarjeta master de procesos.

Para las pruebas de funcionamiento de la tarjeta master de procesos se realizo una aplicación en el programa LabVIEW para poder actuar y leer cada una de las funciones que nos ofrece esta tarjeta como son:

- 4 Entradas digitales.
- 4 Salidas digitales.
- Un punto de alimentación a un servo motor el cual es controlado a través de modulación del ancho de pulso.
- Un contacto de falla.
- Una entrada análoga de 0 a 10 VDC y 0 a 20 mA.
- Una salida análoga de 0 a 10 VDC y 0 a 20 mA.
- Entrada para un sensor de temperatura de tipo Termocupla.
- Entrada para un sensor de temperatura de tipo RTD y una celda de carga.

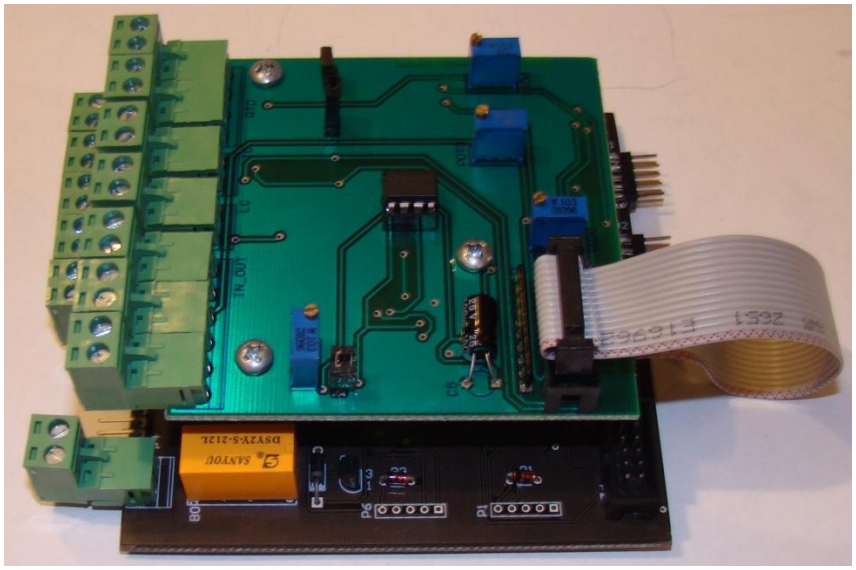


Figura 4.7. Tarjeta master de procesos.

TARJETA MASTER DE PROCESOS				
N°	DESCRIPCIÓN	REQUERIMIENTOS	CUMPLE	
			SI	NO
1	Probar el funcionamiento de las 4 entradas digitales mediante software.	Se realizara una pequeña aplicación en el software LabVIEW valiéndonos de las comunicaciones antes descritas para la adquisición de datos de las entradas digitales. Se aplicara en todas las entradas un voltaje de 24 VDC.	X	
2	Probar el funcionamiento de las 4 salidas digitales mediante software.	Se realizara una pequeña aplicación en el software LabVIEW valiéndonos de las comunicaciones antes descritas para la adquisición de datos de las entradas digitales. Se aplicara en todas las entradas un voltaje de 24 VDC.	X	
3	Prueba de servo motor.	Se enviarán datos a través de LabVIEW hacia el módulo para modificar los valores de la salida del PWM y así lograr el posicionamiento del servomotor.	X	
4	Prueba de contacto de falla.	Al momento de energizar el módulo el relé correspondiente al contacto de falla será energizado para cambiar su estado, este deberá permanecer así todo el tiempo que el módulo se encuentre operativo. Si el contacto nos indicara un estado normalmente abierto nos indicara que existe algún problema en el módulo.	X	
5	Medición de valores del lazo de corriente en formato de 12 bit.	Colocar los jumpers en las posiciones adecuadas para realizar las respectivas mediciones del lazo de corriente y empezar la lectura a de los valores a través de una aplicación básica de LabVIEW.	X	
6	Medición de valores de niveles de tensión en formato de 12 bit.	Colocar los jumpers en las posiciones adecuadas para realizar las respectivas mediciones de niveles de tensión y empezar la lectura a de los valores a través de una aplicación básica de LabVIEW.	X	
7	Medición de valores del lazo de corriente.	Colocar los jumpers en las posiciones adecuadas para empezar la emisión de la señal del lazo de corriente y enviar dados desde el software LabVIEW hacia el módulo para obtener diferentes valores.	X	

8	Medición de niveles de tensión generados.	Colocar los jumpers en las posiciones adecuadas para empezar la emisión de señales de tipo voltaje y enviar dados desde el software LabVIEW hacia el módulo para obtener diferentes valores.	X	
9	Medición de valores de temperatura con una termocupla en un formato de 16 bit.	Conectar una termocupla de (J, K, R, T) a los terminales respectivos y adquirir los datos a través de la interfaz en LabVIEW.		X
10	Medición de valores de temperatura con un sensor de tipo RTD en un formato de 16 bit.	Conectar el sensor en los terminales respectivos y empezar la lectura del dato en formato de 16 bit hacia LabVIEW.	X	
11	Medición de valores de peso con un sensor de tipo celda de carga en un formato de 16 bit.	Conectar el sensor en los terminales respectivos y empezar la lectura del dato en formato de 16 bit hacia LabVIEW.	X	

Tabla 4.17. Tabla de pruebas de la tarjeta master de procesos.

4.7.1. Resumen costos Tarjeta Master de Proceso.

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
Microcontrolador	PIC16F887	7	1	7
Microcontrolador	PIC16F819	2,4	1	2,4
Borneras	Borneras 5mm macho y hembra	1	13	13
Capacitores	Cerámicos SMD	0,28	8	2,24
Capacitores	Electrolíticos	0,07	1	0,07
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
Diodos	Diodo 1N4148	0,09	2	0,18
Diodos	Diodo Rectificador 1N4007	0,07	1	0,07
Diodo Led	Leds SMD	0,71	3	2,13
Diodo Led	Leds verde 3mm	0,10	4	0,40
Resistencias	Resistencias SMD1206	0,03	24	0,72
Opto acopladores	PC817	0,35	4	1,4
Rpack	R pack de 10 k ohmios	0,40	1	0,4
Transistor	2N3904	0,1	1	0,1
Relé	8 pines 12 voltios, 2 contactos	2	1	2
Cristal	Cristal de 20 Mhz SMD	0,60	1	0,6
TOTAL				63,91

Tabla 4.18. Costos de la tarjeta master de procesos.

4.7.2. Resumen costos Tarjeta de sensores

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Circuito Impreso	PCB para montaje de elementos	50	1	30
Borneras	Borneras 5mm macho y hembra	1	12	12
Capacitores	Cerámicos SMD	0,28	13	3,64
Capacitores	Electrolíticos	0,07	1	0,07
Header Macho	Header macho doble de 40 pines	0,8	1	0,8
Header Macho	Header macho simple de 40 pines	0,4	1	0,4
Potenciómetros	Potenciómetro de precisión 10k Ohm	0,8	4	3,2
Resistencias	Resistencias SMD1206	0,03	24	0,60
Rpack	R pack de 10 k ohmios	0,40	1	0,4
MCP9800	Sensor de temperatura I2C	1,36	1	1,36
MCP3551	ADC Delta Sigma 22 bits SPI	4,52	2	9,04
MCP1702	Regulador 3,3 Voltios Microchip	0,48	2	0,96
MCP3202	ADC SPI 12 bits	3,28	1	3,28
MCP4822	DAC SPI 12 bits	3,41	1	3,41
MCP3421	ADC Delta Sigma 18 bits I2C	2,72	1	2,72
AD822	Amplificador Operacional Rail to Rail	6,73	3	20,19
TOTAL				102,07

Tabla 4.19. Costos de la tarjeta de sensores.

4.8. Resumen costos adicionales

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Back Panel	PCB para montaje de elementos	25	1	25
Header Hembra	Header hembra doble de 40 pines	0,8	4	3,2
Caja de acrílico	Con ranuras para montar las tarjetas	30	1	30
Estaño	Rollo de estaño delgado	4	4	16
Pomada	Para soldar	2	1	2
TOTAL				76,2

Tabla 4.20. Resumen de costos adicionales.

Tomando en cuenta que son 8 tarjetas, y por tarjeta toma un aproximado de 2 horas para armar con todos los elementos, el proyecto total se armará entre 2

personas, se considera que en 8 horas estarán las tarjetas armadas y listas para ser probadas, asumiendo un valor de 5 dólares la hora de trabajo se tiene un valor total de 80 dólares la mano de obra, considerando que se armará entre 2 personas el proyecto.

4.9. Resumen de Costos

Elemento	Descripción	Precio Unitario (USD)	Cantidad	Precio Total (USD)
Módulo Máster de Comunicaciones	Tarjeta MCM gestiona todo tipo de comunicación que maneja el Módulo, CPU del equipo	120,56	1	120,56
Módulo de Entradas discretas	Tarjeta MDE, 16 entradas discretas de 24VDC en grupos de 8	71,21	1	71,21
Módulo de Salidas Discretas	Tarjeta MDS, 8 salidas a Relé	74,18	1	74,18
Módulo de Entradas Análogas	Tarjeta MAE, 4 entradas análogas de 0-20mA ó 0-10VDC	83,06	1	83,06
Módulo de Salidas Análogas	Tarjeta MAS, 4 salidas análogas de 0-20mA ó 0-10VDC	81,84	1	81,84
Módulo Master de Procesos + tarjeta para sensores	Tarjeta MPM, se encarga de realizar el control y monitoreo de una miniplanta de procesos	165,98	1	165,98
Fuente de Alimentación	Tarjeta fuente de alimentación de 5, 12 y 24 VDC + circuitos de fuerza	83,44	1	83,44
Material Adicional	Material adicional utilizado para la construcción del Módulo	76,20	1	76,20
Mano de Obra	Asumiendo que entre 2 personas arman el módulo	80	1	80
TOTAL				836,44

Tabla 4.21. Resumen Total de Costos.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Puesto que el Módulo Esclavo fue realizado con elementos de fácil adquisición en su gran mayoría, es fácil de duplicar, y podría ser usado en distintas aplicaciones tales como sistemas de adquisición de datos industriales, en edificios inteligentes como módulo de expansión de puertos para PLC o en proyectos estudiantiles.

- Módulos de similares características se encuentran en el mercado con costos sumamente elevados, por lo que este proyecto se convierte en una opción económica para desarrollo de aplicaciones de distinto ámbito ya que los diferentes elementos utilizados en la construcción del módulo son de costos reducidos y se los puede encontrar fácilmente en el mercado.

- Se optó por el uso de Circuitos Integrados con comunicación SPI e I2C tales como los MCP4822, MCP3202, MCP3551, MCP3421, etc. por dos razones: facilitar el montaje de las tarjetas en forma modular, las cuales usan los mismos puntos de conexión para alimentación y comunicación y por la facilidad de la implementación y desarrollo de los protocolos antes mencionados en el microcontrolador que actúa como Master.

- El microcontrolador que se utilizó para que trabaje como dispositivo Modbus esclavo fue el PIC16F887 ya que nos brinda gran capacidad de memoria de programa (8kb), un puerto USART, un puerto MSSP para I2C y SPI, 2 módulos CCP, y pines de entrada y salida multipropósito, además es de bajo costo y se lo puede adquirir a nivel local muy fácilmente en empaquetado TQFP.

- En la tarjeta master de comunicaciones se utilizó un microcontrolador extra para la implementación de la comunicación USB para lo que se seleccionó el PIC que cumpla con nuestros requerimientos de funcionalidad y tamaño con lo que se llego a la conclusión de que el más idóneo fue el PIC18F2550.

- Se trató en lo posible de usar la mayoría de elementos tanto circuitos integrados, condensadores, resistencias, cristales y leds de tipo suelda superficial para ahorrar al máximo el espacio en los circuitos impresos y hacerlos del menor tamaño posible para lograr así un costo menor en la fabricación de los mismos.

- La selección del software de desarrollo de los programas para los microcontroladores fue basada en el lenguaje de programación, librerías existentes que ayuden en el desarrollo del firmware e interface con el usuario, todas estas características las posee el compilador CCS – PCWHD el cual permite la programación en lenguaje C, cuenta con una gran variedad de librerías y ejemplos de aplicaciones además de poseer un entorno muy amigable de desarrollo.

- Se comprobó que las tramas generadas en la Tarjeta Master como respuestas a las peticiones simuladas por el ModScan32 y el ModbusPoll, cumplen con los requerimientos de dicho protocolo, por tanto se concluye que efectivamente la parte correspondiente a la generación de tramas MODBUS es correcta.

- En el protocolo de pruebas se pudo comprobar que el módulo es compatible con diferentes equipos que disponen de un puerto con comunicación Modbus RTU por medio del cual se realizó una conexión punto a punto sobre RS – 232, también se realizaron pruebas de

comunicaciones en una red RS – 485 y vía Ethernet, obteniendo los resultados esperados en peticiones y respuestas sobre Modbus.

- National Instruments dispone de una gran variedad de herramientas para el desarrollo de proyectos como el módulo en cuestión, entre estas herramientas tenemos a Driver Wizard del paquete VISA, el cual nos sirve para crear controladores para que el hardware pueda interactuar con el sistema operativo basándonos en las características propias del dispositivo USB RAW, además se dispone de un bloque de función el cual puede ser usado en el entorno de programación gráfica de Labview llamado Intrument I/O Assistant, el cual servirá como enlace entre las variables físicas y la interfaz gráfica creada en Labview.

- Por las múltiples prestaciones que posee el Módulo de adquisición de datos, fácilmente puede competir en el mercado con equipos que realizan funciones similares, para este caso se va a citar el módulo de pruebas presentado por expositores de National Instruments del Ecuador, el cual simula una planta de procesos básica con motores, sensores, actuadores, etc. Y tiene un costo de USD 380, este equipo debe funcionar con una de las tarjetas de adquisición de datos de NI la cual tiene un costo de USD 1200, el módulo construido busca tener en un solo equipo la mini planta de procesos y el módulo de adquisición de datos a un precio bastante reducido en comparación a los que se ofrecen en el mercado.

- En el mercado nacional se puede encontrar los elementos que fueron utilizados en el Módulo de adquisición de datos a un precio muy accesible, por lo que una inversión sería bien recompensada, basándonos en la ganancia que se puede obtener, al realizar un trabajo de calidad, muy útil en aplicaciones industriales y que puede remplazar a equipos de las mismas características pero a costo reducido.

- El único inconveniente en cuanto a costos, es el que se presenta al momento de realizar los circuitos impresos, una sola empresa ecuatoriana realiza esta labor y ofrece un trabajo de calidad, el cual por efectos de oferta y demanda presenta precios relativamente elevados.

- Es muy importante el control de flujo cuando de comunicación half dúplex se trata ya que debemos asegurarnos que no exista una superposición de información, tomando en cuenta que para este tipo de configuración en determinado momento se admite solo transmisión ó solo recepción, considerando que los equipos conectados en una red RS485 Half Duplex comparten el bus formado por dos líneas A y B.

Recomendaciones

- Al momento de utilizar el puerto serial MSSP de microcontrolador utilizando el módulo I2C y SPI al mismo tiempo se recomienda usar un CLK (reloj de sincronización) distinto para cada caso para evitar conflictos de datos.

- Sugerimos el uso de amplificadores operacionales tales como el AD822 o el AD620 debido a sus características de trabajo como son “RAIL TO RAIL” (rango a rango) y su mayor resistencia a factores externos tales como temperatura y ruido los que ayuda a realizar conversiones mas exactas y con mayor rango de lectura, cosa que en circuitos integrados tales como TL082 o LM358 no se las puede obtener.

- El compilador CCS – PCWHD por su entorno amigable de desarrollo y por sus características de compilación que reducen el código basura al mínimo para aprovechar al máximo la memoria que disponemos en nuestro microcontrolador, es una muy buena opción para los estudiantes que deseen incursionar en la programación el lenguaje C.

- Pretendemos por medio del presente trabajo impulsar el desarrollo propio de diferentes tipos de equipos electrónicos en los estudiantes de Ingeniería Eléctrica de la UPS para que así en un futuro tengan una mayor capacidad de respuesta ante retos que se les ponga a nivel académico y cotidiano.

- Se recomienda el uso del software analizador de tramas Modbus Poll ya que nos fue de mucha utilidad al momento de comparar las tramas recibidas desde dispositivos industriales de marcas tales como Allen Bradley y REDLion con las tramas que se generaron desde el módulo de adquisición de datos.

Glosario

Automatización	Aplicación de procedimientos automáticos a un aparato, proceso o sistema
CAN	(*Controller Area Network*). Es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH, basado en una topología bus para la transmisión de mensajes en entornos distribuidos. Además ofrece una solución a la gestión de la comunicación entre múltiples (unidades centrales de proceso).
CMOS	*Complementary metal-oxide-semiconductor*. Es una de las familias lógicas empleadas en la fabricación de circuitos integrados. Su principal característica consiste que en estado de reposo, el consumo de energía es únicamente el debido a las corrientes parásitas
Código Ascii	(American Standard Code for Information Interchange — *Código Estándar Estadounidense para el Intercambio de Información*), es un código de caracteres basado en el alfabeto latino, tal como se usa en inglés moderno y en otras lenguas occidentales.
Compilador	Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar.
Conversor ADC	Es un dispositivo electrónico capaz de convertir una entrada

	analógica de voltaje en un valor binario.
Conversor DAC	Es un dispositivo electrónico capaz de convertir una entrada binaria en un valor de voltaje a la salida.
Driver	Un controlador de dispositivo, es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz -posiblemente estandarizada- para usarlo. Por tanto, es una pieza esencial, sin la cual no se podría usar el hardware.
Efecto Seebeck	Consiste en la generación de una diferencia de potencial eléctrico al someter a una diferencia de temperatura dos metales o semiconductores diferentes.
Firmware	Es un bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria de tipo de solo lectura (ROM, EEPROM, flash, etc), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Está fuertemente integrado con la electrónica del dispositivo siendo el software que tiene directa interacción con el hardware: es el encargado de controlarlo para ejecutar correctamente las instrucciones externas.
Galga extensiométrica	Dispositivo de medida universal que se utiliza para la medición electrónica de diversas magnitudes mecánicas como pueden ser la presión, carga, torque, deformación, posición, etc. cuerpo debida a la fuerza aplicada sobre él.
Gateway	Una pasarela o puerta de enlace, es un dispositivo, que permite

	interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red, al protocolo usado en la red de destino.
HMI	Interfaz de usuario por sus siglas en idioma inglés, (Human Machine Interface) que se usa para referirse a la interacción entre humanos y máquinas. Aplicable a sistemas de Automatización de procesos.
Lenguaje C	C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B. Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.
Lenguaje Ensamblador	Assembler (assembly language en inglés) es un lenguaje de programación de bajo nivel para los computadores, microprocesadores, y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura dada, constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador.
LSB	Siglas de Less Significative Bit, bit menos significativo.

Memoria EEPROM	EEPROM son las siglas de *Electrically Erasable Programmable Read-Only Memory* (ROM programable y borrable eléctricamente). Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente.
Memoria Flash	La memoria flash es una tecnología de almacenamiento —derivada de la memoria EEPROM— que permite la lecto-escritura de múltiples posiciones de memoria en la misma operación. Gracias a ello, la tecnología *flash*, siempre mediante impulsos eléctricos, permite velocidades de funcionamiento muy superiores frente a la tecnología EEPROM, que sólo permitía actuar sobre una única celda de memoria en cada operación de programación.
Memoria RAM	La memoria de acceso aleatorio (en inglés: *random-access memory*),se utiliza como memoria de trabajo para el sistema operativo, los programas y la mayoría del software.
Memoria ROM	La memoria de solo lectura, conocida también como ROM (acrónimo en inglés de *read-only memory*), es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solo la lectura de la información y no su escritura.
Modelo OSI	El modelo de interconexión de sistemas abiertos, también llamado OSI (en inglés *open system interconnection*) es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización en el año 1984. Es decir, es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

MSB	Siglas de Most Significative Bit, bit más significativo.
OLE	Object Linking and Embedding (OLE) cuya traducción literal es "incrustación y enlazado de objetos" es el nombre de un sistema de objeto distribuido y un protocolo desarrollado por Microsoft.
OPC Server	El OPC (*OLE for Process Control*) es un estándar de comunicación en el campo del control y supervisión de procesos industriales. La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. El servidor OPC es la fuente de datos (como un dispositivo hardware a nivel de planta) y cualquier aplicación basada en OPC puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor.
Optoacoplador	Es un dispositivo de emisión y recepción que funciona como un interruptor excitado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac. Se suelen utilizar para aislar electricamente a dispositivos muy sensibles.
PAC	Un controlador de automatización programable, o PAC (del inglés *Programmable Automation Controller*), es una tecnología industrial orientada al control automatizado, al diseño de prototipos y a la medición. El PAC se refiere al conjunto formado por un controlador (una CPU típicamente), módulos de entradas y salidas, y uno o múltiples buses de datos que lo interconectan todo.
PLC	(*Programmable Logic Controller* en sus siglas en inglés) son dispositivos electrónicos muy usados en automatización

	industrial.
Protocolo de comunicaciones	En el campo de las telecomunicaciones, es el conjunto de reglas normalizadas para la representación, señalización, autenticación y detección de errores necesario para enviar información a través de un canal de comunicación.
Puente de Wheatstone	Se utiliza para medir resistencias desconocidas. Estos están constituidos por cuatro resistencias que forman un circuito cerrado, siendo una de ellas la resistencia bajo medida
PWM	La modulación por ancho de pulsos de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.
Router	Conocido como encaminador, enrutador, direccionador o ruteador es un dispositivo de hardware usado para la interconexión de redes informáticas que permite asegurar el direccionamiento de paquetes de datos entre ellas o determinar la mejor ruta que deben tomar.
SCADA	Proviene de las siglas "Supervisory Control And Data Acquisition" (Control de Supervisión y Adquisición de Datos): Es un sistema basado en computadores que permite supervisar y controlar variables de proceso a distancia, proporcionando comunicación con los dispositivos de campo.
Sistema Binario	Es un sistema de numeración en el que los números se

	representan utilizando solamente las cifras cero y uno (*0* y *1*).
Sistema Decimal	Es un sistema de numeración posicional en el que las cantidades se representan utilizando como base aritmética las potencias del número diez.
Sistema hexadecimal	Es un sistema de numeración que emplea 16 símbolos. Su uso actual está muy vinculado a la informática y ciencias de la computación, pues los computadores suelen utilizar el byte u octeto como unidad básica de memoria.
Transductor	Los transductores son aquellas partes de una cadena de medición que transforman una magnitud física en una señal eléctrica.
Transceptor	Dispositivo que realiza, dentro de una misma caja o chasis, funciones tanto de trasmisión como de recepción, utilizando componentes de circuito comunes para ambas funciones.
Transistor darlington	Es un dispositivo semiconductor que combina dos transistores bipolares en un tándem (a veces llamado *par Darlington*) en un único dispositivo. La configuración (originalmente realizada con dos transistores separados) fue inventada por el ingeniero de los Laboratorios Bell Sidney Darlington. La idea de poner dos o tres transistores sobre un chip fue patentada por él, pero no la idea de poner un número arbitrario de transistores que originaría la idea moderna de circuito integrado.
TTL	*Transistor-transistor logic*. Es una tecnología de construcción de circuitos electrónicos digitales. En los componentes fabricados

	con tecnología TTL los elementos de entrada y salida del dispositivo son transistores bipolares.
UAL	Unidad aritmético lógica, también conocida como ALU (siglas en inglés de *arithmetic logic unit*), es un circuito digital que calcula operaciones aritméticas (como suma, resta, multiplicación, etc.) y operaciones lógicas (si, y, o, no), entre dos números.
UDP/IP	Una manera directa de enviar y recibir datagramas a través una red IP. Se utiliza sobre todo cuando la velocidad es un factor importante en la transmisión de la información, por ejemplo, RealAudio utiliza el UDP.
USART	Es el acrónimo de *Universal Synchronous/Asynchronous Receiver Transmitter*, que traducido al español viene a ser algo parecido a *Transmisor y Receptor Sincrónico/Asincrónico Universal*.
VID	VID - Vendor ID (es un número de 16 bits vendido a un fabricante de productos USB).
PID	PID - Product ID, identificación del producto en cuestión

Referencias.

[1]. CREUS, Antonio, *"Instrumentación Industrial"*, 6ª edición; Marcombo S.A.; 1997.

[2]. BOLTON, W. "*MECATRONICA: Sistemas de Control Electrónico en Ingeniería Mecánica y Eléctrica*". Segunda edición. Editorial Alfaomega, 2001, México.

[3]. TEXAS INSTRUMENTS, "*Understanding Data Converters Application Report*", 1997, ref nº SLAA013.

[4]. PALLÁS, Ramón, "*Adquisición y Distribución de Señales*". Editorial Marcombo.

[5]. LOVEDAY, G. C. "*Diseño de Hardware Electrónico*". Editorial Paraninfo.

[6]. MICROCHIP, "*Embedded Control handbook*" Volumen 1.

[7]. GONZALES, Nestor *"Comunicaciones y redes de procesamiento de datos"*, editorial McGraw-Hill.

[8]. HUIDROBRO, José, *"Comunicaciones: Interfaces, módems, protocolos, redes y normas",* Editorial Paraninfo, segunda edición, 1992.

[9]. FORTEZA, Bonnin, "*Fuentes de alimentación reguladas Electrónicamente*", Editorial Marcombo.

[10]. http://live.altium.com

[11]. http://www.ccsinfo.com/

[12]. http://ww1.microchip.com/downloads/en/devicedoc/21953a.pdf

[13]. http://ww1.microchip.com/downloads/en/DeviceDoc/21034D.pdf

[14]. http://ww1.microchip.com/downloads/en/devicedoc/21950b.pdf

[15]. http://ww1.microchip.com/downloads/en/DeviceDoc/22003b.pdf

[16]. http://ww1.microchip.com/downloads/en/devicedoc/21953a.pdf

[17]. http://www.analog.com/static/imported-files/data_sheets/AD822.pdf

[18]. http://www.parallax.com/Portals/0/Downloads/docs/books/edu/ICSpanish.pdf

[19]. http://www.sapiensman.com/control_automatico/

[20]. http://www.olimex.cl

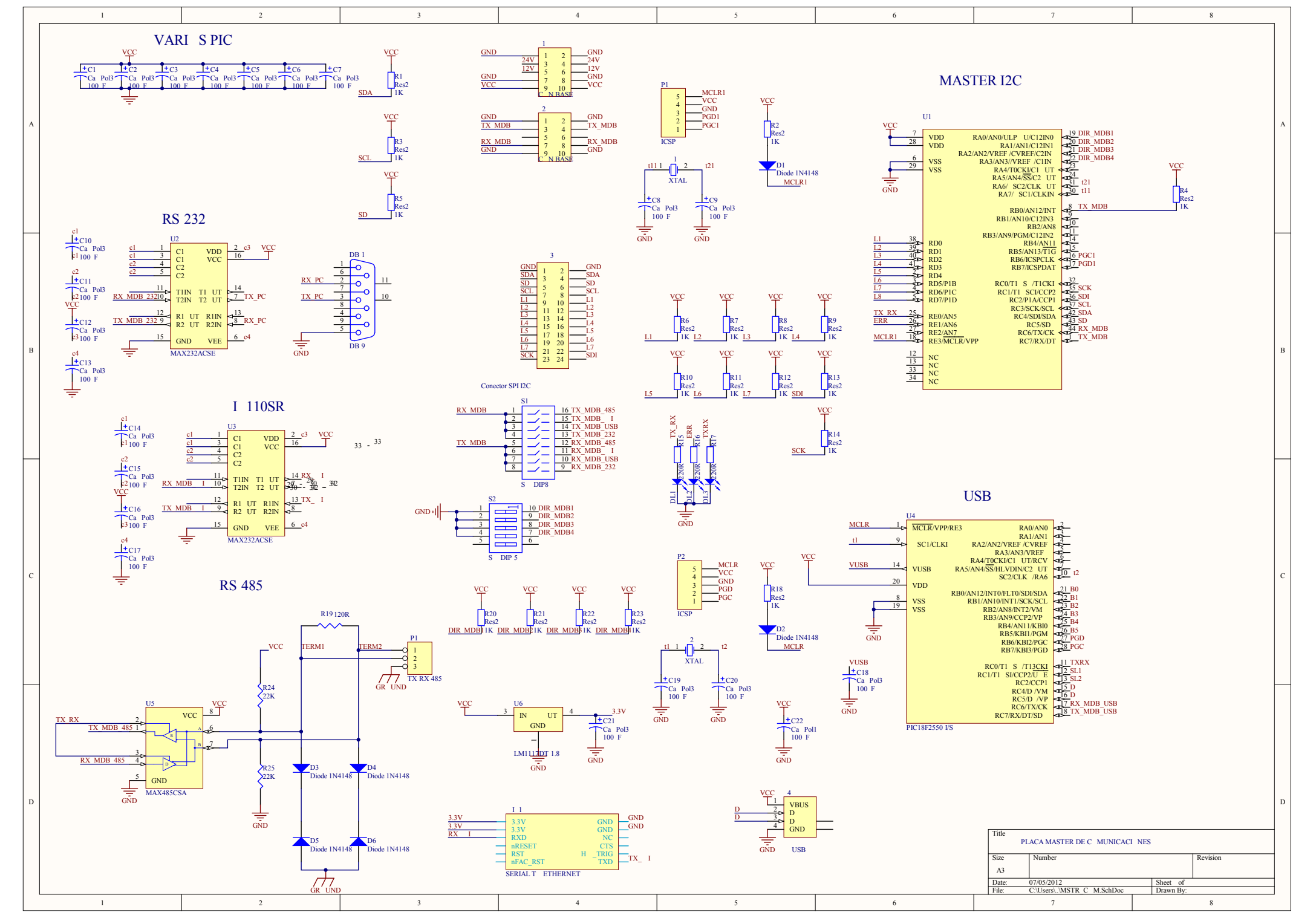
[21]. http://www.ucontrol.com

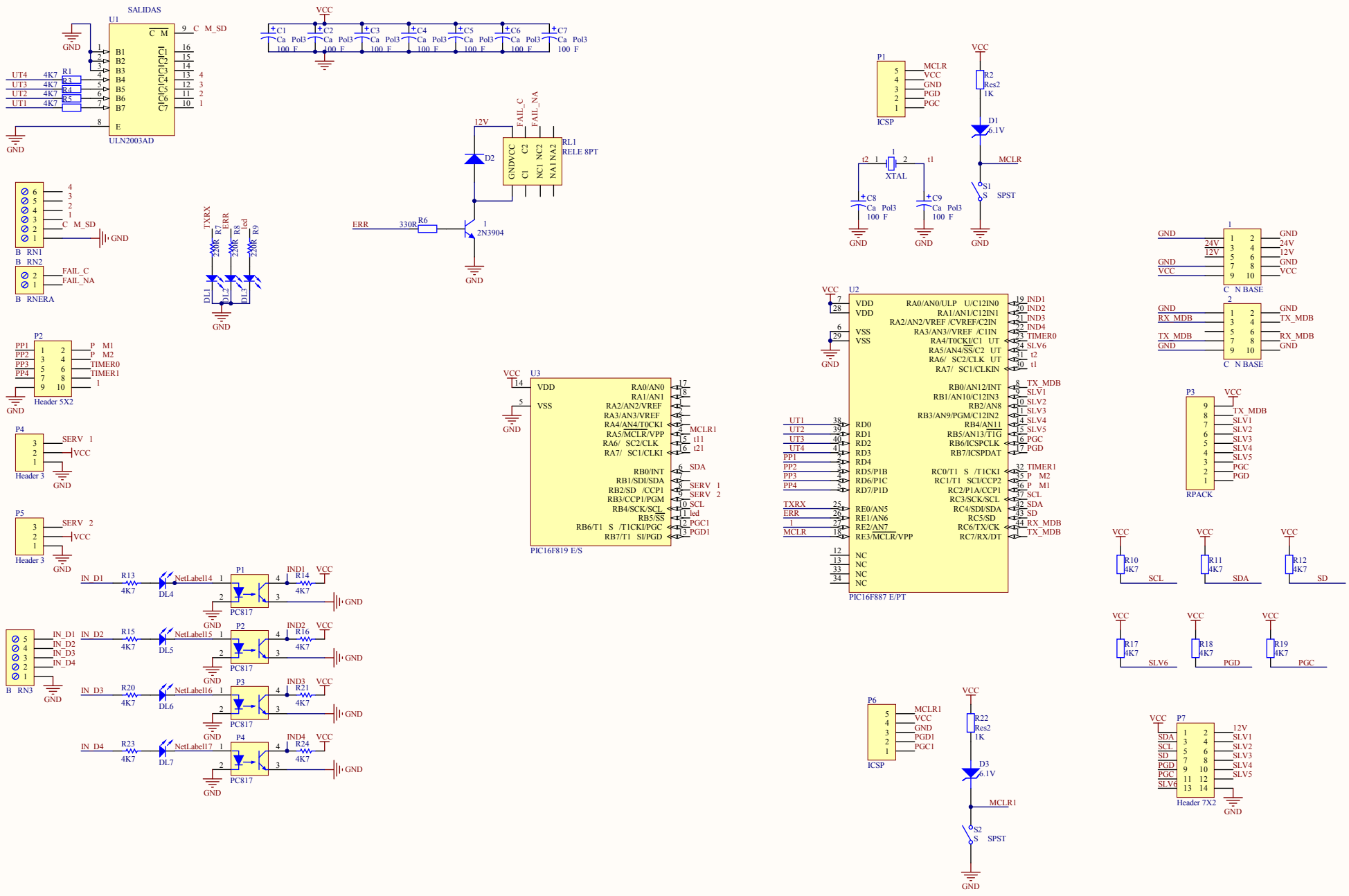
[22]. http://www.aquihayapuntes.com.

ANEXOS

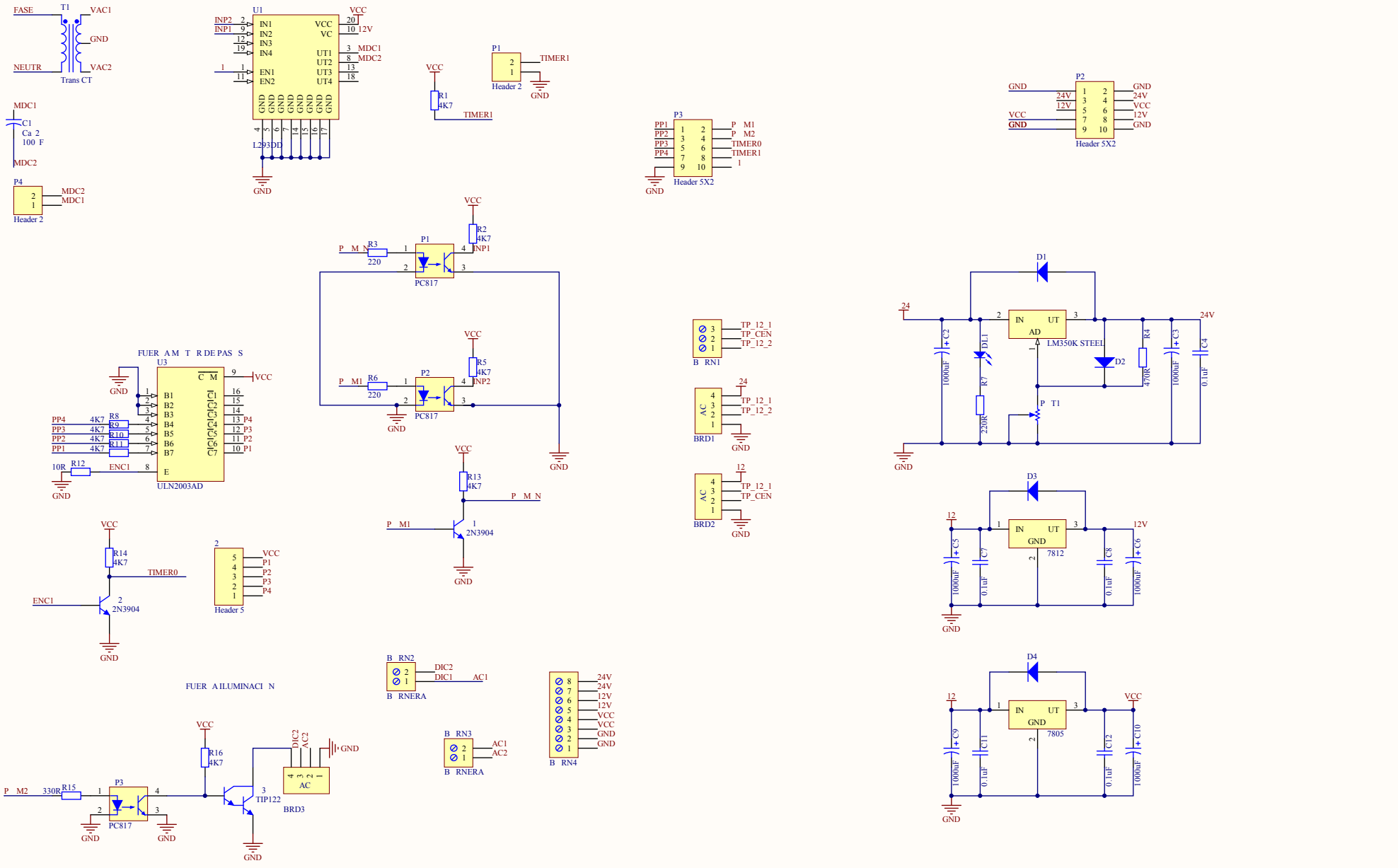
ANEXO I

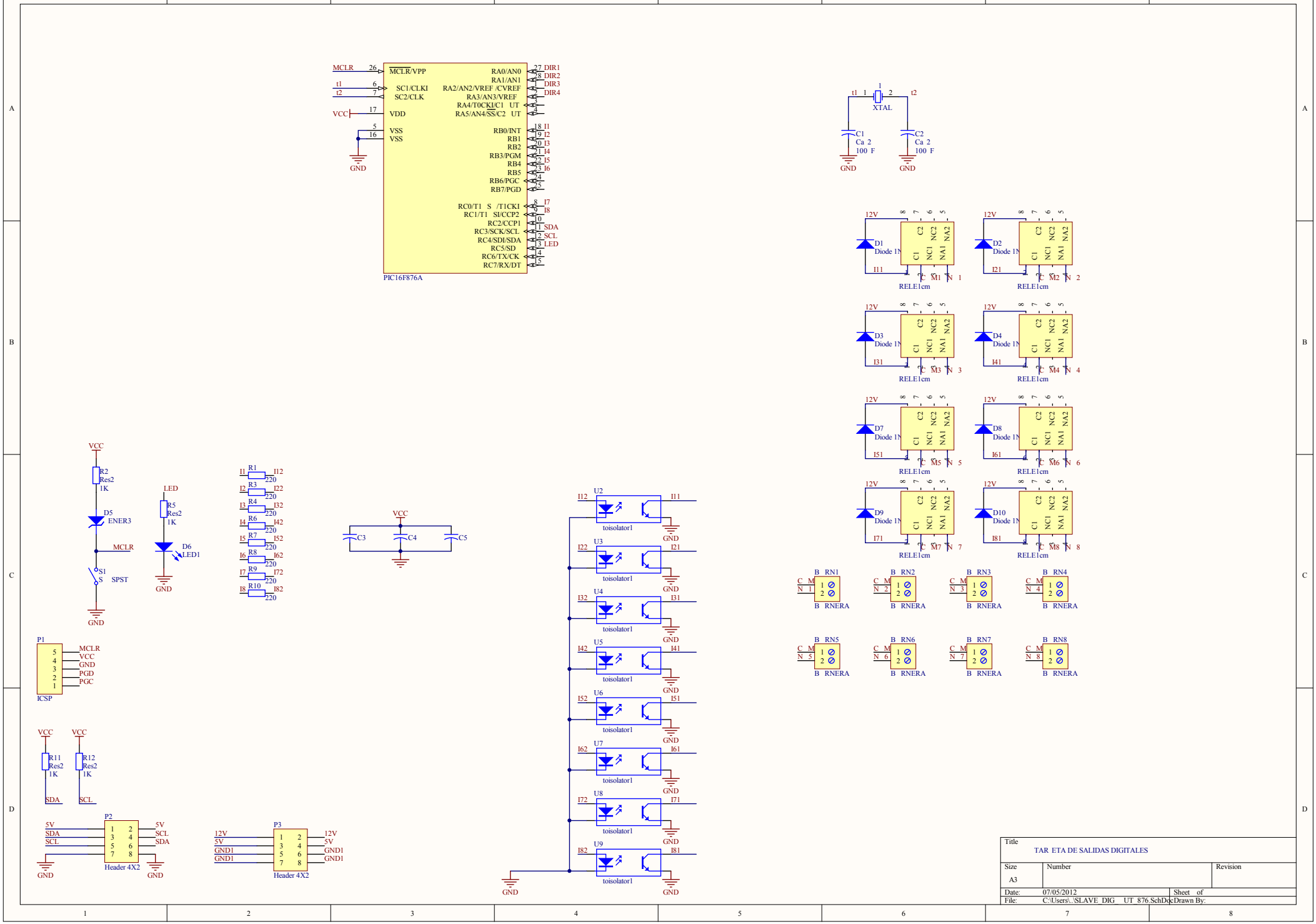
DIAGRAMAS ESQUEMÁTICOS DE LOS CIRCUITOS IMPRESOS

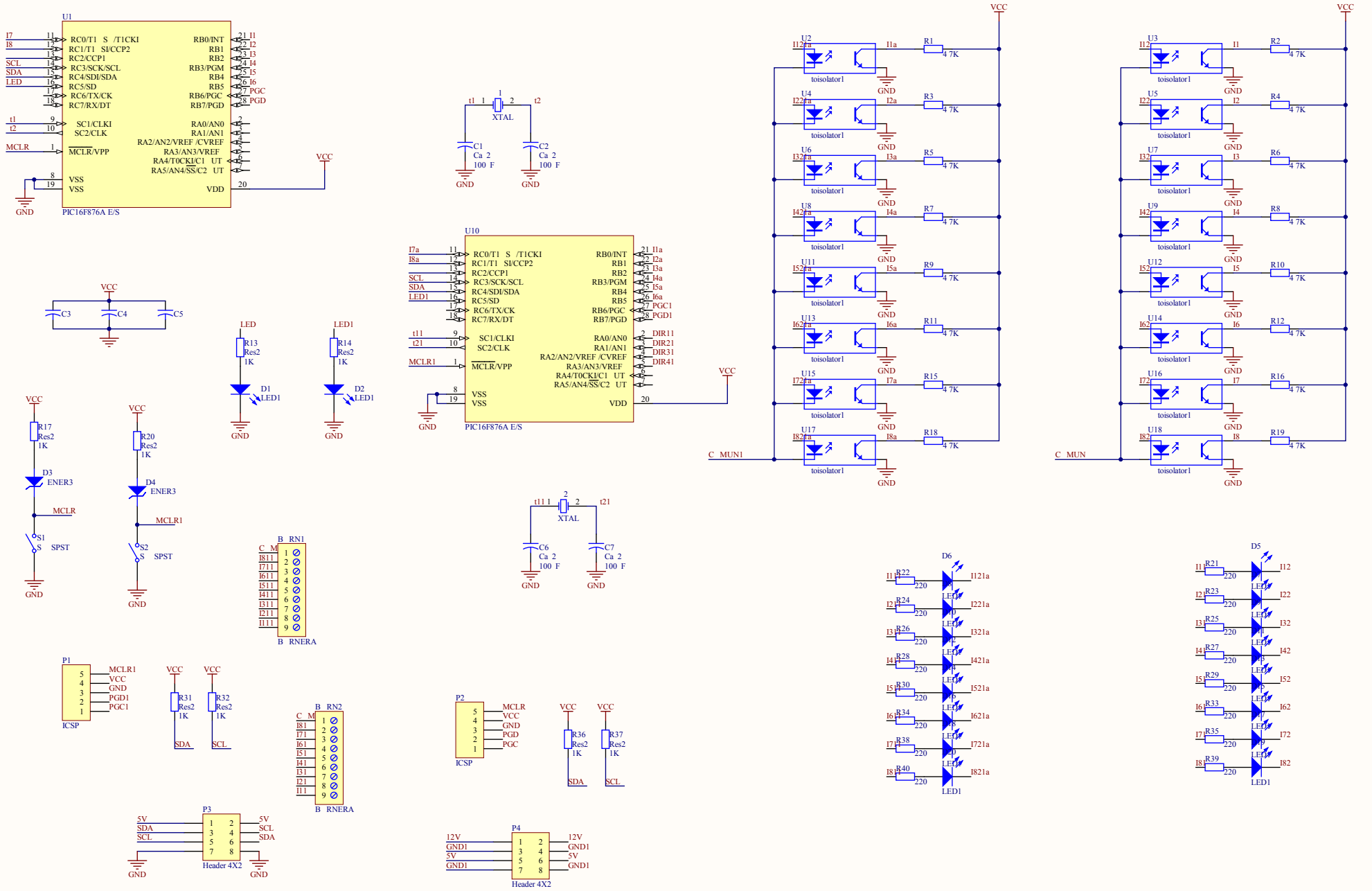

VARI S PIC
RS 232
MAX232ACSE
DB 9
I 110SR
MAX232ACSE
RS 485
MAX485CSA
TX RX 485
Conector SPI I2C
S DIP8
S DIP 5
MASTER I2C
PIC18F2550 I/S
USB
SERIAL T ETHERNET
PLACA MASTER DE C MUNICACI NES
Size
A3
Number
Revision
Date: 07/05/2012
Sheet of
File: C:\Users\..\MSTR_C M.SchDoc
Drawn By:



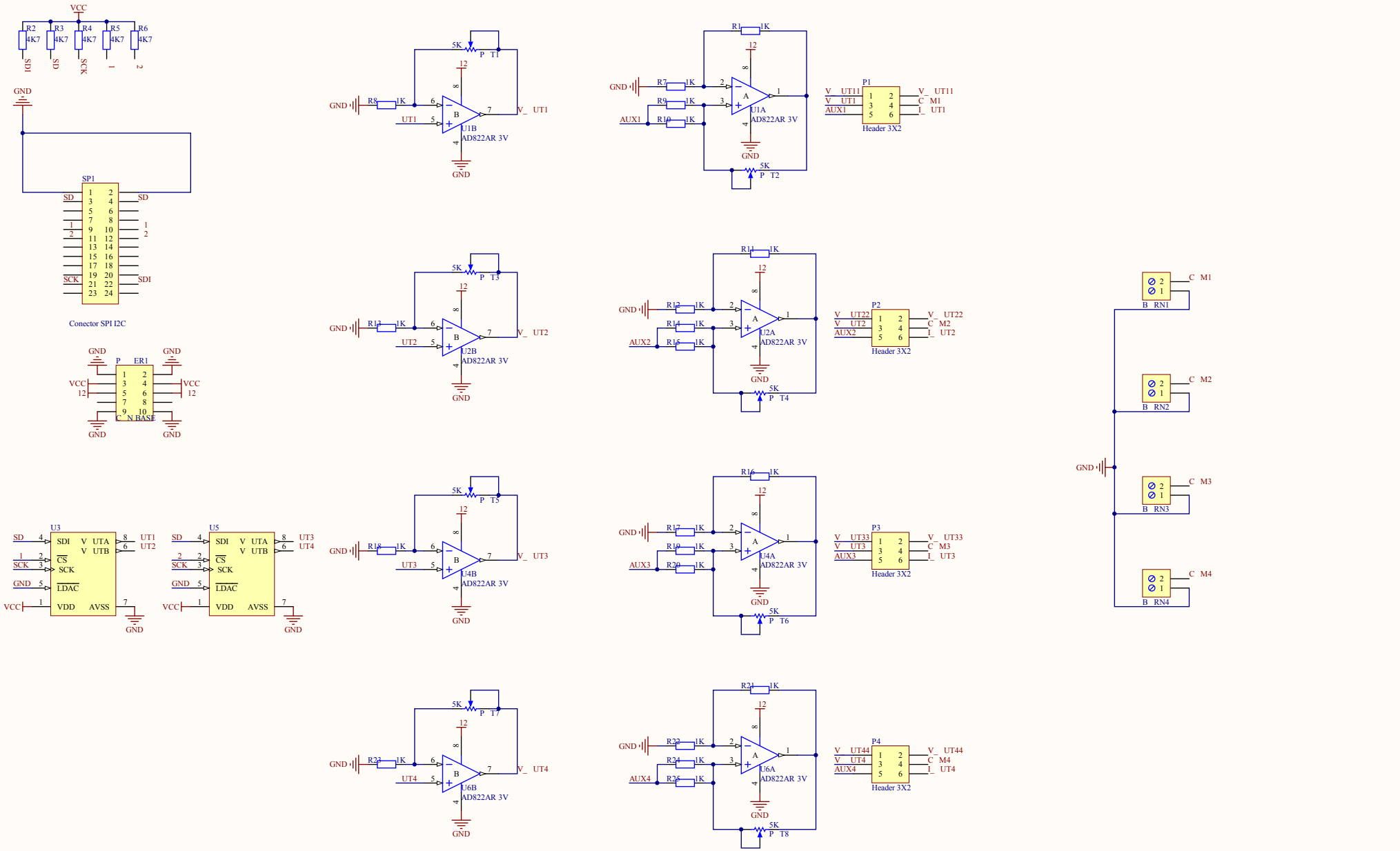
Title	PLACA MASTER DE PR CES S	
Size: A3	Number	Revision
Date: 07/05/2012		Sheet of
File: C:\Users\..\MSTR_PRC.SchDoc		Drawn By:


FASE
T1
VAC1
GND
NEUTR
VAC2
Trans CT
MDC1
C1
Ca 2
100 F
MDC2
P4
Header 2
U1
INP2
INP1
IN1
IN2
IN3
IN4
EN1
EN2
VCC
VC
10 12V
MDC1
MDC2
L293DD
R1
4K7
TIMER1
P1
Header 2
P3
PP1
PP2
PP3
PP4
P M1
P M2
TIMER0
Header 5X2
P2
24V
12V
Header 5X2
FUER A M T R DE PAS S
U3
ULN2003AD
R8
R9
R10
R11
R12
ENC1
10R
R2
R3
220
PC817
R5
R6
R13
2N3904
P M N
R14
TIMER0
Header 5
B RN1
TP_12_1
TP_CEN
TP_12_2
BRD1
BRD2
D1
LM350K STEEL
D2
D3
D4
7812
7805
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
R4
R7
220R
470R
P T1
1000uF
0.1uF
FUER A ILUMINACI N
R15
330R
R16
TIP122
BRD3
B RN2
B RNERA
B RN3
B RN4
DIC2
DIC1
AC1
AC2
Title
TAR ETA DE FUER A C NTR L DE PR CES S
Size
A3
Number
Revision
Date:
07/05/2012
Sheet of
File:
C:\Users\..\FUER A.SchDoc
Drawn By:

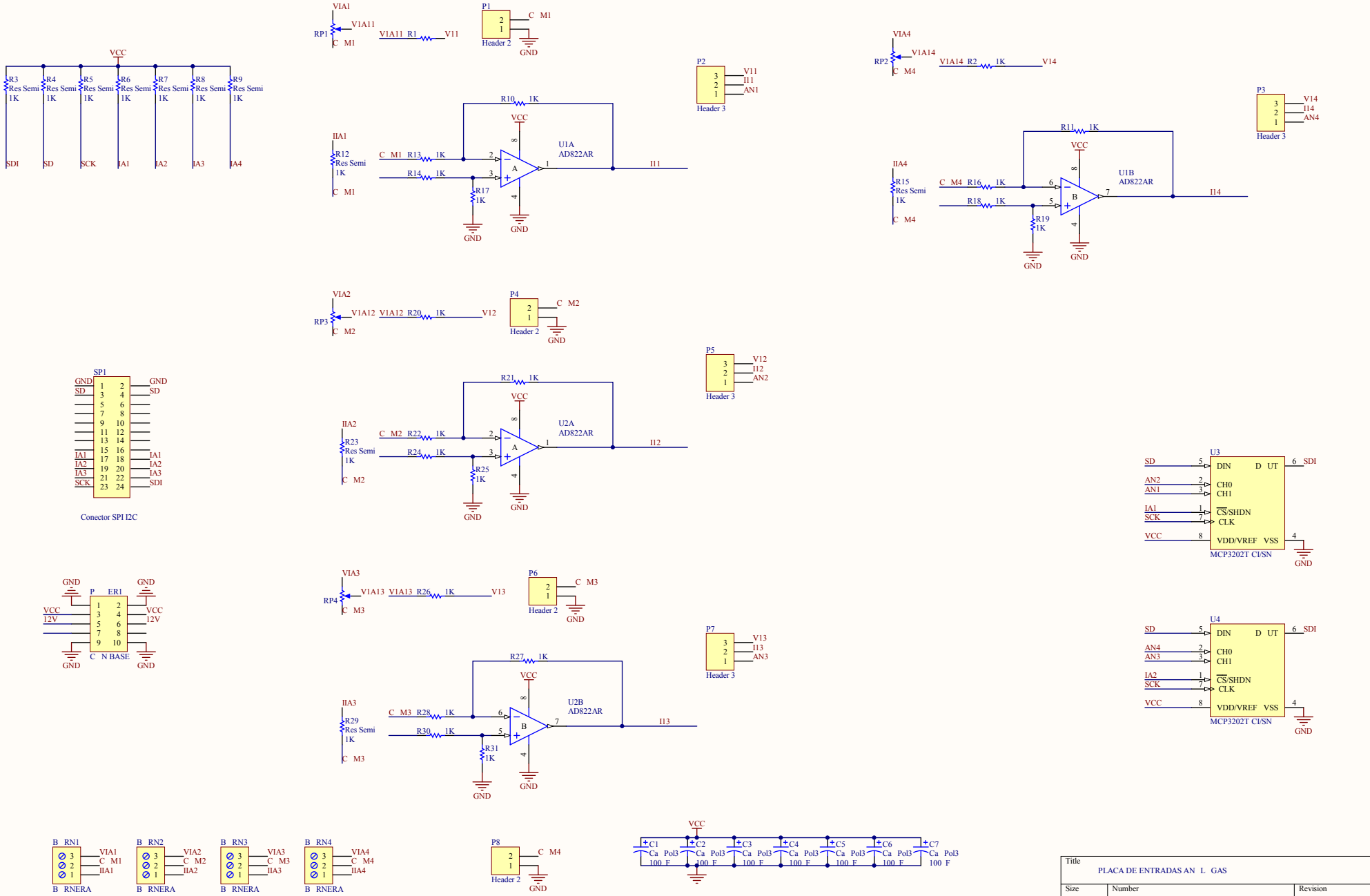

MCLR
t1
t2
VCC
GND
MCLR/VPP
SC1/CLKI
SC2/CLK
VDD
VSS
VSS
RA0/AN0
RA1/AN1
RA2/AN2/VREF /CVREF
RA3/AN3/VREF
RA4/T0CKI/C1 UT
RA5/AN4/SS/C2 UT
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1 S /T1CKI
RC1/T1 SI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SD
RC6/TX/CK
RC7/RX/DT
DIR1
DIR2
DIR3
DIR4
I1
I2
I3
I4
I5
I6
I7
I8
SDA
SCL
LED
PIC16F876A
XTAL
C1
C2
Ca 2
100 F
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
Diode 1N
RELE1cm
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
Res2
1K
220
ENER3
LED1
SPST
S1
C3
C4
C5
U2
U3
U4
U5
U6
U7
U8
U9
toisolator1
RN1
RN2
RN3
RN4
RN5
RN6
RN7
RN8
RNERA
P1
ICSP
P2
P3
Header 4X2
5V
12V
GND1
Title
TAR ETA DE SALIDAS DIGITALES
Size
A3
Number
Revision
Date:
07/05/2012
Sheet of
File:
C:\Users\..\SLAVE_DIG_UT_876.SchDoc
Drawn By:

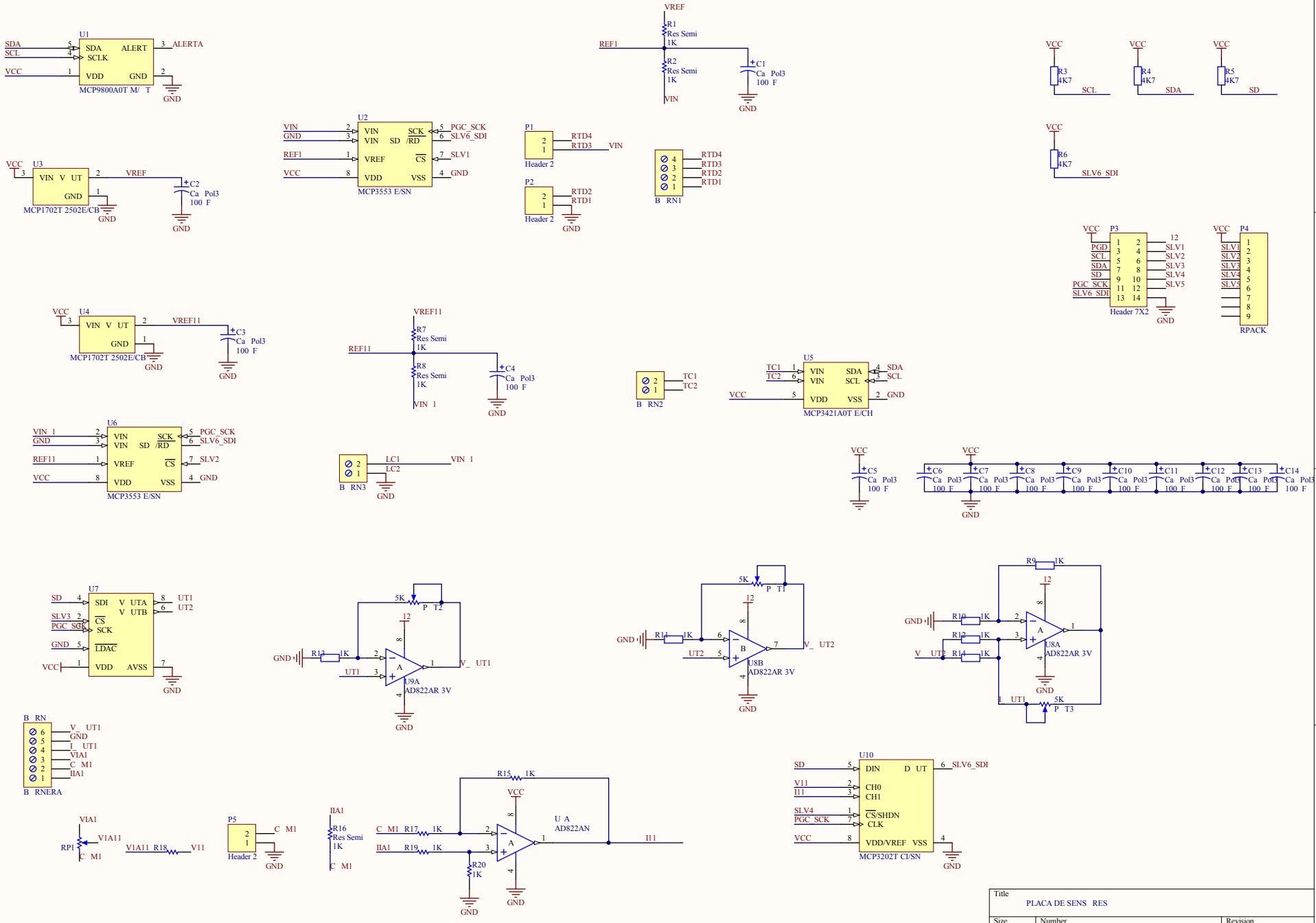


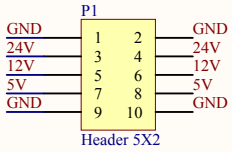
Title	TAR ETA DE ENTRADAS DIGITALES	
Size A3	Number	Revision
Date: 07/05/2012		Sheet of
File: C:\Users\..\SLAVE_DIG_IN_876.SchDoc		Drawn By:

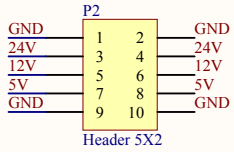


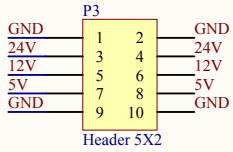
Title	TAR ETA DE SALIDAS AN L GAS	
Size: A3	Number	Revision
Date: 07/05/2012		Sheet of
File: C:\Users\..\SLAVE_AN UT.SchDoc		Drawn By:

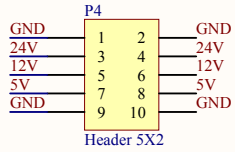

VCC
R3 Res Semi 1K
R4 Res Semi 1K
R5 Res Semi 1K
R6 Res Semi 1K
R7 Res Semi 1K
R8 Res Semi 1K
R9 Res Semi 1K
SDI
SD
SCK
IA1
IA2
IA3
IA4
VIA1
RP1
V1A11
C M1
V1A11 R1
V11
P1
Header 2
GND
P2
Header 3
V11
I11
AN1
R10 1K
IIA1
R12 Res Semi 1K
C M1 R13 1K
R14 1K
R17 1K
U1A
AD822AR
I11
VIA4
RP2
V1A14
C M4
V1A14 R2 1K
V14
P3
V14
I14
AN4
R11 1K
IIA4
R15 Res Semi 1K
C M4 R16 1K
R18 1K
R19 1K
U1B
AD822AR
I14
VIA2
RP3
V1A12
C M2
V1A12 R20 1K
V12
P4
C M2
P5
V12
I12
AN2
R21 1K
IIA2
R23 Res Semi 1K
C M2 R22 1K
R24 1K
R25 1K
U2A
AD822AR
I12
SP1
GND
SD
IA1
IA2
IA3
SCK
Conector SPI I2C
U3
DIN
D UT
CH0
CH1
CS/SHDN
CLK
VDD/VREF
VSS
MCP3202T CI/SN
AN1
VIA3
RP4
V1A13
C M3
V1A13 R26 1K
V13
P6
C M3
P7
V13
I13
AN3
R27 1K
IIA3
R29 Res Semi 1K
C M3 R28 1K
R30 1K
R31 1K
U2B
AD822AR
I13
U4
AN4
P ER1
C N BASE
VCC
12V
B RN1
B RN2
B RN3
B RN4
B RNERA
P8
C1
C2
C3
C4
C5
C6
C7
Ca Pol3
100 F
Title
PLACA DE ENTRADAS AN L GAS
Size
A3
Number
Revision
Date: 07/05/2012
File: C:\Users\..\SLAVE_AN_IN.SchDoc
Sheet of
Drawn By:

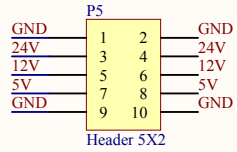

U1
SDA
SCL
VCC
SDA
SCLK
ALERT
VDD
GND
ALERTA
MCP9800A0T M/ T
GND
VREF
R1
Res Semi
1K
REF1
R2
Res Semi
1K
VIN
C1
Ca Pol3
100 F
GND
VCC
R3
4K7
SCL
R4
4K7
SDA
R5
4K7
SD
U2
VIN
GND
REF1
VCC
VIN
SD
VREF
VDD
SCK
/RD
CS
SSS
PGC_SCK
SLV6_SDI
SLV1
GND
MCP3553 E/SN
P1
Header 2
RTD4
RTD3
VIN
P2
Header 2
RTD2
RTD1
GND
B RN1
RTD4
RTD3
RTD2
RTD1
VCC
R6
4K7
SLV6_SDI
U3
VCC
VIN V UT
GND
VREF
MCP1702T 2502E/CB
C2
Ca Pol3
100 F
GND
GND
VCC
P3
PGD
SCL
SDA
SD
PGC_SCK
SLV6_SDI
SLV1
SLV2
SLV3
SLV4
SLV5
Header 7X2
GND
VCC
P4
SLV1
SLV2
SLV3
SLV4
SLV5
RPACK
U4
VCC
VIN V UT
GND
VREF11
MCP1702T 2502E/CB
C3
Ca Pol3
100 F
GND
GND
VREF11
R7
Res Semi
1K
REF11
R8
Res Semi
1K
VIN 1
C4
Ca Pol3
100 F
GND
B RN2
TC1
TC2
U5
TC1
TC2
VCC
VIN
VIN
VDD
SDA
SCL
VSS
SDA
SCL
GND
MCP3421A0T E/CH
U6
VIN 1
GND
REF11
VCC
VIN
VIN
VREF
VDD
SCK
/RD
CS
SSS
PGC_SCK
SLV6_SDI
SLV2
GND
MCP3553 E/SN
B RN3
LC1
LC2
VIN 1
GND
VCC
C5
Ca Pol3
100 F
VCC
C6
C7
C8
C9
C10
C11
C12
C13
C14
Ca Pol3
100 F
GND
U7
SD
SLV3
PGC_SCK
GND
VCC
SDI
V UTA
V UTB
CS
SCK
LDAC
VDD
AVSS
UT1
UT2
GND
5K
P T2
12
R13
1K
GND
U9A
AD822AR 3V
V_ UT1
5K
P T1
12
R11
1K
GND
U8B
AD822AR 3V
V_ UT2
R9
1K
12
GND
R10
1K
R12
1K
V_ UT2
R14
1K
U8A
AD822AR 3V
GND
1_ UT1
5K
P T3
B RN
V_ UT1
GND
I_ UT1
VIA1
C M1
IIA1
B RNERA
VIA1
RP1
C M1
V1A11
V1A11 R18
V11
P5
Header 2
C M1
GND
IIA1
R16
Res Semi
1K
C M1
R15
1K
VCC
C M1 R17
1K
IIA1 R19
1K
R20
1K
GND
U A
AD822AN
I11
U10
SD
V11
I11
SLV4
PGC_SCK
VCC
DIN
CH0
CH1
CS/SHDN
CLK
VDD/VREF
D UT
VSS
SLV6_SDI
MCP3202T CI/SN
GND
Title
PLACA DE SENS RES
Size
A3
Number
Revision
Date:
07/05/2012
File:
C:\Users\..\SENS RES.SchDoc
Sheet of
Drawn By:

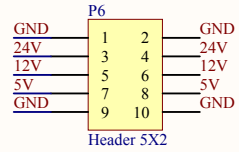


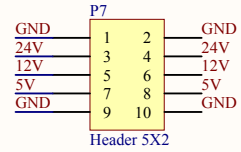


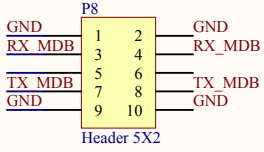


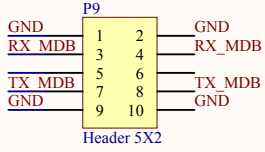


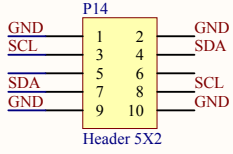


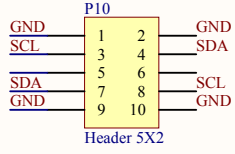


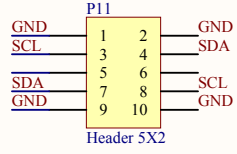


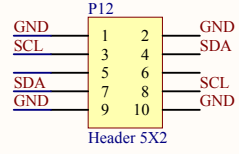


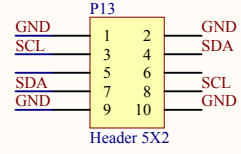












Title: PLACA BACK PANEL (INTERFACES)		
Size: A4	Number:	Revision:
Date: 07/05/2012		Sheet of
File: C:\Users\..\INTERFACES.SchDoc		Drawn By:

ANEXO II

HOJAS DE DATOS

PIC16F818/819
Data Sheet

18/20-Pin
Enhanced Flash Microcontrollers
with nanoWatt Technology

PIC16F818/819

18/20-Pin Enhanced Flash Microcontrollers with nanoWatt Technology

Low-Power Features:

- Power-Managed modes:
  - Primary Run: XT, RC oscillator, 87 μA, 1 MHz, 2V
  - INTRC: 7 μA, 31.25 kHz, 2V
  - Sleep: 0.2 μA, 2V
- Timer1 oscillator: 1.8 μA, 32 kHz, 2V
- Watchdog Timer: 0.7 μA, 2V
- Wide operating voltage range:
  - Industrial: 2.0V to 5.5V

Oscillators:

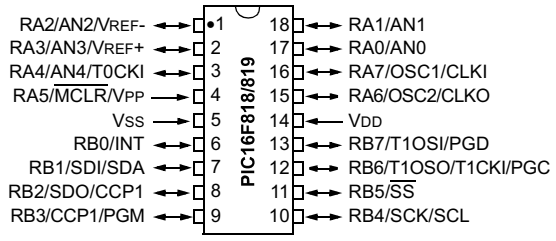
- Three Crystal modes:
  - LP, XT, HS: up to 20 MHz
- Two External RC modes
- One External Clock mode:
  - ECIO: up to 20 MHz
- Internal oscillator block:
  - 8 user selectable frequencies: 31 kHz, 125 kHz, 250 kHz, 500 kHz, 1 MHz, 2 MHz, 4 MHz, 8 MHz

Peripheral Features:

- 16 I/O pins with individual direction control
- High sink/source current: 25 mA
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Capture, Compare, PWM (CCP) module:
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit, 5-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master/Slave) and I^2C™ (Slave)

Pin Diagram

18-Pin PDIP, SOIC



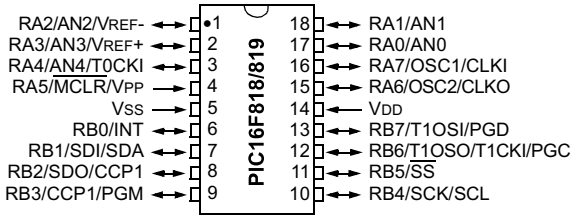
Special Microcontroller Features:

- 100,000 erase/write cycles Enhanced Flash program memory typical
- 1,000,000 typical erase/write cycles EEPROM data memory typical
- EEPROM Data Retention: > 40 years
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Processor read/write access to program memory
- Low-Voltage Programming
- In-Circuit Debugging via two pins

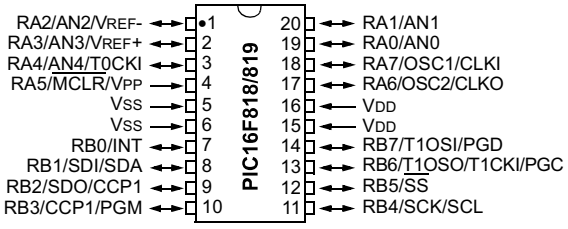
Device	Program Memory		Data Memory		I/O Pins	10-bit A/D (ch)	CCP (PWM)	SSP		Timers 8/16-bit
	Flash (Bytes)	# Single-Word Instructions	SRAM (Bytes)	EEPROM (Bytes)				SPI™	Slave I^2C™	
PIC16F818	1792	1024	128	128	16	5	1	Y	Y	2/1
PIC16F819	3584	2048	256	256	16	5	1	Y	Y	2/1

Pin Diagrams

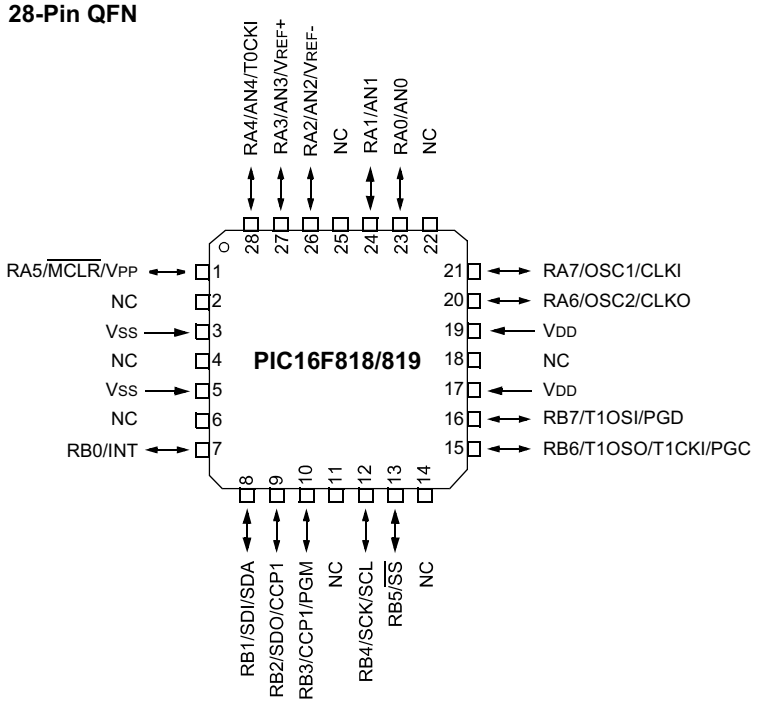
18-Pin PDIP, SOIC



20-Pin SSOP



28-Pin QFN



1.0 DEVICE OVERVIEW

This document contains device specific information for the operation of the PIC16F818/819 devices. Additional information may be found in the *"PICmicro® Mid-Range MCU Family Reference Manual"* (DS33023) which may be 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.

The PIC16F818/819 belongs to the Mid-Range family of the PICmicro® devices. The devices differ from each other in the amount of Flash program memory, data memory and data EEPROM (see Table 1-1). A block diagram of the devices is shown in Figure 1-1. These devices contain features that are new to the PIC16 product line:

- Internal RC oscillator with eight selectable frequencies, including 31.25 kHz, 125 kHz, 250 kHz, 500 kHz, 1 MHz, 2 MHz, 4 MHz and 8 MHz. The INTRC can be configured as the system clock via the configuration bits. Refer to **Section 4.5 "Internal Oscillator Block"** and **Section 12.1 "Configuration Bits"** for further details.
- The Timer1 module current consumption has been greatly reduced from 20 μA (previous PIC16 devices) to 1.8 μA typical (32 kHz at 2V), which is ideal for real-time clock applications. Refer to **Section 6.0 "Timer0 Module"** for further details.
- The amount of oscillator selections has increased. The RC and INTRC modes can be selected with an I/O pin configured as an I/O or a clock output (FOSC/4). An external clock can be configured with an I/O pin. Refer to **Section 4.0 "Oscillator Configurations"** for further details.

TABLE 1-1: AVAILABLE MEMORY IN PIC16F818/819 DEVICES

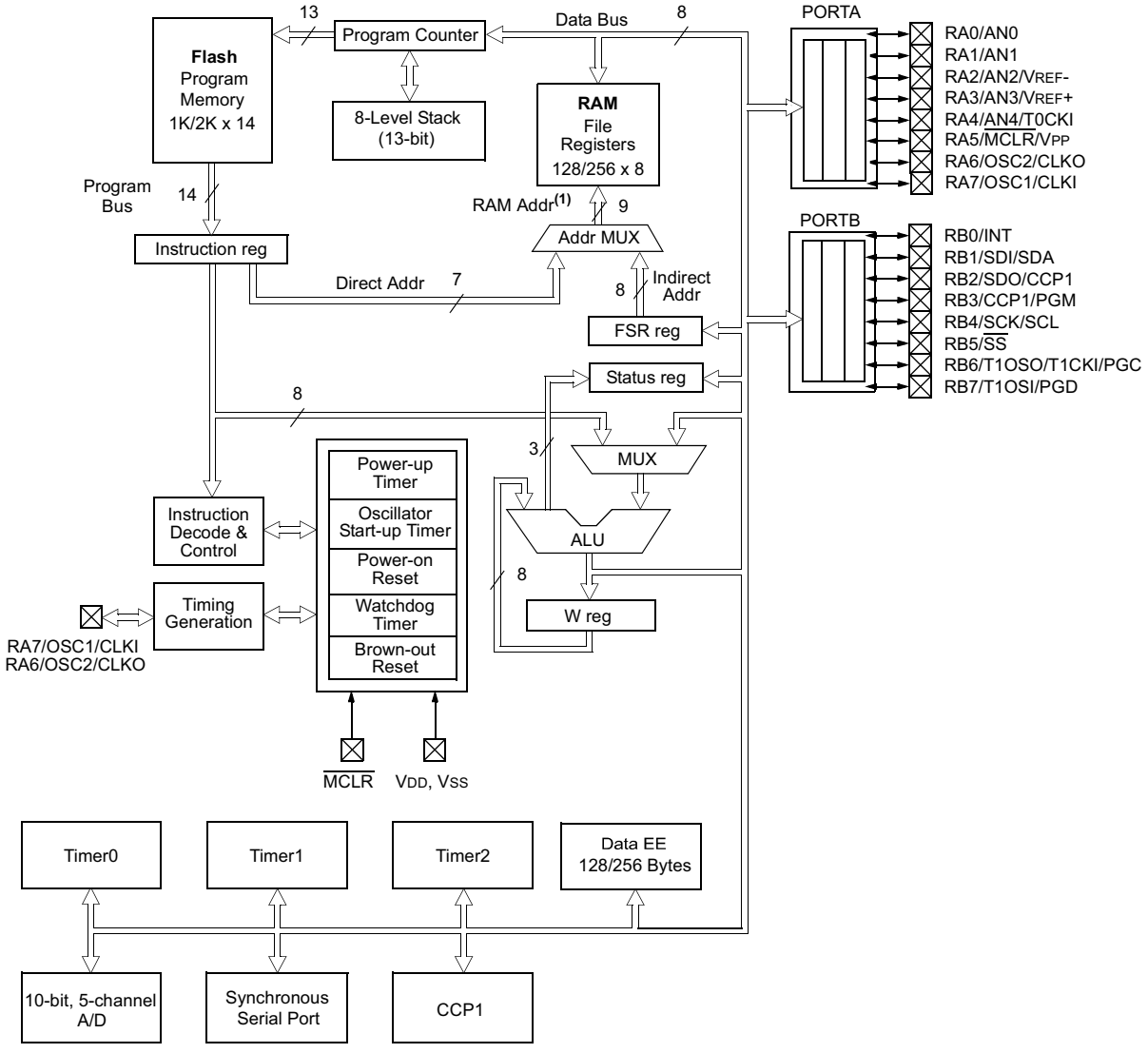
Device	Program Flash	Data Memory	Data EEPROM
PIC16F818	1K x 14	128 x 8	128 x 8
PIC16F819	2K x14	256 x 8	256 x 8

There are 16 I/O pins that are user configurable on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External Interrupt
- Change on PORTB Interrupt
- Timer0 Clock Input
- Low-Power Timer1 Clock/Oscillator
- Capture/Compare/PWM
- 10-bit, 5-channel Analog-to-Digital Converter
- SPI/I^2C
- $\overline{MCLR}$ (RA5) can be configured as an Input

Table 1-2 details the pinout of the devices with descriptions and details for each pin.

FIGURE 1-1: PIC16F818/819 BLOCK DIAGRAM



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

TABLE 1-2: PIC16F818/819 PINOUT DESCRIPTIONS

Pin Name	PDIP/ SOIC Pin#	SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
						PORTA is a bidirectional I/O port.
RA0/AN0	17	19	23			
RA0				I/O	TTL	Bidirectional I/O pin.
AN0				I	Analog	Analog input channel 0.
RA1/AN1	18	20	24			
RA1				I/O	TTL	Bidirectional I/O pin.
AN1				I	Analog	Analog input channel 1.
RA2/AN2/VREF-	1	1	26			
RA2				I/O	TTL	Bidirectional I/O pin.
AN2				I	Analog	Analog input channel 2.
VREF-				I	Analog	A/D reference voltage (low) input.
RA3/AN3/VREF+	2	2	27			
RA3				I/O	TTL	Bidirectional I/O pin.
AN3				I	Analog	Analog input channel 3.
VREF+				I	Analog	A/D reference voltage (high) input.
RA4/AN4/T0CKI	3	3	28			
RA4				I/O	ST	Bidirectional I/O pin.
AN4				I	Analog	Analog input channel 4.
T0CKI				I	ST	Clock input to the TMR0 timer/counter.
RA5/$\overline{\text{MCLR}}$/VPP	4	4	1			
RA5				I	ST	Input pin.
$\overline{\text{MCLR}}$				I	ST	Master Clear (Reset). Input/programming voltage input. This pin is an active-low Reset to the device.
VPP				P	–	Programming threshold voltage.
RA6/OSC2/CLKO	15	17	20			
RA6				I/O	ST	Bidirectional I/O pin.
OSC2				O	–	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	–	In RC mode, this pin outputs CLKO signal which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA7/OSC1/CLKI	16	18	21			
RA7				I/O	ST	Bidirectional I/O pin.
OSC1				I	ST/CMOS(3)	Oscillator crystal input.
CLKI				I	–	External clock source input.

Legend: I = Input O = Output I/O = Input/Output P = Power
– = Not used TTL = TTL Input ST = Schmitt Trigger Input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F818/819 PINOUT DESCRIPTIONS (CONTINUED)

Pin Name	PDIP/ SOIC Pin#	SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	7	7			
RB0				I/O	TTL	Bidirectional I/O pin.
INT				I	ST(1)	External interrupt pin.
RB1/SDI/SDA	7	8	8			
RB1				I/O	TTL	Bidirectional I/O pin.
SDI				I	ST	SPI™ data in.
SDA				I/O	ST	I^2C™ data.
RB2/SDO/CCP1	8	9	9			
RB2				I/O	TTL	Bidirectional I/O pin.
SDO				O	ST	SPI data out.
CCP1				I/O	ST	Capture input, Compare output, PWM output.
RB3/CCP1/PGM	9	10	10			
RB3				I/O	TTL	Bidirectional I/O pin.
CCP1				I/O	ST	Capture input, Compare output, PWM output.
PGM				I	ST	Low-Voltage ICSP™ Programming enable pin.
RB4/SCK/SCL	10	11	12			
RB4				I/O	TTL	Bidirectional I/O pin. Interrupt-on-change pin.
SCK				I/O	ST	Synchronous serial clock input/output for SPI.
SCL				I	ST	Synchronous serial clock input for I^2C.
RB5/$\overline{SS}$	11	12	13			
RB5				I/O	TTL	Bidirectional I/O pin. Interrupt-on-change pin.
$\overline{SS}$				I	TTL	Slave select for SPI in Slave mode.
RB6/T1OSO/T1CKI/PGC	12	13	15			
RB6				I/O	TTL	Interrupt-on-change pin.
T1OSO				O	ST	Timer1 Oscillator output.
T1CKI				I	ST	Timer1 clock input.
PGC				I	ST(2)	In-circuit debugger and ICSP programming clock pin.
RB7/T1OSI/PGD	13	14	16			
RB7				I/O	TTL	Interrupt-on-change pin.
T1OSI				I	ST	Timer1 oscillator input.
PGD				I	ST(2)	In-circuit debugger and ICSP programming data pin.
VSS	5	5, 6	3, 5	P	–	Ground reference for logic and I/O pins.
VDD	14	15, 16	17, 19	P	–	Positive supply for logic and I/O pins.

Legend: I = Input O = Output I/O = Input/Output P = Power
– = Not used TTL = TTL Input ST = Schmitt Trigger Input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

PIC16F87XA
Data Sheet

28/40/44-Pin Enhanced Flash Microcontrollers

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

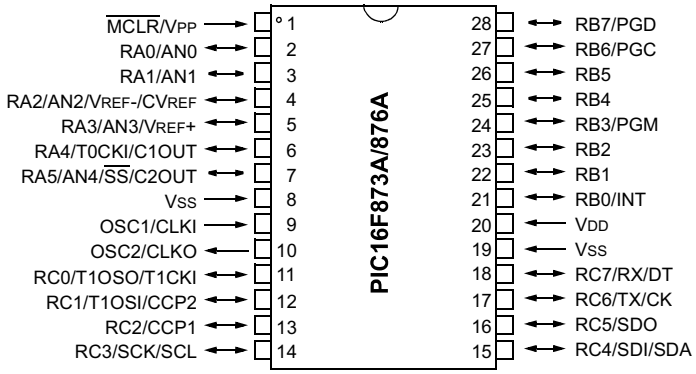
CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

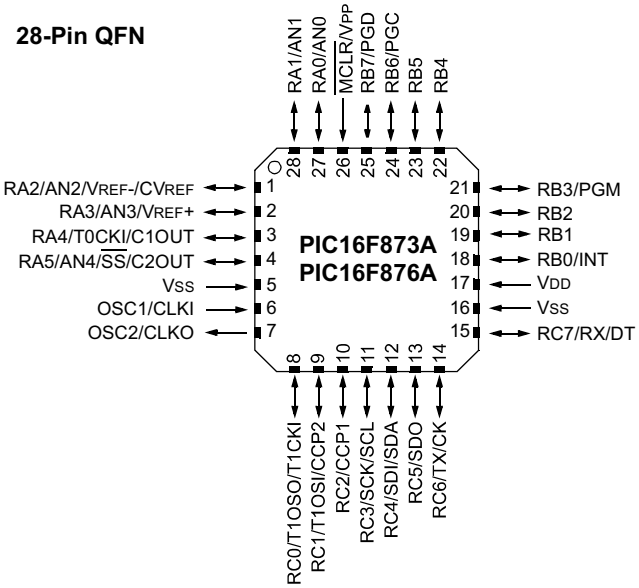
Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Pin Diagrams

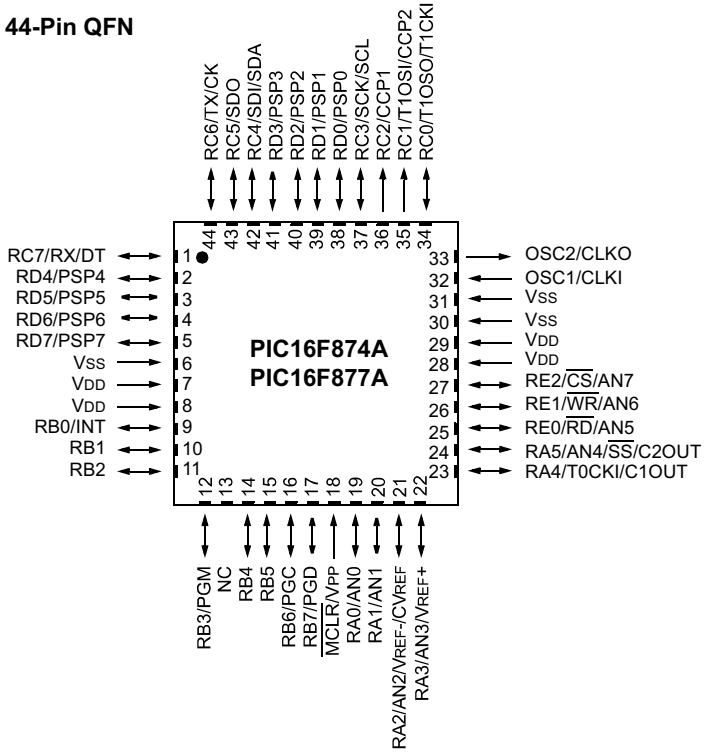
28-Pin PDIP, SOIC, SSOP



28-Pin QFN



44-Pin QFN



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 implemented only on the 40/44-pin devices

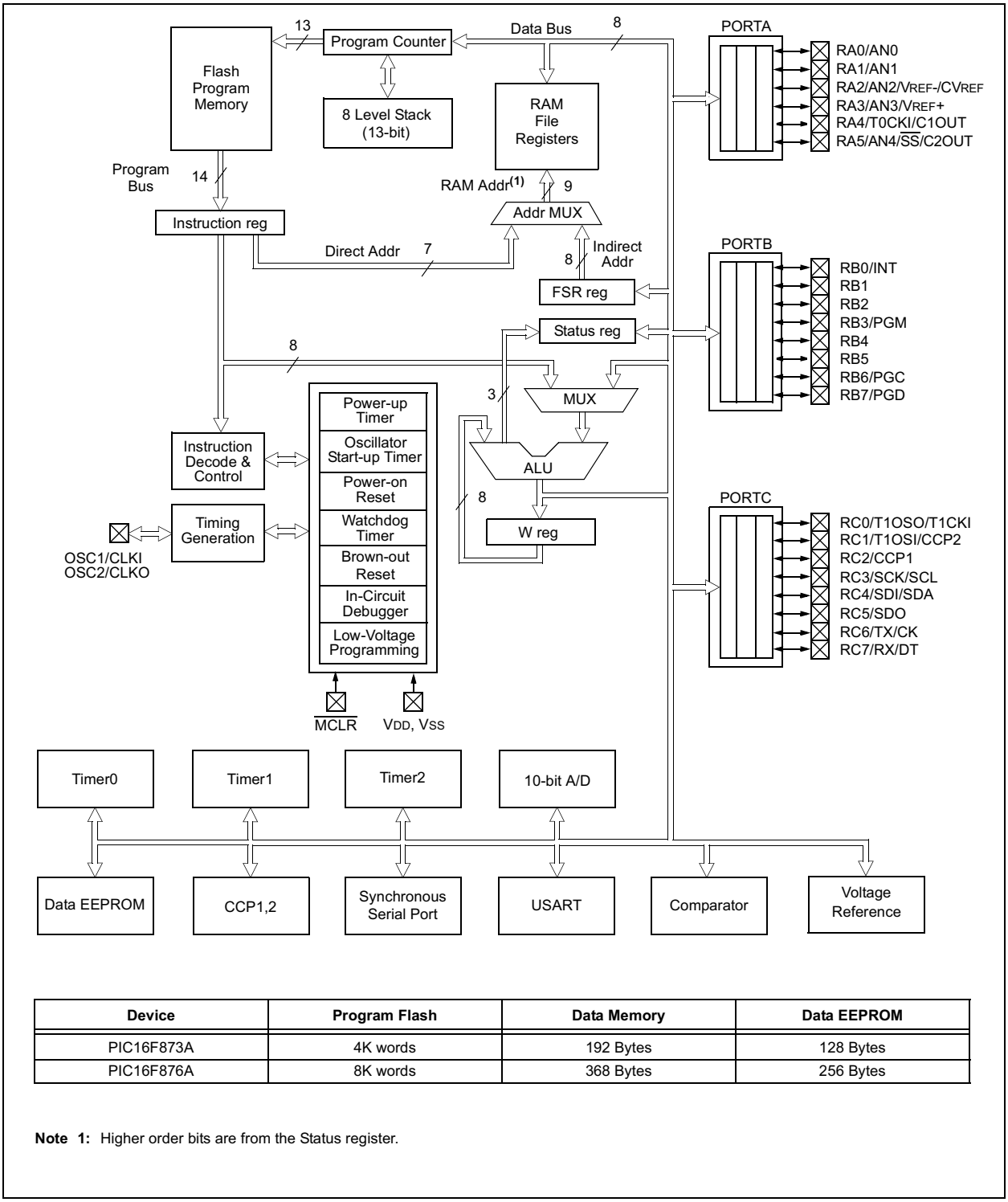
The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

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	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

FIGURE 1-1: PIC16F873A/876A BLOCK DIAGRAM



Device	Program Flash	Data Memory	Data EEPROM
PIC16F873A	4K words	192 Bytes	128 Bytes
PIC16F876A	8K words	368 Bytes	256 Bytes

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

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKI	9	6		ST/CMOS(3)	Oscillator crystal or external clock input.
OSC1			I		Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS.
CLKI			I		External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO	10	7		—	Oscillator crystal or clock output.
OSC2			O		Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O		In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	26		ST	Master Clear (input) or programming voltage (output).
$\overline{MCLR}$			I		Master Clear (Reset) input. This pin is an active low Reset to the device.
VPP			P		Programming voltage input.
					PORTA is a bidirectional I/O port.
RA0/AN0	2	27		TTL	
RA0			I/O		Digital I/O.
AN0			I		Analog input 0.
RA1/AN1	3	28		TTL	
RA1			I/O		Digital I/O.
AN1			I		Analog input 1.
RA2/AN2/VREF-/	4	1		TTL	
CVREF			I/O		Digital I/O.
RA2			I		Analog input 2.
AN2			I		A/D reference voltage (Low) input.
VREF-			O		Comparator VREF output.
CVREF					
RA3/AN3/VREF+	5	2		TTL	
RA3			I/O		Digital I/O.
AN3			I		Analog input 3.
VREF+			I		A/D reference voltage (High) input.
RA4/T0CKI/C1OUT	6	3		ST	
RA4			I/O		Digital I/O – Open-drain when configured as output.
T0CKI			I		Timer0 external clock input.
C1OUT			O		Comparator 1 output.
RA5/AN4/$\overline{SS}$/C2OUT	7	4		TTL	
RA5			I/O		Digital I/O.
AN4			I		Analog input 4.
$\overline{SS}$			I		SPI slave select input.
C2OUT			O		Comparator 2 output.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION (CONTINUED)

Pin Name	PDIP, SOIC, SSOP Pin#	QFN Pin#	I/O/P Type	Buffer Type	Description
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT	21	18		TTL/ST(1)	
RB0			I/O		Digital I/O.
INT			I		External interrupt.
RB1	22	19	I/O	TTL	Digital I/O.
RB2	23	20	I/O	TTL	Digital I/O.
RB3/PGM	24	21		TTL	
RB3			I/O		Digital I/O.
PGM			I		Low-voltage (single-supply) ICSP programming enable pin.
RB4	25	22	I/O	TTL	Digital I/O.
RB5	26	23	I/O	TTL	Digital I/O.
RB6/PGC	27	24		TTL/ST(2)	
RB6			I/O		Digital I/O.
PGC			I		In-circuit debugger and ICSP programming clock.
RB7/PGD	28	25		TTL/ST(2)	
RB7			I/O		Digital I/O.
PGD			I/O		In-circuit debugger and ICSP programming data.
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	11	8		ST	
RC0			I/O		Digital I/O.
T1OSO			O		Timer1 oscillator output.
T1CKI			I		Timer1 external clock input.
RC1/T1OSI/CCP2	12	9		ST	
RC1			I/O		Digital I/O.
T1OSI			I		Timer1 oscillator input.
CCP2			I/O		Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1	13	10		ST	
RC2			I/O		Digital I/O.
CCP1			I/O		Capture1 input, Compare1 output, PWM1 output.
RC3/SCK/SCL	14	11		ST	
RC3			I/O		Digital I/O.
SCK			I/O		Synchronous serial clock input/output for SPI mode.
SCL			I/O		Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	15	12		ST	
RC4			I/O		Digital I/O.
SDI			I		SPI data in.
SDA			I/O		I^2C data I/O.
RC5/SDO	16	13		ST	
RC5			I/O		Digital I/O.
SDO			O		SPI data out.
RC6/TX/CK	17	14		ST	
RC6			I/O		Digital I/O.
TX			O		USART asynchronous transmit.
CK			I/O		USART1 synchronous clock.
RC7/RX/DT	18	15		ST	
RC7			I/O		Digital I/O.
RX			I		USART asynchronous receive.
DT			I/O		USART synchronous data.
VSS	8, 19	5, 6	P	—	Ground reference for logic and I/O pins.
VDD	20	17	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC Oscillator mode and a CMOS input otherwise.

PIC16F882/883/884/886/887
Data Sheet

28/40/44-Pin, Enhanced Flash-Based 8-Bit CMOS Microcontrollers with nanoWatt Technology

PIC16F882/883/884/886/887

28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology

High-Performance RISC CPU:

- Only 35 instructions to learn:
  - All single-cycle instructions except branches
- Operating speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%
  - Software selectable frequency range of 8 MHz to 31 kHz
  - Software tunable
  - Two-Speed Start-up mode
  - Crystal fail detect for critical applications
  - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide operating voltage range (2.0V-5.5V)
- Industrial and Extended Temperature range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with software control option
- Enhanced low-current Watchdog Timer (WDT) with on-chip oscillator (software selectable nominal 268 seconds with full prescaler) with software enable
- Multiplexed Master Clear with pull-up/input pin
- Programmable code protection
- High Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years
- Program memory Read/Write during run time
- In-Circuit Debugger (on board)

Low-Power Features:

- Standby Current:
  - 50 nA @ 2.0V, typical
- Operating Current:
  - 11 μA @ 32 kHz, 2.0V, typical
  - 220 μA @ 4 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 μA @ 2.0V, typical

Peripheral Features:

- 24/35 I/O pins with individual direction control:
  - High current source/sink for direct LED drive
  - Interrupt-on-Change pin
  - Individually programmable weak pull-ups
  - Ultra Low-Power Wake-up (ULPWU)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (CV_{REF}) module (% of V_{DD})
  - Fixed voltage reference (0.6V)
  - Comparator inputs and outputs externally accessible
  - SR Latch mode
  - External Timer1 Gate (count enable)
- A/D Converter:
  - 10-bit resolution and 11/14 channels
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Dedicated low-power 32 kHz oscillator
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Enhanced Capture, Compare, PWM+ module:
  - 16-bit Capture, max. resolution 12.5 ns
  - Compare, max. resolution 200 ns
  - 10-bit PWM with 1, 2 or 4 output channels, programmable “dead time”, max. frequency 20 kHz
  - PWM output steering control
- Capture, Compare, PWM module:
  - 16-bit Capture, max. resolution 12.5 ns
  - 16-bit Compare, max. resolution 200 ns
  - 10-bit PWM, max. frequency 20 kHz
- Enhanced USART module:
  - Supports RS-485, RS-232, and LIN 2.0
  - Auto-Baud Detect
  - Auto-Wake-Up on Start bit
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I^2C™ Master and Slave Modes with I^2C address mask

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	ECCP/ CCP	EUSART	MSSP	Comparators	Timers 8/16-bit
	Flash (words)	SRAM (bytes)	EEPROM (bytes)							
PIC16F882	2048	128	128	28	11	1/1	1	1	2	2/1
PIC16F883	4096	256	256	24	11	1/1	1	1	2	2/1
PIC16F884	4096	256	256	35	14	1/1	1	1	2	2/1
PIC16F886	8192	368	256	24	11	1/1	1	1	2	2/1
PIC16F887	8192	368	256	35	14	1/1	1	1	2	2/1

Pin Diagrams – PIC16F884/887, 44-Pin TQFP

44-pin TQFP

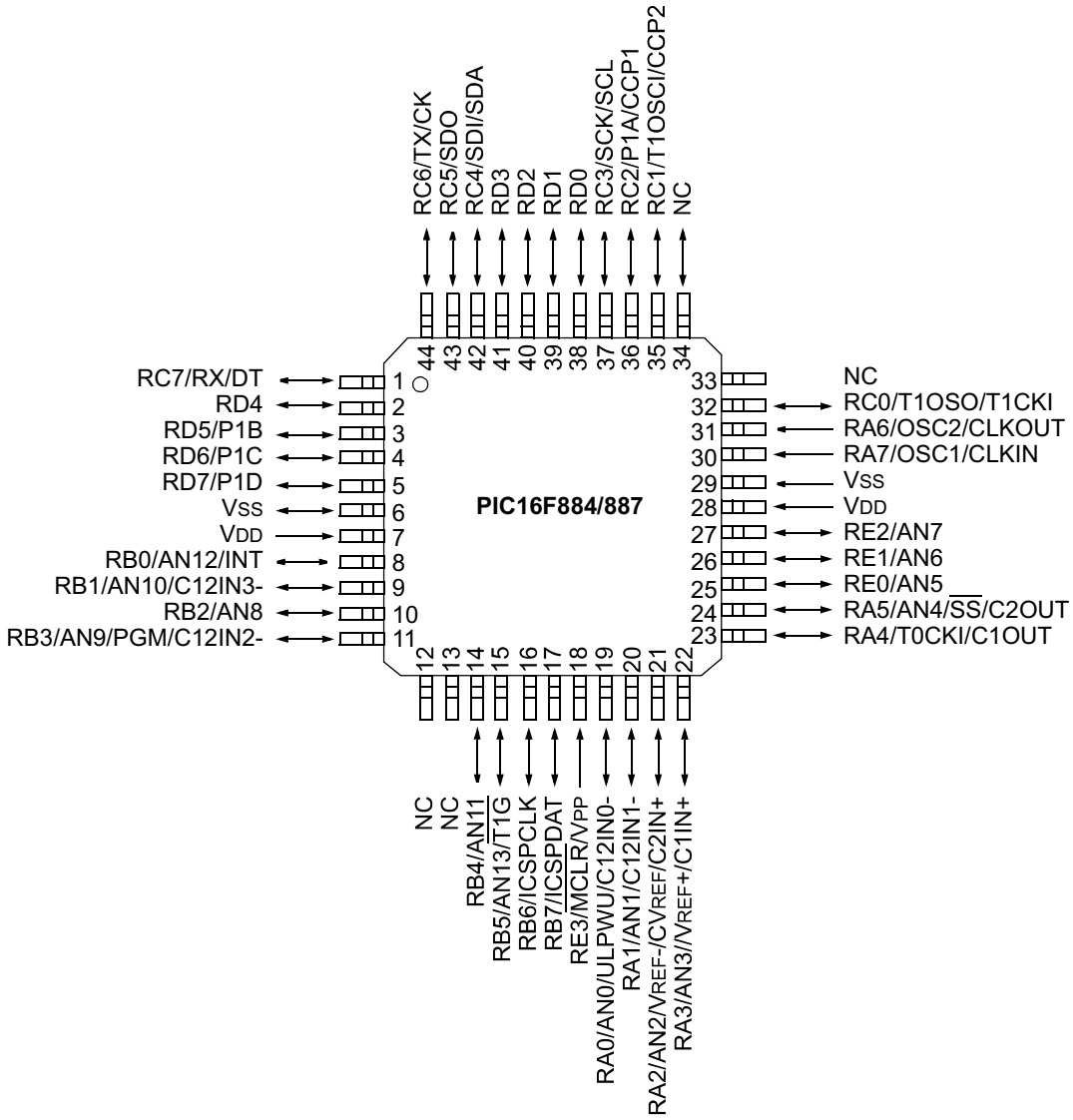


TABLE 5: PIC16F884/887 44-PIN SUMMARY (TQFP)

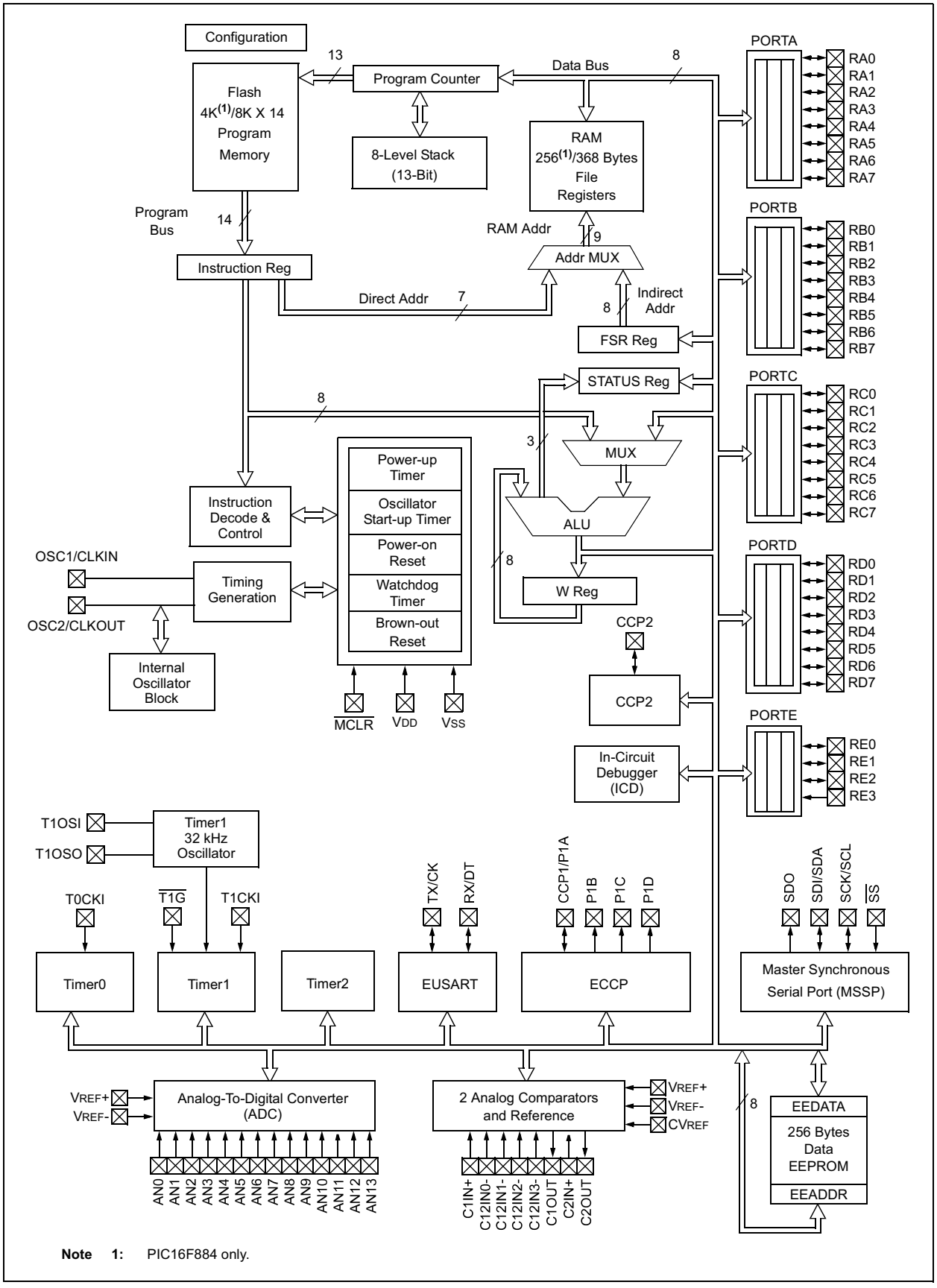
I/O	Pin	Analog	Comparators	Timers	ECCP	EUSART	MSSP	Interrupt	Pull-up	Basic
RA0	19	AN0/ULPWU	C12IN0-	—	—	—	—	—	—	—
RA1	20	AN1	C12IN1-	—	—	—	—	—	—	—
RA2	21	AN2	C2IN+	—	—	—	—	—	—	VREF-/CVREF
RA3	22	AN3	C1IN+	—	—	—	—	—	—	VREF+
RA4	23	—	C1OUT	T0CKI	—	—	—	—	—	—
RA5	24	AN4	C2OUT	—	—	—	$\overline{SS}$	—	—	—
RA6	31	—	—	—	—	—	—	—	—	OSC2/CLKOUT
RA7	31	—	—	—	—	—	—	—	—	OSC1/CLKIN
RB0	8	AN12	—	—	—	—	—	IOC/INT	Y	—
RB1	9	AN10	C12IN3-	—	—	—	—	IOC	Y	—
RB2	10	AN8	—	—	—	—	—	IOC	Y	—
RB3	11	AN9	C12IN2-	—	—	—	—	IOC	Y	PGM
RB4	14	AN11	—	—	—	—	—	IOC	Y	—
RB5	15	AN13	—	$\overline{T1G}$	—	—	—	IOC	Y	—
RB6	16	—	—	—	—	—	—	IOC	Y	ICSPCLK
RB7	17	—	—	—	—	—	—	IOC	Y	ICSPDAT
RC0	32	—	—	T1OSO/T1CKI	—	—	—	—	—	—
RC1	35	—	—	T1OSI	CCP2	—	—	—	—	—
RC2	36	—	—	—	CCP1/P1A	—	—	—	—	—
RC3	37	—	—	—	—	—	SCK/SCL	—	—	—
RC4	42	—	—	—	—	—	SDI/SDA	—	—	—
RC5	43	—	—	—	—	—	SDO	—	—	—
RC6	44	—	—	—	—	TX/CK	—	—	—	—
RC7	1	—	—	—	—	RX/DT	—	—	—	—
RD0	38	—	—	—	—	—	—	—	—	—
RD1	39	—	—	—	—	—	—	—	—	—
RD2	40	—	—	—	—	—	—	—	—	—
RD3	41	—	—	—	—	—	—	—	—	—
RD4	2	—	—	—	—	—	—	—	—	—
RD5	3	—	—	—	P1B	—	—	—	—	—
RD6	4	—	—	—	P1C	—	—	—	—	—
RD7	5	—	—	—	P1D	—	—	—	—	—
RE0	25	AN5	—	—	—	—	—	—	—	—
RE1	26	AN6	—	—	—	—	—	—	—	—
RE2	27	AN7	—	—	—	—	—	—	—	—
RE3	18	—	—	—	—	—	—	—	Y(1)	$\overline{MCLR}$/VPP
—	7	—	—	—	—	—	—	—	—	VDD
—	28	—	—	—	—	—	—	—	—	VDD
—	6	—	—	—	—	—	—	—	—	VSS
—	13	—	—	—	—	—	—	—	—	NC (no connect)
—	29	—	—	—	—	—	—	—	—	VSS
—	34	—	—	—	—	—	—	—	—	NC (no connect)
—	33	—	—	—	—	—	—	—	—	NC (no connect)
—	12	—	—	—	—	—	—	—	—	NC (no connect)

Note 1: Pull-up activated only with external $\overline{MCLR}$ configuration.

1.0 DEVICE OVERVIEW

The PIC16F882/883/884/886/887 is covered by this data sheet. The PIC16F882/883/886 is available in 28-pin PDIP, SOIC, SSOP and QFN packages. The PIC16F884/887 is available in a 40-pin PDIP and 44-pin QFN and TQFP packages. Figure 1-1 shows the block diagram of PIC16F882/883/886 and Figure 1-2 shows a block diagram of the PIC16F884/887 device. Table 1-1 and Table 1-2 show the corresponding pinout descriptions.

FIGURE 1-2: PIC16F884/PIC16F887 BLOCK DIAGRAM



Note 1: PIC16F884 only.

TABLE 1-1: PIC16F882/883/886 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0/ULPWU/C12IN0-	RA0	TTL	CMOS	General purpose I/O.
	AN0	AN	—	A/D Channel 0 input.
	ULPWU	AN	—	Ultra Low-Power Wake-up input.
	C12IN0-	AN	—	Comparator C1 or C2 negative input.
RA1/AN1/C12IN1-	RA1	TTL	CMOS	General purpose I/O. Individually enabled pull-up.
	AN1	AN	—	A/D Channel 1 input.
	C12IN1-	AN	—	Comparator C1 or C2 negative input.
RA2/AN2/VREF-/CVREF/C2IN+	RA2	TTL	CMOS	General purpose I/O.
	AN2	AN	—	A/D Channel 2.
	VREF-	AN	—	A/D Negative Voltage Reference input.
	CVREF	—	AN	Comparator Voltage Reference output.
	C2IN+	AN	—	Comparator C2 positive input.
RA3/AN3/VREF+/C1IN+	RA3	TTL	—	General purpose I/O.
	AN3	AN	—	A/D Channel 3.
	VREF+	AN	—	Programming voltage.
	C1IN+	AN	—	Comparator C1 positive input.
RA4/T0CKI/C1OUT	RA4	TTL	CMOS	General purpose I/O. Individually enabled pull-up.
	T0CKI	ST	—	Timer0 clock input.
	C1OUT	—	CMOS	Comparator C1 output.
RA5/AN4/$\overline{SS}$/C2OUT	RA5	TTL	CMOS	General purpose I/O.
	AN4	AN	—	A/D Channel 4.
	$\overline{SS}$	ST	—	Slave Select input.
	C2OUT	—	CMOS	Comparator C2 output.
RA6/OSC2/CLKOUT	RA6	TTL	CMOS	General purpose I/O.
	OSC2	—	XTAL	Master Clear with internal pull-up.
	CLKOUT	—	CMOS	FOSC/4 output.
RA7/OSC1/CLKIN	RA7	TTL	CMOS	General purpose I/O.
	OSC1	XTAL	—	Crystal/Resonator.
	CLKIN	ST	—	External clock input/RC oscillator connection.
RB0/AN12/INT	RB0	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN12	AN	—	A/D Channel 12.
	INT	ST	—	External interrupt.
RB1/AN10/P1C/C12IN3-	RB1	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN10	AN	—	A/D Channel 10.
	P1C	—	CMOS	PWM output.
	C12IN3-	AN	—	Comparator C1 or C2 negative input.
RB2/AN8/P1B	RB2	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN8	AN	—	A/D Channel 8.
	P1B	—	CMOS	PWM output.

Legend: AN = Analog input or output; CMOS = CMOS compatible input or output; OD = Open Drain
TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels
HV = High Voltage; XTAL = Crystal

TABLE 1-2: PIC16F884/887 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0/ULPWU/C12IN0-	RA0	TTL	CMOS	General purpose I/O.
	AN0	AN	—	A/D Channel 0 input.
	ULPWU	AN	—	Ultra Low-Power Wake-up input.
	C12IN0-	AN	—	Comparator C1 or C2 negative input.
RA1/AN1/C12IN1-	RA1	TTL	CMOS	General purpose I/O.
	AN1	AN	—	A/D Channel 1 input.
	C12IN1-	AN	—	Comparator C1 or C2 negative input.
RA2/AN2/VREF-/CVREF/C2IN+	RA2	TTL	CMOS	General purpose I/O.
	AN2	AN	—	A/D Channel 2.
	VREF-	AN	—	A/D Negative Voltage Reference input.
	CVREF	—	AN	Comparator Voltage Reference output.
	C2IN+	AN	—	Comparator C2 positive input.
RA3/AN3/VREF+/C1IN+	RA3	TTL	CMOS	General purpose I/O.
	AN3	AN	—	A/D Channel 3.
	VREF+	AN	—	A/D Positive Voltage Reference input.
	C1IN+	AN	—	Comparator C1 positive input.
RA4/T0CKI/C1OUT	RA4	TTL	CMOS	General purpose I/O.
	T0CKI	ST	—	Timer0 clock input.
	C1OUT	—	CMOS	Comparator C1 output.
RA5/AN4/$\overline{SS}$/C2OUT	RA5	TTL	CMOS	General purpose I/O.
	AN4	AN	—	A/D Channel 4.
	$\overline{SS}$	ST	—	Slave Select input.
	C2OUT	—	CMOS	Comparator C2 output.
RA6/OSC2/CLKOUT	RA6	TTL	CMOS	General purpose I/O.
	OSC2	—	XTAL	Crystal/Resonator.
	CLKOUT	—	CMOS	FOSC/4 output.
RA7/OSC1/CLKIN	RA7	TTL	CMOS	General purpose I/O.
	OSC1	XTAL	—	Crystal/Resonator.
	CLKIN	ST	—	External clock input/RC oscillator connection.
RB0/AN12/INT	RB0	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN12	AN	—	A/D Channel 12.
	INT	ST	—	External interrupt.
RB1/AN10/C12IN3-	RB1	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN10	AN	—	A/D Channel 10.
	C12IN3-	AN	—	Comparator C1 or C2 negative input.
RB2/AN8	RB2	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN8	AN	—	A/D Channel 8.
RB3/AN9/PGM/C12IN2-	RB3	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN9	AN	—	A/D Channel 9.
	PGM	ST	—	Low-voltage ICSP™ Programming enable pin.
	C12IN2-	AN	—	Comparator C1 or C2 negative input.

Legend: AN = Analog input or output CMOS = CMOS compatible input or output OD = Open Drain
TTL = TTL compatible input ST = Schmitt Trigger input with CMOS levels
HV = High Voltage XTAL = Crystal

TABLE 1-2: PIC16F884/887 PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RB4/AN11	RB4	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN11	AN	—	A/D Channel 11.
RB5/AN13/$\overline{T1G}$	RB5	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN13	AN	—	A/D Channel 13.
	$\overline{T1G}$	ST	—	Timer1 Gate input.
RB6/ICSPCLK	RB6	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	ICSPCLK	ST	—	Serial Programming Clock.
RB7/ICSPDAT	RB7	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	ICSPDAT	ST	TTL	ICSP™ Data I/O.
RC0/T1OSO/T1CKI	RC0	ST	CMOS	General purpose I/O.
	T1OSO	—	XTAL	Timer1 oscillator output.
	T1CKI	ST	—	Timer1 clock input.
RC1/T1OSI/CCP2	RC1	ST	CMOS	General purpose I/O.
	T1OSI	XTAL	—	Timer1 oscillator input.
	CCP2	ST	CMOS	Capture/Compare/PWM2.
RC2/P1A/CCP1	RC2	ST	CMOS	General purpose I/O.
	P1A	ST	CMOS	PWM output.
	CCP1	—	CMOS	Capture/Compare/PWM1.
RC3/SCK/SCL	RC3	ST	CMOS	General purpose I/O.
	SCK	ST	CMOS	SPI clock.
	SCL	ST	OD	I^2C™ clock.
RC4/SDI/SDA	RC4	ST	CMOS	General purpose I/O.
	SDI	ST	—	SPI data input.
	SDA	ST	OD	I^2C data input/output.
RC5/SDO	RC5	ST	CMOS	General purpose I/O.
	SDO	—	CMOS	SPI data output.
RC6/TX/CK	RC6	ST	CMOS	General purpose I/O.
	TX	—	CMOS	EUSART asynchronous transmit.
	CK	ST	CMOS	EUSART synchronous clock.
RC7/RX/DT	RC7	ST	CMOS	General purpose I/O.
	RX	ST	—	EUSART asynchronous input.
	DT	ST	CMOS	EUSART synchronous data.
RD0	RD0	TTL	CMOS	General purpose I/O.
RD1	RD1	TTL	CMOS	General purpose I/O.
RD2	RD2	TTL	CMOS	General purpose I/O.
RD3	RD3	TTL	CMOS	General purpose I/O.
RD4	RD4	TTL	CMOS	General purpose I/O.
RD5/P1B	RD5	TTL	CMOS	General purpose I/O.
	P1B	—	CMOS	PWM output.
RD6/P1C	RD6	TTL	CMOS	General purpose I/O.
	P1C	—	CMOS	PWM output.

Legend: AN = Analog input or output; CMOS = CMOS compatible input or output; OD = Open Drain
TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels
HV = High Voltage; XTAL = Crystal

TABLE 1-2: PIC16F884/887 PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RD7/P1D	RD7	TTL	CMOS	General purpose I/O.
	P1D	AN	—	PWM output.
RE0/AN5	RE0	TTL	CMOS	General purpose I/O.
	AN5	AN	—	A/D Channel 5.
RE1/AN6	RE1	ST	CMOS	General purpose I/O.
	AN6	AN	—	A/D Channel 6.
RE2/AN7	RE2	TTL	CMOS	General purpose I/O.
	AN7	AN	—	A/D Channel 7.
RE3/$\overline{MCLR}$/V_{PP}	RE3	TTL	—	General purpose input.
	$\overline{MCLR}$	ST	—	Master Clear with internal pull-up.
	V_{PP}	HV	—	Programming voltage.
V_{SS}	V_{SS}	Power	—	Ground reference.
V_{DD}	V_{DD}	Power	—	Positive supply.

Legend: AN = Analog input or output CMOS = CMOS compatible input or output OD = Open Drain
TTL = TTL compatible input ST = Schmitt Trigger input with CMOS levels
HV = High Voltage XTAL = Crystal

PIC18F2455/2550/4455/4550
Data Sheet

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1 Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Idle mode Currents Down to 5.8 μA Typical
- Sleep mode Currents Down to 0.1 μA Typical
- Timer1 Oscillator: 1.1 μA Typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA Typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High-Precision PLL for USB
- Two External Clock modes, Up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator Options allow Microcontroller and USB module to Run at Different Clock Speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 10-Bit, Up to 13-Channel Analog-to-Digital Converter (A/D) module with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

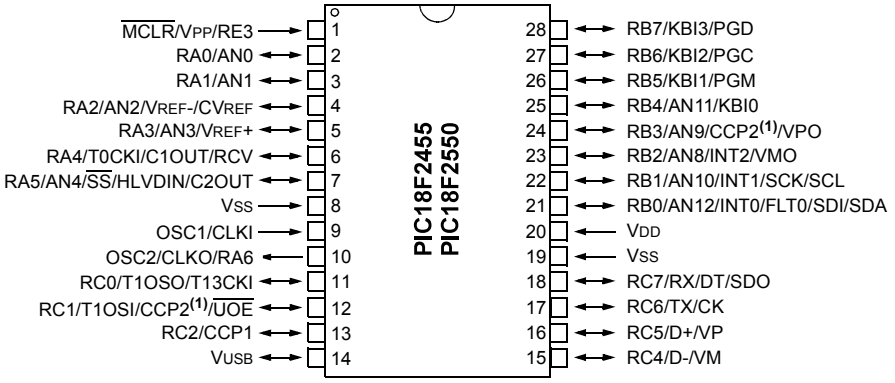
Special Microcontroller Features:

- C Compiler Optimized Architecture with Optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory Typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory Typical
- Flash/Data EEPROM Retention: > 40 Years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Optional Dedicated ICD/ICSP Port (44-pin, TQFP package only)
- Wide Operating Voltage Range (2.0V to 5.5V)

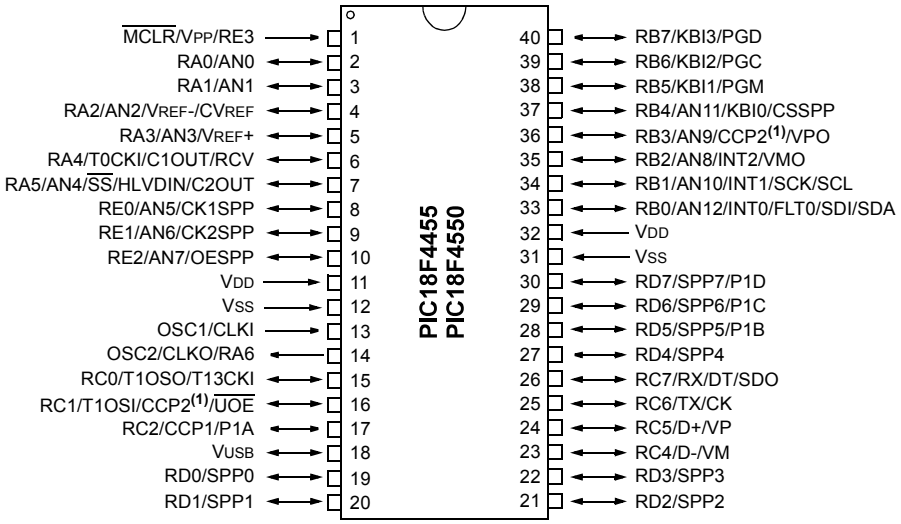
Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC18F2455
- PIC18F2550
- PIC18F4455
- PIC18F4550
- PIC18LF2455
- PIC18LF2550
- PIC18LF4455
- PIC18LF4550

This family of devices offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Enhanced Flash program memory. In addition to these features, the PIC18F2455/2550/4455/4550 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2455/2550/4455/4550 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4%, of normal operation requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 28.0 "Electrical Characteristics"** for values.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18F2455/2550/4455/4550 family incorporate a fully featured Universal Serial Bus communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full-speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators.

1.1.3 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2455/2550/4455/4550 family offer twelve different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes using crystals or ceramic resonators.
- Four External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O).
- An internal oscillator block which provides an 8 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and V_{DD}), as well as a range of 6 user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the High-Speed Crystal and External Oscillator modes, which allows a wide range of clock speeds from 4 MHz to 48 MHz.
- Asynchronous dual clock operation, allowing the USB module to run from a high-frequency oscillator while the rest of the microcontroller is clocked from an internal low-power oscillator.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-Programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine, located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2455/2550/4455/4550 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Literal Offset Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages such as C.
- **Enhanced CCP Module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown for disabling PWM outputs on interrupt or other select conditions, and auto-restart to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. The TX/CK and RX/DT signals can be inverted, eliminating the need for inverting buffers. Other enhancements include Automatic Baud Rate Detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated, without waiting for a sampling period and thus, reducing code overhead.
- **Dedicated ICD/ICSP Port:** These devices introduce the use of debugger and programming pins that are not multiplexed with other micro-controller features. Offered as an option in select packages, this feature allows users to develop I/O intensive applications while retaining the ability to program and debug in the circuit.

1.3 Details on Individual Family Members

Devices in the PIC18F2455/2550/4455/4550 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in six ways:

1. Flash program memory (24 Kbytes for PIC18FX455 devices, 32 Kbytes for PIC18FX550 devices).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports and 1 input only port on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have two standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Streaming Parallel Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

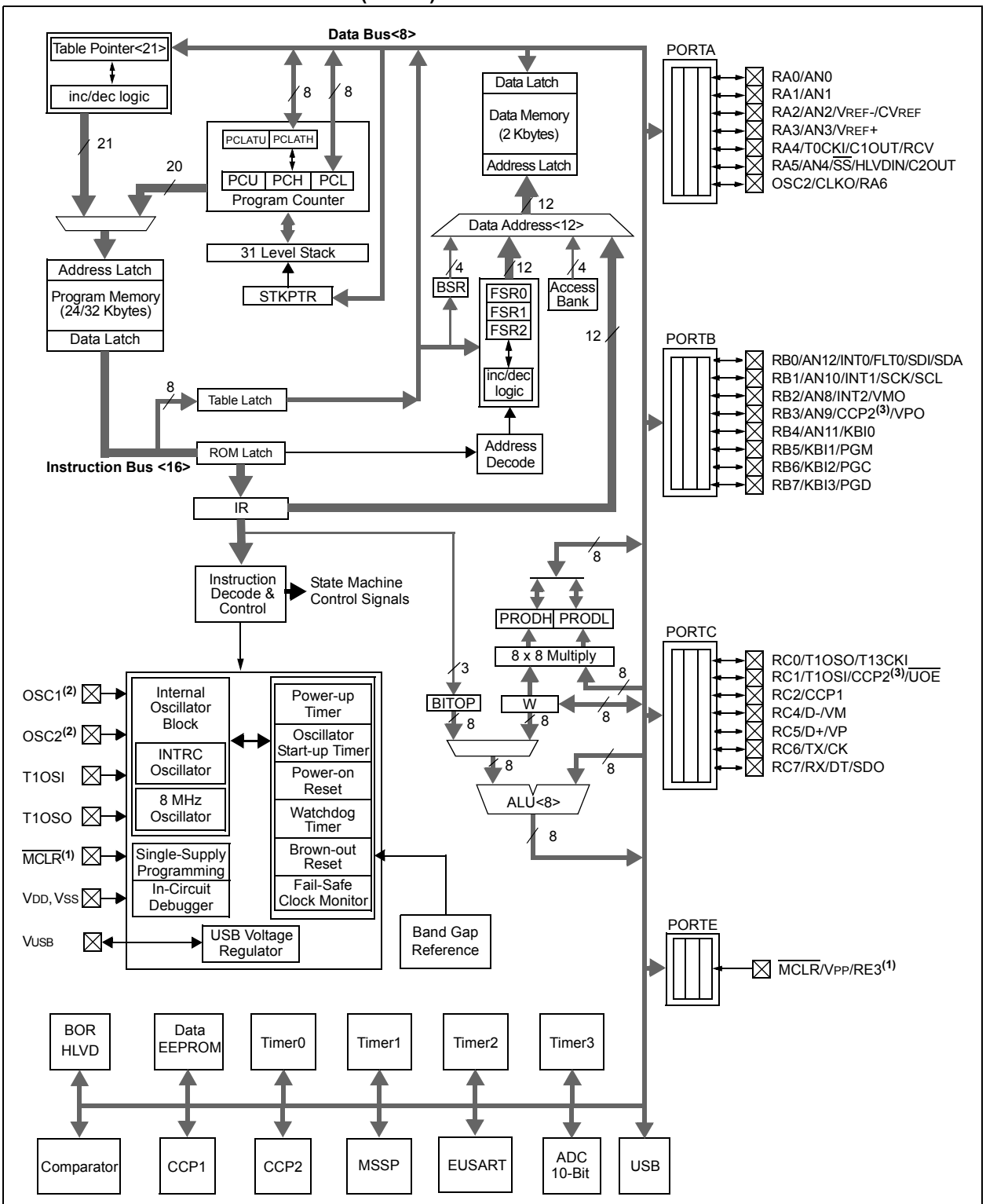
The pinouts for all devices are listed in Table 1-2 and Table 1-3.

Like all Microchip PIC18 devices, members of the PIC18F2455/2550/4455/4550 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an “F” in the part number (such as PIC18**F**2550), accommodate an operating V_{DD} range of 4.2V to 5.5V. Low-voltage parts, designated by “LF” (such as PIC18**LF**2550), function over an extended V_{DD} range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-Pin PDIP 28-Pin SOIC	28-Pin PDIP 28-Pin SOIC	40-Pin PDIP 44-Pin QFN 44-Pin TQFP	40-Pin PDIP 44-Pin QFN 44-Pin TQFP

FIGURE 1-1: PIC18F2455/2550 (28-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

3: RB3 is the alternate pin for CCP2 multiplexing.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
$\overline{\text{MCLR}}$/VPP/RE3	1			Master Clear (input) or programming voltage (input).
$\overline{\text{MCLR}}$		I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP		P		Programming voltage input.
RE3		I	ST	Digital input.
OSC1/CLKI	9			Oscillator crystal or external clock input.
OSC1		I	Analog	Oscillator crystal input or external clock source input.
CLKI		I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pin.)
OSC2/CLKO/RA6	10			Oscillator crystal or clock output.
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In select modes, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
				PORTA is a bidirectional I/O port.
RA0/AN0	2			
RA0		I/O	TTL	Digital I/O.
AN0		I	Analog	Analog input 0.
RA1/AN1	3			
RA1		I/O	TTL	Digital I/O.
AN1		I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4			
RA2		I/O	TTL	Digital I/O.
AN2		I	Analog	Analog input 2.
VREF-		I	Analog	A/D reference voltage (low) input.
CVREF		O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5			
RA3		I/O	TTL	Digital I/O.
AN3		I	Analog	Analog input 3.
VREF+		I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6			
RA4		I/O	ST	Digital I/O.
T0CKI		I	ST	Timer0 external clock input.
C1OUT		O	—	Comparator 1 output.
RCV		I	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{SS}$/ HLVDIN/C2OUT	7			
RA5		I/O	TTL	Digital I/O.
AN4		I	Analog	Analog input 4.
$\overline{SS}$		I	TTL	SPI slave select input.
HLVDIN		I	Analog	High/Low-Voltage Detect input.
C2OUT		O	—	Comparator 2 output.
RA6	—	—	—	See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
				PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/FLT0/ SDI/SDA	21			
RB0		I/O	TTL	Digital I/O.
AN12		I	Analog	Analog input 12.
INT0		I	ST	External interrupt 0.
FLT0		I	ST	PWM Fault input (CCP1 module).
SDI		I	ST	SPI data in.
SDA		I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/ SCL	22			
RB1		I/O	TTL	Digital I/O.
AN10		I	Analog	Analog input 10.
INT1		I	ST	External interrupt 1.
SCK		I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL		I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	23			
RB2		I/O	TTL	Digital I/O.
AN8		I	Analog	Analog input 8.
INT2		I	ST	External interrupt 2.
VMO		O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	24			
RB3		I/O	TTL	Digital I/O.
AN9		I	Analog	Analog input 9.
CCP2(1)		I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
VPO		O	—	External USB transceiver VPO output.
RB4/AN11/KBI0	25			
RB4		I/O	TTL	Digital I/O.
AN11		I	Analog	Analog input 11.
KBI0		I	TTL	Interrupt-on-change pin.
RB5/KBI1/PGM	26			
RB5		I/O	TTL	Digital I/O.
KBI1		I	TTL	Interrupt-on-change pin.
PGM		I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	27			
RB6		I/O	TTL	Digital I/O.
KBI2		I	TTL	Interrupt-on-change pin.
PGC		I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	28			
RB7		I/O	TTL	Digital I/O.
KBI3		I	TTL	Interrupt-on-change pin.
PGD		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
				PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	11			
RC0		I/O	ST	Digital I/O.
T1OSO		O	—	Timer1 oscillator output.
T13CKI		I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2/$\overline{UOE}$	12			
RC1		I/O	ST	Digital I/O.
T1OSI		I	CMOS	Timer1 oscillator input.
CCP2**(2)**		I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
$\overline{UOE}$		O	—	External USB transceiver $\overline{OE}$ output.
RC2/CCP1	13			
RC2		I/O	ST	Digital I/O.
CCP1		I/O	ST	Capture 1 input/Compare 1 output/PWM1 output.
RC4/D-/VM	15			
RC4		I	TTL	Digital input.
D-		I/O	—	USB differential minus line (input/output).
VM		I	TTL	External USB transceiver VM input.
RC5/D+/VP	16			
RC5		I	TTL	Digital input.
D+		I/O	—	USB differential plus line (input/output).
VP		O	TTL	External USB transceiver VP input.
RC6/TX/CK	17			
RC6		I/O	ST	Digital I/O.
TX		O	—	EUSART asynchronous transmit.
CK		I/O	ST	EUSART synchronous clock (see RX/DT).
RC7/RX/DT/SDO	18			
RC7		I/O	ST	Digital I/O.
RX		I	ST	EUSART asynchronous receive.
DT		I/O	ST	EUSART synchronous data (see TX/CK).
SDO		O	—	SPI data out.
RE3	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VUSB	14	P	—	Internal USB 3.3V voltage regulator output, positive supply for internal USB transceiver.
VSS	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

MCP1702

2 μA Low Dropout Positive Voltage Regulator Product Brief

Features:

- 2.0 μA Typical Quiescent Current
- Input Operating Voltage Range: 2.7V to 13.2V
- Low Dropout Voltage:
  - 650 mV (typ.) @ 250 mA (V_{OUT} = 2.5V)
- 250 mA Output Current for Output Voltages ≥ 2.5V
- 200 mA Output Current for Output Voltages < 2.5V
- High-Accuracy Output Voltage: ±2% (max.)
- Low Temperature Drift: ±100 ppm/°C (typ.)
- Excellent Line Regulation: 0.2%/V (typ.)
- Package Options: 3-Pin SOT-23A, 3-Pin SOT-89, and 3-Pin TO-92
- Short Circuit Protection and Thermal Shutdown Protection
- Stable with 1.0 μF to 22 μF Output Capacitance
- Standard Output Voltage Options:
  - 1.20V, 1.5V, 1.8V, 2.5V, 2.8V, 3.0V, 3.3V, 4.0V, 5.0V

Applications:

- Battery-Powered Devices
- Battery-Powered Alarm Circuits
- Smoke Detectors
- CO^2 Detectors
- Smart Battery Packs
- PDAs
- Low Quiescent Current Voltage Reference
- Cameras and Portable Video Equipment
- Pagers and Cellular Phones
- Solar-Powered Instruments
- Consumer Products
- Microcontroller Power
- Battery Powered Data Loggers

Related Literature:

- AN765, "*Using Microchip's Micropower LDOs*", DS00765, Microchip Technology Inc., 2002
- AN766, "*Pin-Compatible CMOS Upgrades to Bipolar LDOs*", DS00766, Microchip Technology Inc., 2002

General Description:

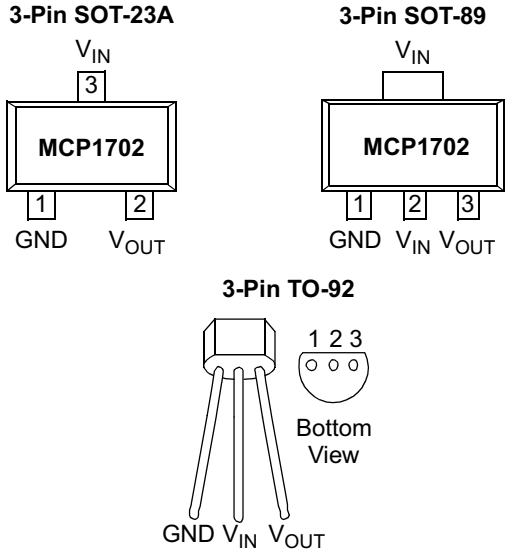
The MCP1702 is a family of CMOS low dropout (LDO), positive voltage regulators that can deliver up to 250 mA of current while consuming only 2.0 μA of quiescent current (typ.). The input operating voltage range is specified up to 13.2V, making it ideal for lithium-ion (one, two or three cells), 9V alkaline and other three to six primary cell battery-powered applications.

The MCP1702 is capable of delivering 250 mA with an input-to-output voltage differential (dropout voltage) of 650 mV. The low dropout voltage extends the battery operating lifetime. It also permits high currents in small packages when operated with minimum $V_{IN} - V_{OUT}$ differentials.

The MCP1702 has a tight tolerance output voltage regulation of ±0.5% (typ.) and very good line regulation at ±0.2%/V. The LDO output is stable when using only 1 μF of output capacitance of either ceramic, tantalum or aluminum-electrolytic style capacitors. The MCP1702 LDO also incorporates short circuit and thermal shutdown protection to ensure maximum reliability.

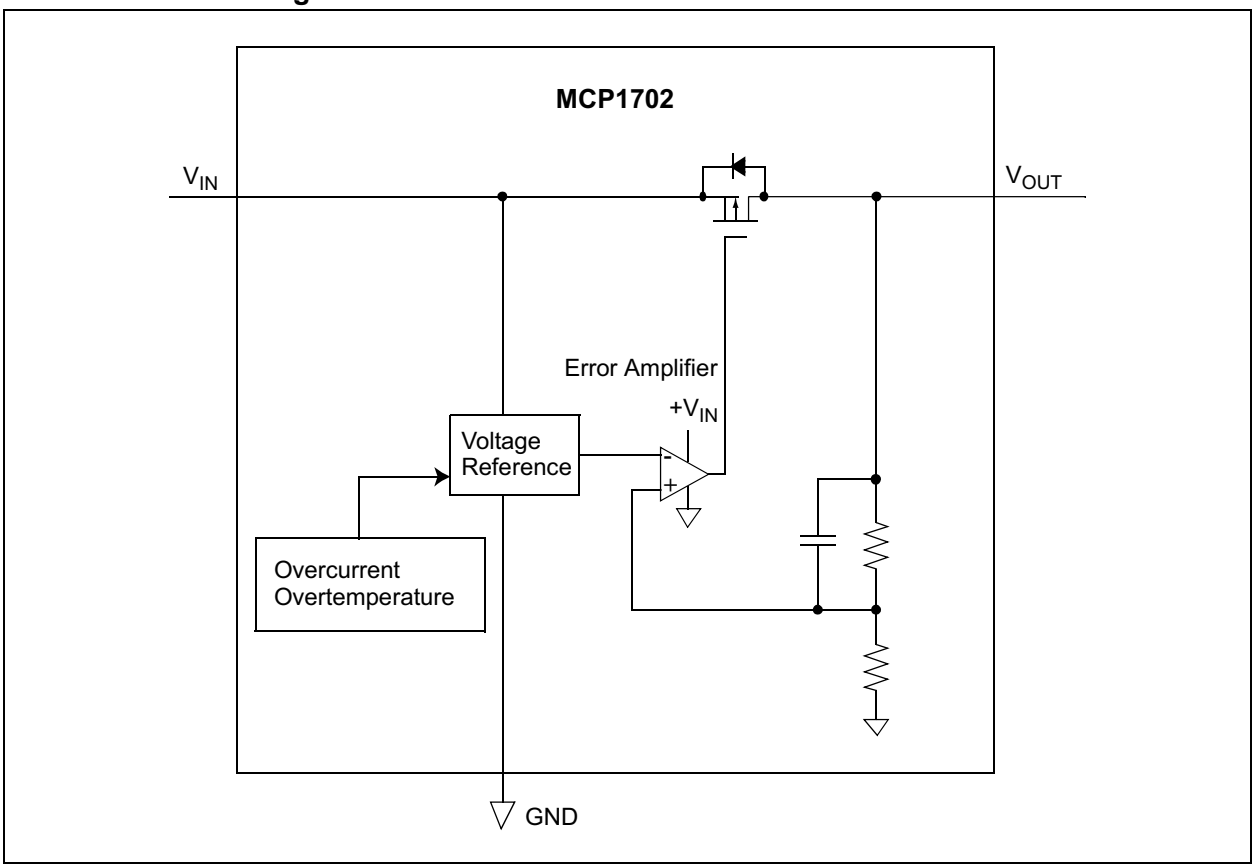
The MCP1702 device is offered in the 3-pin SOT-23A, 3-pin SOT-89, and 3-Pin TO-92 package options with a temperature range of -40°C to +125°C.

Package Types

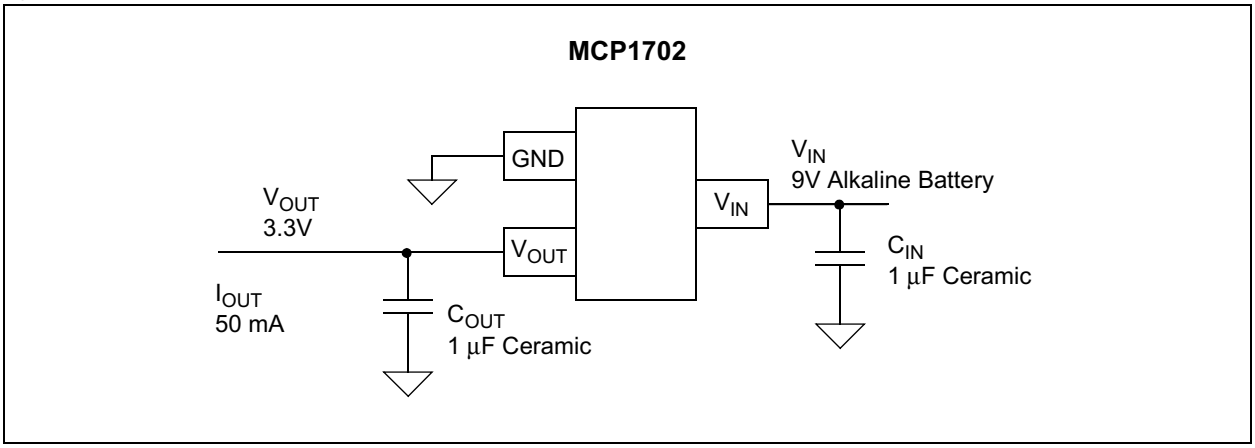


Note: 3-Pin SOT-23A is equivalent to the EIAJ SC-59.

Functional Block Diagram



Typical Application Circuits



MCP3202

2.7V Dual Channel 12-Bit A/D Converter with SPI Serial Interface

Features

- 12-bit resolution
- ±1 LSB max DNL
- ±1 LSB max INL (MCP3202-B)
- ±2 LSB max INL (MCP3202-C)
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V-5.5V
- 100 ksps max. sampling rate at V_{DD} = 5V
- 50 ksps max. sampling rate at V_{DD} = 2.7V
- Low power CMOS technology
  - 500 nA typical standby current, 5 µA max.
  - 550 µA max. active current at 5V
- Industrial temp range: -40°C to +85°C
- 8-pin MSOP, PDIP, SOIC and TSSOP packages

Applications

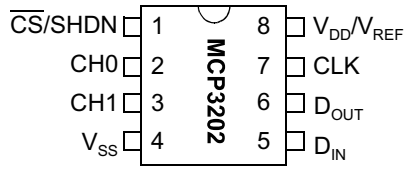
- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

Description

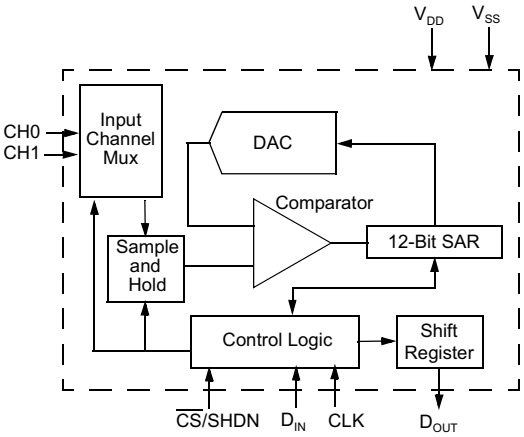
The Microchip Technology Inc. MCP3202 is a successive approximation 12-bit Analog-to-Digital (A/D) Converter with on-board sample and hold circuitry. The MCP3202 is programmable to provide a single pseudo-differential input pair or dual single-ended inputs. Differential Nonlinearity (DNL) is specified at ±1 LSB, and Integral Nonlinearity (INL) is offered in ±1 LSB (MCP3202-B) and ±2 LSB (MCP3202-C) versions. Communication with the device is done using a simple serial interface compatible with the SPI protocol. The device is capable of conversion rates of up to 100 ksps at 5V and 50 ksps at 2.7V. The MCP3202 device operates over a broad voltage range (2.7V-5.5V). Low-current design permits operation with typical standby and active currents of only 500 nA and 375 µA, respectively. The MCP3202 is offered in 8-pin MSOP, PDIP, TSSOP and 150 mil SOIC packages.

Package Types

PDIP, MSOP, SOIC, TSSOP



Functional Block Diagram



1.0 ELECTRICAL CHARACTERISTICS

1.1 Maximum Ratings*

V_{DD}.....7.0V

All inputs and outputs w.r.t. V_{SS} ..... -0.6V to V_{DD} +0.6V

Storage temperature .....-65°C to +150°C

Ambient temp. with power applied .....-65°C to +125°C

ESD protection on all pins (HBM)..... > 4 kV

***Notice:** Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIN FUNCTION TABLE

Name	Function
V_{DD}/V_{REF}	+2.7V to 5.5V Power Supply and Reference Voltage Input
CH0	Channel 0 Analog Input
CH1	Channel 1 Analog Input
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
$\overline{CS}$/SHDN	$\overline{Chip\ Select}$/Shutdown Input

ELECTRICAL CHARACTERISTICS

All parameters apply at V_{DD} = 5.5V, V_{SS} = 0V, T_{AMB} = -40°C to +85°C, f_{SAMPLE} = 100 ksps and f_{CLK} = 18*f_{SAMPLE} unless otherwise noted.

Parameter	Sym	Min.	Typ.	Max.	Units	Conditions
Conversion Rate:						
Conversion Time	t_{CONV}	—	—	12	clock cycles	
Analog Input Sample Time	t_{SAMPLE}		1.5		clock cycles	
Throughput Rate	f_{SAMPLE}	— —	— —	100 50	ksps ksps	V_{DD} = V_{REF} = 5V V_{DD} = V_{REF} = 2.7V
DC Accuracy:						
Resolution			12		bits	
Integral Nonlinearity	INL	— —	±0.75 ±1	±1 ±2	LSB LSB	MCP3202-B MCP3202-C
Differential Nonlinearity	DNL	—	±0.5	±1	LSB	No missing codes over temperature
Offset Error		—	±1.25	±3	LSB	
Gain Error		—	±1.25	±5	LSB	
Dynamic Performance:						
Total Harmonic Distortion	THD	—	-82	—	dB	V_{IN} = 0.1V to 4.9V@1 kHz
Signal to Noise and Distortion (SINAD)	SINAD	—	72	—	dB	V_{IN} = 0.1V to 4.9V@1 kHz
Spurious Free Dynamic Range	SFDR	—	86	—	dB	V_{IN} = 0.1V to 4.9V@1 kHz
Analog Inputs:						
Input Voltage Range for CH0 or CH1 in Single-Ended Mode		V_{SS}	—	V_{DD}	V	
Input Voltage Range for IN+ in Pseudo-Differential Mode	IN+	IN-	—	V_{DD}+IN-		See Sections 3.1 and 4.1
Input Voltage Range for IN- in Pseudo-Differential Mode	IN-	V_{SS}-100	—	V_{SS}+100	mV	See Sections 3.1 and 4.1
Leakage Current		—	.001	±1	µA	
Switch Resistance	R_{SS}	—	1 k	—	Ω	See Figure 4-1

Note 1: This parameter is established by characterization and not 100% tested.

2: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See Section 6.2 for more information.

ELECTRICAL CHARACTERISTICS (CONTINUED)

All parameters apply at V_{DD} = 5.5V, V_{SS} = 0V, T_{AMB} = -40°C to +85°C, f_{SAMPLE} = 100 ksps and f_{CLK} = 18*f_{SAMPLE} unless otherwise noted.

Parameter	Sym	Min.	Typ.	Max.	Units	Conditions
Sample Capacitor	C_{SAMPLE}	—	20	—	pF	See Figure 4-1
Digital Input/Output:						
Data Coding Format		Straight Binary				
High Level Input Voltage	V_{IH}	0.7 V_{DD}	—	—	V	
Low Level Input Voltage	V_{IL}	—	—	0.3 V_{DD}	V	
High Level Output Voltage	V_{OH}	4.1	—	—	V	I_{OH} = -1 mA, V_{DD} = 4.5V
Low Level Output Voltage	V_{OL}	—	—	0.4	V	I_{OL} = 1 mA, V_{DD} = 4.5V
Input Leakage Current	I_{LI}	-10	—	10	µA	V_{IN} = V_{SS} or V_{DD}
Output Leakage Current	I_{LO}	-10	—	10	µA	V_{OUT} = V_{SS} or V_{DD}
Pin Capacitance (All Inputs/Outputs)	C_{IN}, C_{OUT}	—	—	10	pF	V_{DD} = 5.0V (**Note 1**) T_{AMB} = 25°C, f = 1 MHz
Timing Parameters:						
Clock Frequency	f_{CLK}	—	—	1.8 0.9	MHz MHz	V_{DD} = 5V (**Note 2**) V_{DD} = 2.7V (**Note 2**)
Clock High Time	t_{HI}	250	—	—	ns	
Clock Low Time	t_{LO}	250	—	—	ns	
$\overline{CS}$ Fall To First Rising CLK Edge	t_{SUCS}	100	—	—	ns	
Data Input Setup Time	t_{SU}	—	—	50	ns	
Data Input Hold Time	t_{HD}	—	—	50	ns	
CLK Fall To Output Data Valid	t_{DO}	—	—	200	ns	See Test Circuits, Figure 1-2
CLK Fall To Output Enable	t_{EN}	—	—	200	ns	See Test Circuits, Figure 1-2
$\overline{CS}$ Rise To Output Disable	t_{DIS}	—	—	100	ns	See Test Circuits, Figure 1-2 **Note 1**
$\overline{CS}$ Disable Time	t_{CSH}	500	—	—	ns	
D_{OUT} Rise Time	t_R	—	—	100	ns	See Test Circuits, Figure 1-2 **Note 1**
D_{OUT} Fall Time	t_F	—	—	100	ns	See Test Circuits, Figure 1-2 **Note 1**
Power Requirements:						
Operating Voltage	V_{DD}	2.7	—	5.5	V	
Operating Current	I_{DD}	—	375	550	µA	V_{DD} = 5.0V, D_{OUT} unloaded
Standby Current	I_{DDS}	—	0.5	5	µA	$\overline{CS}$ = V_{DD} = 5.0V
Temperature Ranges:						
Specified Temperature Range	T_A	-40	—	+85	°C	
Operating Temperature Range	T_A	-40	—	+85	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistance:						
Thermal Resistance, 8L-PDIP	θ_{JA}	—	85	—	°C/W	
Thermal Resistance, 8L-SOIC	θ_{JA}	—	163	—	°C/W	
Thermal Resistance, 8L-MSOP	θ_{JA}	—	206	—	°C/W	
Thermal Resistance, 8L-TSSOP	θ_{JA}	—		—	°C/W	

Note 1: This parameter is established by characterization and not 100% tested.

2: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See Section 6.2 for more information.

MCP3421

18-Bit Analog-to-Digital Converter with I²C Interface and On-Board Reference

Features

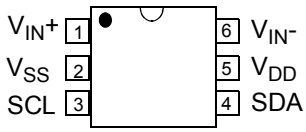
- 18-bit ΔΣ ADC in a SOT-23-6 package
- Differential input operation
- Self calibration of Internal Offset and Gain per each conversion
- On-board Voltage Reference:
  - Accuracy: 2.048V ± 0.05%
  - Drift: 5 ppm/°C
- On-board Programmable Gain Amplifier (PGA):
  - Gains of 1,2,4 or 8
- On-board Oscillator
- INL: 10 ppm of FSR (FSR = 4.096V/PGA)
- Programmable Data Rate Options:
  - 3.75 SPS (18 bits)
  - 15 SPS (16 bits)
  - 60 SPS (14 bits)
  - 240 SPS (12 bits)
- One-Shot or Continuous Conversion Options
- Low current consumption:
  - 145 µA typical (V_{DD}= 3V, Continuous Conversion)
  - 39 µA typical (V_{DD}= 3V, One-Shot Conversion with 1 SPS)
- Supports I²C Serial Interface:
  - Standard, Fast and High Speed Modes
- Single Supply Operation: 2.7V to 5.5V
- Extended Temperature Range: -40°C to 125°C

Typical Applications

- Portable Instrumentation
- Weigh Scales and Fuel Gauges
- Temperature Sensing with RTD, Thermistor, and Thermocouple
- Bridge Sensing for Pressure, Strain, and Force.

Package Types

SOT-23-6 **Top View**



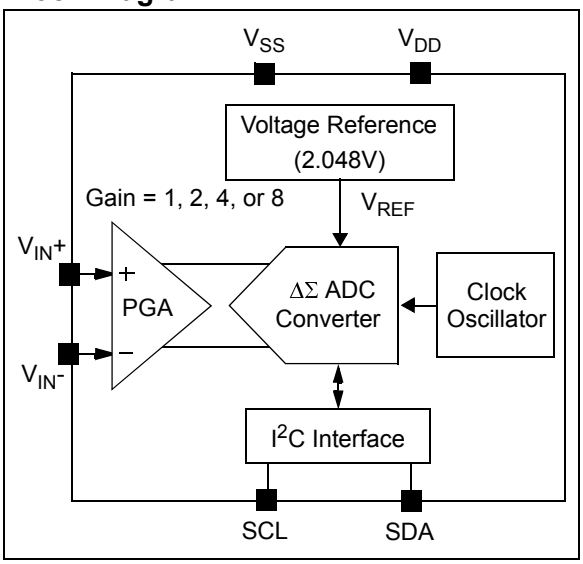
Description

The MCP3421 is a single channel low-noise, high accuracy ΔΣ A/D converter with differential inputs and up to 18 bits of resolution in a small SOT-23-6 package. The on-board precision 2.048V reference voltage enables an input range of ±2.048V differentially (Δ voltage = 4.096V). The device uses a two-wire I²C compatible serial interface and operates from a single 2.7V to 5.5V power supply.

The MCP3421 device performs conversion at rates of 3.75, 15, 60, or 240 samples per second (SPS) depending on the user controllable configuration bit settings using the two-wire I²C serial interface. This device has an on-board programmable gain amplifier (PGA). The user can select the PGA gain of x1, x2, x4, or x8 before the analog-to-digital conversion takes place. This allows the MCP3421 device to convert a smaller input signal with high resolution. The device has two conversion modes: (a) Continuous mode and (b) One-Shot mode. In One-Shot mode, the device enters a low current standby mode automatically after one conversion. This reduces current consumption greatly during idle periods.

The MCP3421 device can be used for various high accuracy analog-to-digital data conversion applications where design simplicity, low power, and small footprint are major considerations.

Block Diagram



1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings†

V_{DD}.....7.0V
All inputs and outputs w.r.t V_{SS} ..... –0.3V to V_{DD}+0.3V
Differential Input Voltage ..... $|V_{DD} - V_{SS}|$
Output Short Circuit Current .....Continuous
Current at Input Pins .....±2 mA
Current at Output and Supply Pins .....±10 mA
Storage Temperature.....-65°C to +150°C
Ambient Temp. with power applied .....-55°C to +125°C
ESD protection on all pins ..... ≥ 6 kV HBM, ≥ 400V MM
Maximum Junction Temperature (T_J).....+150°C

†Notice: Stresses above those listed under “Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

Electrical Specifications: Unless otherwise specified, all parameters apply for T_A = -40°C to +85°C, V_{DD} = +5.0V, V_{SS} = 0V, V_{IN}+ = V_{IN}- = V_{REF}/2. All ppm units use 2*V_{REF} as full-scale range.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Analog Inputs						
Differential Input Range		—	±2.048/PGA	—	V	$V_{IN} = V_{IN}+ - V_{IN}-$
Common-Mode Voltage Range (absolute) **(Note 1)**		V_{SS}-0.3	—	V_{DD}+0.3	V	
Differential Input Impedance **(Note 2)**	Z_{IND} (f)	—	2.25/PGA	—	MΩ	During normal mode operation
Common Mode input Impedance	Z_{INC} (f)	—	25	—	MΩ	PGA = 1, 2, 4, 8
System Performance						
Resolution and No Missing Codes **(Note 8)**		12	—	—	Bits	DR = 240 SPS
		14	—	—	Bits	DR = 60 SPS
		16	—	—	Bits	DR = 15 SPS
		18	—	—	Bits	DR = 3.75 SPS
Data Rate **(Note 3)**	DR	176	240	328	SPS	S1,S0 = ‘00’, (12 bits mode)
		44	60	82	SPS	S1,S0 = ‘01’, (14 bits mode)
		11	15	20.5	SPS	S1,S0 = ‘10’, (16 bits mode)
		2.75	3.75	5.1	SPS	S1,S0 = ‘11’, (18 bits mode)
Output Noise		—	1.5	—	μV_{RMS}	T_A = 25°C, DR = 3.75 SPS, PGA = 1, V_{IN} = 0

Note 1: Any input voltage below or greater than this voltage causes leakage current through the ESD diodes at the input pins. This parameter is ensured by characterization and not 100% tested.
2: This input impedance is due to 3.2 pF internal input sampling capacitor.
3: The total conversion speed includes auto-calibration of offset and gain.
4: INL is the difference between the endpoints line and the measured code at the center of the quantization band.
5: Includes all errors from on-board PGA and V_{REF}.
6: Full Scale Range (FSR) = 2 x 2.048/PGA = 4.096/PGA.
7: This parameter is ensured by characterization and not 100% tested.
8: This parameter is ensured by design and not 100% tested.

ELECTRICAL CHARACTERISTICS (CONTINUED)

Electrical Specifications: Unless otherwise specified, all parameters apply for T_A = -40°C to +85°C, V_{DD} = +5.0V, V_{SS} = 0V, $V_{IN}+ = V_{IN}- = V_{REF}/2$. All ppm units use $2*V_{REF}$ as full-scale range.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Integral Nonlinearity **(Note 4)**	INL	—	10	35	ppm of FSR	DR = 3.75 SPS **(Note 6)**
Internal Reference Voltage	V_{REF}	—	2.048	—	V	
Gain Error **(Note 5)**		—	0.05	0.35	%	PGA = 1, DR = 3.75 SPS
PGA Gain Error Match **(Note 5)**		—	0.1	—	%	Between any 2 PGA gains
Gain Error Drift **(Note 5)**		—	5	40	ppm/°C	PGA=1, DR=3.75 SPS
Offset Error	V_{OS}	—	15	40	µV	Tested at PGA = 1 V_{DD} = 5.0V and DR = 3.75 SPS
Offset Drift vs. Temperature		—	50	—	nV/°C	V_{DD} = 5.0V
Common-Mode Rejection		—	105	—	dB	at DC and PGA =1,
		—	110	—	dB	at DC and PGA =8, T_A = +25°C
Gain vs. V_{DD}		—	5	—	ppm/V	T_A = +25°C, V_{DD} = 2.7V to 5.5V, PGA = 1
Power Supply Rejection at DC		—	100	—	dB	T_A = +25°C, V_{DD} = 2.7V to 5.5V, PGA = 1
Power Requirements						
Voltage Range	V_{DD}	2.7	—	5.5	V	
Supply Current during Conversion	I_{DDA}	—	155	190	µA	V_{DD} = 5.0V
		—	145	—	µA	V_{DD} = 3.0V
Supply Current during Standby Mode	I_{DDS}	—	0.1	0.5	µA	
I^2C Digital Inputs and Digital Outputs						
High level input voltage	V_{IH}	0.7 V_{DD}	—	V_{DD}	V	
Low level input voltage	V_{IL}	—	—	0.3V_{DD}	V	
Low level output voltage	V_{OL}	—	—	0.4	V	I_{OL} = 3 mA, V_{DD} = +5.0V
Hysteresis of Schmitt Trigger for inputs **(Note 7)**	V_{HYST}	0.05V_{DD}	—	—	V	f_{SCL} = 100 kHz
Supply Current when I^2C bus line is active	I_{DDB}	—	—	10	µA	
Input Leakage Current	I_{ILH}	—	—	1	µA	V_{IH} = 5.5V
	I_{ILL}	-1	—	—	µA	V_{IL} = GND
Pin Capacitance and I^2C Bus Capacitance						
Pin capacitance	C_{PIN}	—	—	10	pF	
I^2C Bus Capacitance	C_b	—	—	400	pF	
Thermal Characteristics						
Specified Temperature Range	T_A	-40	—	+85	°C	
Operating Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	

Note 1: Any input voltage below or greater than this voltage causes leakage current through the ESD diodes at the input pins. This parameter is ensured by characterization and not 100% tested.

2: This input impedance is due to 3.2 pF internal input sampling capacitor.

3: The total conversion speed includes auto-calibration of offset and gain.

4: INL is the difference between the endpoints line and the measured code at the center of the quantization band.

5: Includes all errors from on-board PGA and V_{REF}.

6: Full Scale Range (FSR) = 2 x 2.048/PGA = 4.096/PGA.

7: This parameter is ensured by characterization and not 100% tested.

8: This parameter is ensured by design and not 100% tested.

MCP3551/3

Low-Power, Single-Channel 22-Bit Delta-Sigma ADCs

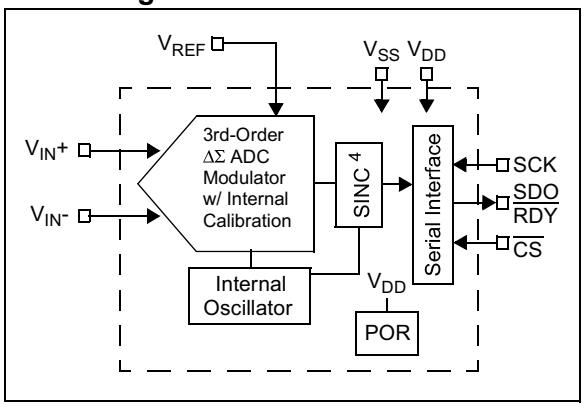
Features

- 22-bit ADC in Small 8-pin MSOP Package with Automatic Internal Offset and Gain Calibration
- Low-Output Noise of 2.5 μV_{RMS} with Effective Resolution of 21.9 bits (MCP3551)
- 3 µV Typical Offset Error
- 2 ppm Typical Full-Scale Error
- 6 ppm Maximum INL Error
- Total Unadjusted Error Less Than 10 ppm
- No Digital Filter Settling Time, Single-Command Conversions through 3-wire SPI™ Interface
- Ultra-Low Conversion Current (MCP3551):
  - 100 µA typical (V_{DD} = 2.7V)
  - 120 µA typical (V_{DD} = 5.0V)
- Differential Input with V_{SS} to V_{DD} Common Mode Range
- 2.7V to 5.5V Single-Supply Operation
- Extended Temperature Range:
  - -40°C to +125°C

Applications

- Weigh Scales
- Direct Temperature Measurement
- 6-digit DVMs
- Instrumentation
- Data Acquisition
- Strain Gauge Measurement

Block Diagram



Description

The Microchip Technology Inc. MCP3551/3 devices are 2.7V to 5.5V low-power, 22-bit Delta-Sigma Analog-to-Digital Converters (ADCs). The devices offer output noise as low as 2.5 μV_{RMS}, with a total unadjusted error of 10 ppm. The family exhibits 6 ppm Integral Non-Linearity (INL) error, 3 µV offset error and less than 2 ppm full-scale error. The MCP3551/3 devices provide high accuracy and low noise performance for applications where sensor measurements (such as pressure, temperature and humidity) are performed. With the internal oscillator and high oversampling rate, minimal external components are required for high-accuracy applications.

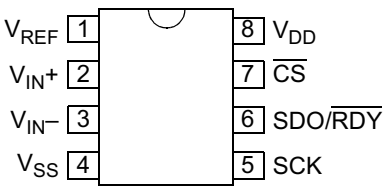
This product line has fully differential analog inputs, making it compatible with a wide variety of sensor, industrial control or process control applications.

The MCP3551/3 devices operate from -40°C to +125°C and are available in the space-saving 8-pin MSOP and SOIC packages.

Package Types:

MCP3551/3

SOIC, MSOP



Device Selection Table

Part Number	Sample Rate	Effective Resolution	50/60 Hz Rejection
MCP3551	13.75 sps	21.9 bits	50/60 Hz (simultaneous)
MCP3553	60 sps	20.6 bits	N/A

1.0 ELECTRICAL CHARACTERISTICS

1.1 Maximum Ratings*

V_{DD}.....7.0V
All inputs and outputs w.r.t V_{SS} ..... –0.3V to V_{DD} + 0.3V
Difference Input Voltage ..... $|V_{DD} - V_{SS}|$
Output Short Circuit Current ..... Continuous
Current at Input Pins .....±2 mA
Current at Output and Supply Pins .....±10 mA
Storage Temperature.....-65°C to +150°C
Ambient temp. with power applied .....-55°C to +125°C
ESD protection on all pins (HBM, MM) ..... ≥ 6 kV, ≥ 400V
Maximum Junction Temperature (T_J) .....+150°C

† Notice: Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

DC CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, all parameters apply at -40°C ≤ T_A ≤ +85°C, V_{DD} = 2.7V or 5.0V. V_{REF} = 2.5V. V_{IN}+ = V_{IN}- = V_{CM} = V_{REF}/2. All ppm units use 2*V_{REF} as full-scale range. Unless otherwise noted, specification applies to entire MCP3551/3 family.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Noise Performance (MCP3551)						
No Missing Codes	NMC	22	—	—	bits	At DC (**Note 5**)
Output Noise	e_N	—	2.5	—	μV_{RMS}	
Effective Resolution	ER	—	21.9	—	bits RMS	V_{REF} = 5V
Noise Performance (MCP3553)						
No Missing Codes	NMC	20	—	—	bits	At DC (**Note 5**)
Output Noise	e_N	—	6	—	μV_{RMS}	
Effective Resolution	ER	—	20.6	—	bits RMS	V_{REF} = 5V
Conversion Times						
MCP3551	t_{CONV}	-0.5%	72.73	+0.5%	ms	
MCP3553	t_{CONV}	-0.5%	16.67	+0.5%	ms	
Accuracy						
Integral Non-Linearity	INL	—	±2	6	ppm	T_A = +25°C only (**Note 2**)
Offset Error	V_{OS}	-12	±3	+12	µV	T_A = +25°C
		—	±4	—	µV	T_A = +85°C
		—	±6	—	µV	T_A = +125°C
Positive Full-Scale Error	$V_{FS,P}$	-10	±2	+10	ppm	T_A = +25°C only
Negative Full-Scale Error	$V_{FS,N}$	-10	±2	+10	ppm	T_A = +25°C only
Offset Drift		—	0.040	—	ppm/°C	
Positive/Negative Full-Scale Error Drift		—	0.028	—	ppm/°C	

Note 1: This parameter is established by characterization and not 100% tested.
2: INL is the difference between the endpoints line and the measured code at the center of the quantization band.
3: This current is due to the leakage current and the current due to the offset voltage between V_{IN}+ and V_{IN}-.
4: Input impedance is inversely proportional to clock frequency; typical values are for the MCP3551 device. V_{REF} = 5V.
5: Characterized by design, but not tested.
6: Rejection performance depends on internal oscillator accuracy; see **Section 4.0 "Device Overview"** for more information on oscillator and digital filter design. MCP3551 device rejection specifications characterized from 49 to 61 Hz.

DC CHARACTERISTICS (CONTINUED)

Electrical Specifications: Unless otherwise indicated, all parameters apply at -40°C ≤ T_A ≤ +85°C, V_{DD} = 2.7V or 5.0V. V_{REF} = 2.5V. V_{IN}+ = V_{IN}- = V_{CM} = V_{REF}/2. All ppm units use 2*V_{REF} as full-scale range. Unless otherwise noted, specification applies to entire MCP3551/3 family.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Rejection Performance(1,6)						
Common Mode DC Rejection		—	-135	—	dB	V_{CM} range from 0 to V_{DD}
Power Supply DC Rejection		—	-115	—	dB	
Common Mode 50/60 Hz Rejection	CMRR	—	-135	—	dB	V_{CM} varies from 0V to V_{DD}
Power Supply 50/60 Hz Rejection	PSRR	—	-85	—	dB	V_{DD} varies from 4.5V to 5.5V
Normal Mode 50 or 60 Hz Rejection	NMRR	—	-85	—	dB	**MCP3551** only, 0 < V_{CM} < V_{DD}, -V_{REF} < V_{IN} = (V_{IN} + -V_{IN}-) < +V_{REF}
Analog Inputs						
Differential Input Range	V_{IN+} – V_{IN-}	-V_{REF}	—	+V_{REF}	V	
Absolute/Common Mode Voltages		V_{SS} - 0.3	—	V_{DD} + 0.3	V	
Analog Input Sampling Capacitor		—	10	—	pF	**Note 5**
Differential Input Impedance		—	2.4	—	MΩ	
Shutdown Mode Leakage Current		—	1	—	nA	V_{IN}+ = V_{IN}- = V_{DD}; $\overline{CS}$ = V_{DD} **(Note 3)**
Reference Input						
Voltage Range		0.1	—	V_{DD}	V	
Reference Input Sampling Capacitor		—	15	—	pF	**Note 5**
Reference Input Impedance		—	2.4	—	MΩ	**Note 4**
Shutdown Mode Reference Leakage Current		—	1	—	nA	V_{IN}+ = V_{IN}- = V_{SS}; $\overline{CS}$ = V_{DD}
Power Requirements						
Power Supply Voltage Range	V_{DD}	2.7	—	5.5	V	
MCP3551 Supply Current	I_{DD}	—	120	170	µA	V_{DD} = 5V
		—	100	—	µA	V_{DD} = 2.7V
MCP3553 Supply Current	I_{DD}	—	140	185	µA	V_{DD} = 5V
		—	120	—	µA	V_{DD} = 2.7V
Supply Current, Sleep Mode	I_{DDSL}	—	10		µA	
Supply Current, Shutdown Mode	I_{DDS}	—	—	1	µA	$\overline{CS}$ = SCK = V_{DD}
Serial Interface						
Voltage Input High ($\overline{CS}$, SCK)	V_{IH}	0.7 V_{DD}	—	—	V	
Voltage Input Low ($\overline{CS}$, SCK)	V_{IL}	—	—	0.4	V	
Voltage Output High (SDO/$\overline{RDY}$)	V_{OH}	V_{DD} - 0.5	—	—	V	V_{OH} = 1 mA, V_{DD} = 5.0V
Voltage Output Low (SDO/$\overline{RDY}$)	V_{OL}	—	—	0.4	V	V_{OH} = -1 mA, V_{DD} = 5.0V
Input leakage Current ($\overline{CS}$, SCK)	I_{LI}	-1	—	1	µA	
Internal Pin Capacitance ($\overline{CS}$, SCK, SDO/$\overline{RDY}$)	C_{INT}	—	5	—	pF	**Note 1**

Note 1: This parameter is established by characterization and not 100% tested.

2: INL is the difference between the endpoints line and the measured code at the center of the quantization band.

3: This current is due to the leakage current and the current due to the offset voltage between V_{IN}+ and V_{IN}-.

4: Input impedance is inversely proportional to clock frequency; typical values are for the MCP3551 device. V_{REF} = 5V.

5: Characterized by design, but not tested.

6: Rejection performance depends on internal oscillator accuracy; see **Section 4.0 "Device Overview"** for more information on oscillator and digital filter design. MCP3551 device rejection specifications characterized from 49 to 61 Hz.

MCP4821/MCP4822

12-Bit DACs with Internal V_{REF} and SPI™ Interface

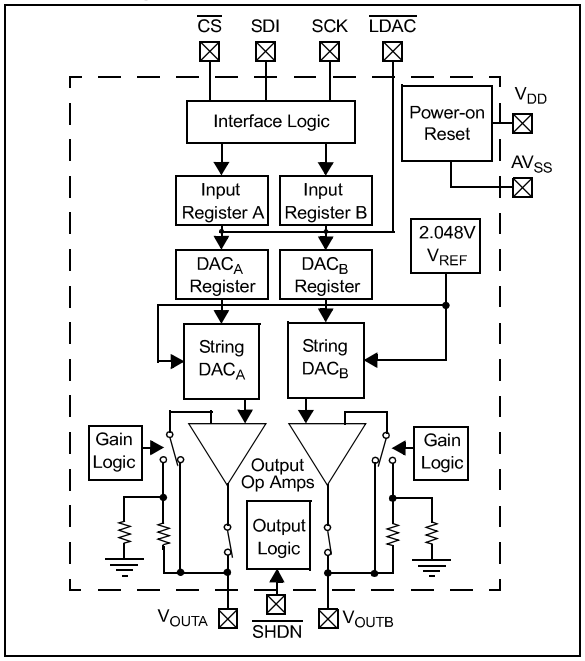
Features

- 12-Bit Resolution
- ±0.2 LSb DNL (typ.)
- ±2 LSb INL (typ.)
- Single or Dual Channel
- Rail-to-Rail Output
- SPI™ Interface with 20 MHz Clock Support
- Simultaneous Latching of the Dual DACs with $\overline{LDAC}$ pin
- Fast Settling Time of 4.5 µs
- Selectable Unity or 2x Gain Output
- 2.048V Internal Band Gap Voltage Reference
- 50 ppm/°C V_{REF} Temperature Coefficient
- 2.7V to 5.5V Single-Supply Operation
- Extended Temperature Range: -40°C to +125°C

Applications

- Set Point or Offset Trimming
- Sensor Calibration
- Precision Selectable Voltage Reference
- Portable Instrumentation (Battery-Powered)
- Calibration of Optical Communication Devices

Block Diagram



Description

The Microchip Technology Inc. MCP482X devices are 2.7V–5.5V, low-power, low DNL, 12-bit Digital-to-Analog Converters (DACs) with internal band gap voltage reference, optional 2x-buffered output and Serial Peripheral Interface (SPI™).

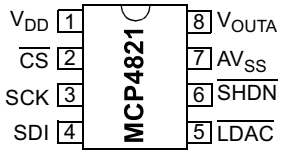
The MCP482X family of DACs provide high accuracy and low noise performance for industrial applications where calibration or compensation of signals (such as temperature, pressure and humidity) are required.

The MCP482X devices are available in the extended temperature range and PDIP, SOIC and MSOP packages.

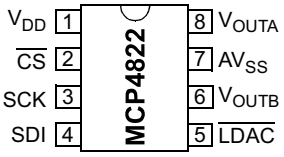
The MCP482X devices utilize a resistive string architecture, with its inherent advantages of low DNL error, low ratio metric temperature coefficient and fast settling time. These devices are specified over the extended temperature range. The MCP482X family includes double-buffered registers, allowing simultaneous updates using the $\overline{LDAC}$ pin. These devices also incorporate a Power-On Reset (POR) circuit to ensure reliable power-up.

Package Types

8-Pin PDIP, SOIC, MSOP



8-Pin PDIP, SOIC, MSOP



1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

V_{DD} 6.5V
All inputs and outputs .....AV_{SS} – 0.3V to V_{DD} + 0.3V
Current at Input Pins .....±2 mA
Current at Supply Pins .....±50 mA
Current at Output Pins .....±25 mA
Storage temperature .....-65°C to +150°C
Ambient temp. with power applied .....-55°C to +125°C
ESD protection on all pins .....≥ 4 kV (HBM), ≥ 400V (MM)
Maximum Junction Temperature (T_J) .....+150°C

† Notice: Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

5V AC/DC CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 5V, AV_{SS} = 0V, V_{REF} = 2.048V, output buffer gain (G) = 2x, R_L = 5 kΩ to GND, C_L = 100 pF, T_A = -40 to +85°C. Typical values at +25°C.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Power Requirements						
Input Voltage	V_{DD}	2.7	—	5.5		
Input Current - **MCP4821** - **MCP4822**	I_{DD}	— —	330 415	400 750	µA	Digital inputs grounded, Output unloaded, code = 0x000
Hardware Shutdown Current	I_{SHDN}	—	0.3	2	µA	
Software Shutdown Current	I_{SHDN_SW}	—	3.3	6	µA	
Power-on-Reset Threshold	V_{POR}	—	2.0	—	V	
DC Accuracy						
Resolution	n	12	—	—	Bits	
INL Error	INL	-12	2	12	LSb	
DNL **(Note 1)**	DNL	-0.75	±0.2	+0.75	LSb	Device is monotonic
Offset Error	V_{OS}	-1	±0.02	1	% of FSR	Code = 0x000h
Offset Error Temperature Coefficient	V_{OS}/°C	—	0.16	—	ppm/°C	-45°C to 25°C
		—	-0.44	—	ppm/°C	+25°C to 85°C
Gain Error	g_E	-2	-0.10	2	% of FSR	Code 0xFFFh, not including offset error
Gain Error Temperature Coefficient	ΔG/°C	—	-3	—	ppm/°C	
Internal Voltage Reference (V_{REF})						
Nominal Reference Voltage	V_{REF}	2.008	2.048	2.088	V	V_{OUTA} when G = 1x and Code = 0xFFFh
Temperature Coefficient **(Note 1)**	ΔV_{REF}/°C	—	125	325	ppm/°C	-40°C to 0°C
		—	0.25	0.65	LSb/°C	-40°C to 0°C
		—	45	160	ppm/°C	0°C to +85°C
		—	0.09	0.32	LSb/°C	0°C to +85°C
Output Noise (V_{REF} Noise)	E_{NREF} (0.1-10 Hz)	—	290	—	μV_{p-p}	Code = 0xFFFh, G = 1
Output Noise Density	e_{NREF} (1 kHz)	—	1.2	—	µV/√Hz	Code = 0xFFFh, G = 1
	e_{NREF} (10 kHz)	—	1.0	—	µV/√Hz	Code = 0xFFFh, G = 1
1/f Corner Frequency	f_{CORNER}	—	400	—	Hz	

Note 1: By design, not production tested.
2: Too small to quantify.

5V AC/DC CHARACTERISTICS (CONTINUED)

Electrical Specifications: Unless otherwise indicated, V_{DD} = 5V, AV_{SS} = 0V, V_{REF} = 2.048V, output buffer gain (G) = 2x, R_L = 5 kΩ to GND, C_L = 100 pF, T_A = -40 to +85°C. Typical values at +25°C.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Output Amplifier						
Output Swing	V_{OUT}	—	0.010 to V_{DD} – 0.040	—		Accuracy is better than 1 LSb for V_{OUT} = 10 mV to (V_{DD} – 40 mV)
Phase Margin	PM	—	66	—	°	
Slew Rate	SR	—	0.55	—	V/µs	
Short Circuit Current	I_{SC}	—	15	24	mA	
Settling Time	$t_{SETTLING}$	—	4.5	—	µs	Within 1/2 LSb of final value from 1/4 to 3/4 full-scale range
Dynamic Performance						
DAC-to-DAC Crosstalk		—	<10	—	nV-s	**Note 2**
Major Code Transition Glitch		—	45	—	nV-s	1 LSb change around major carry (0111...1111 to 1000...0000)
Digital Feedthrough		—	<10	—	nV-s	**Note 2**
Analog Crosstalk		—	<10	—	nV-s	**Note 2**

Note 1: By design, not production tested.
2: Too small to quantify.

3V AC/DC CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 3V, AV_{SS} = 0V, V_{REF} = 2.048V external, output buffer gain (G) = 1x, R_L = 5 kΩ to GND, C_L = 100 pF, T_A = -40 to +85°C. Typical values at 25°C

Parameters	Sym	Min	Typ	Max	Units	Conditions
Power Requirements						
Input Voltage	V_{DD}	2.7	—	5.5		
Input Current - **MCP4821** - **MCP4822**	I_{DD}	— —	300 415	400 750	µA	Digital inputs grounded, Output unloaded, code = 0x000
Hardware Shutdown Current	I_{SHDN}	—	0.25	2	µA	
Software Shutdown Current	I_{SHDN_SW}	—	2	6	µA	
Power-On Reset threshold	V_{POR}	—	2.0	—	V	
DC Accuracy						
Resolution	n	12	—	—	Bits	
INL Error	INL	-12	±3	12	LSb	
DNL **(Note 1)**	DNL	-0.75	±0.3	0.75	LSb	Device is monotonic
Offset Error	V_{OS}	-1	±0.02	1	% of FSR	Code 0x000h
Offset Error Temperature Coefficient	V_{OS}/°C	—	0.5	—	ppm/°C	-45°C to +25°C
		—	-0.77	—	ppm/°C	+25°C to +85°C
Gain Error	g_E	-2	-0.15	2	% of FSR	Code 0xFFFh, not including offset error
Gain Error Temperature Coefficient	ΔG/°C	—	-3	—	ppm/°C	

Note 1: By design, not production tested.
2: Too small to quantify.

MCP9800/1/2/3

2-Wire High-Accuracy Temperature Sensor

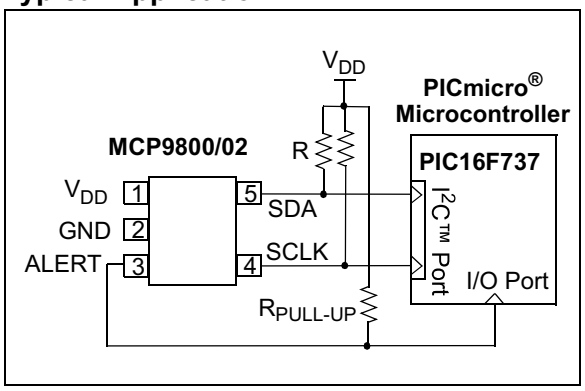
Features

- Temperature-to-Digital Converter
- Accuracy with 12-bit Resolution:
  - ±0.5°C (typ.) at +25°C
  - ±1°C (max.) from -10°C to +85°C
  - ±2°C (max.) from -10°C to +125°C
  - ±3°C (max.) from -55°C to +125°C
- User-selectable Resolution: 9 – 12 bit
- Operating Voltage Range: 2.7V to 5.5V
- 2-wire Interface: I²C™/SMBus Compatible
- Operating Current: 200 µA (typ.)
- Shutdown Current: 1 µA (max.)
- Power-saving One-shot Temperature Measurement
- Available Packages: SOT-23-5, MSOP-8, SOIC-8

Typical Applications

- Personal Computers and Servers
- Hard Disk Drives and Other PC Peripherals
- Entertainment Systems
- Office Equipment
- Data Communication Equipment
- Mobile Phones
- General-purpose Temperature Monitoring

Typical Application



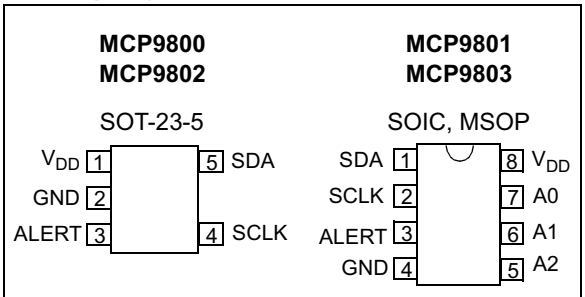
Description

Microchip Technology Inc.'s MCP9800/1/2/3 family of digital temperature sensors converts temperatures between -55°C and +125°C to a digital word. They provide an accuracy of ±1°C (max.) from -10°C to +85°C.

The MCP9800/1/2/3 family comes with user-programmable registers that provide flexibility for temperature sensing applications. The register settings allow user-selectable 9-bit to 12-bit temperature measurement resolution, configuration of the power-saving Shutdown and One-shot (single conversion on command while in the Shutdown) modes and the specification of both temperature alert output and hysteresis limits. When the temperature changes beyond the specified limits, the MCP9800/1/2/3 outputs an alert signal. The user has the option of setting the alert output signal polarity as an active-low or active-high comparator output for thermostat operation, or as temperature event interrupt output for microprocessor-based systems.

This sensor has an industry standard 2-wire, I²C™/SMBus compatible serial interface, allowing up to eight devices to be controlled in a single serial bus. These features make the MCP9800/1/2/3 ideal for sophisticated multi-zone temperature-monitoring applications.

Package Types



MCP9800/02A0: A2, A1, A0 are internally set to (0, 0, 0)

MCP9800/02A5: A2, A1, A0 are internally set to (1, 0, 1)

MCP9802/03: Serial Bus time-out 35 ms (typ.)

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

V_{DD}..... 6.0V

Voltage at all Input/Output pins .... GND – 0.3V to 5.5V

Storage temperature .....-65°C to +150°C

Ambient temp. with power applied .....-55°C to +125°C

Junction Temperature (T_J)..... 150°C

ESD protection on all pins (HBM:MM)..... (4 kV:400V)

Latch-Up Current at each pin ..... ±200 mA

†Notice: Stresses above those listed under "Maximum ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIN FUNCTION TABLE

NAME	FUNCTION
SDA	Bidirectional Serial Data (open-drain output)
SCLK	Serial Clock Input
ALERT	Temperature Alert Output (open-drain)
A2	Address Select Pin (bit 2)
A1	Address Select Pin (bit 1)
A0	Address Select Pin (bit 0)
V_{DD}	Power Supply Input
GND	Ground

DC CHARACTERISTICS

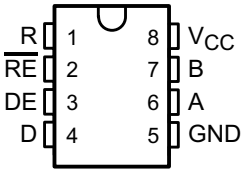
Electrical Specifications: Unless otherwise indicated, V_{DD} = 2.7V to 5.5V, GND = Ground, and T_A = -55°C to +125°C.

Parameters	Sym	Min	Typ	Max	Unit	Conditions
Power Supply						
Operating Voltage Range	V_{DD}	2.7	—	5.5	V	
Operating Current	I_{DD}	—	200	400	µA	Continuous Operation
Shutdown Current	I_{SHDN}	—	0.1	1	µA	Shutdown Mode
Power On Reset Threshold (POR)	V_{POR}	—	1.7	—	V	V_{DD} falling edge
Temperature Sensor Accuracy						
Accuracy with 12-bit Resolution:						
T_A = +25°C	T_{ACY}	—	±0.5	—	°C	V_{DD} = 3.3V
-10°C < T_A ≤ +85°C	T_{ACY}	-1.0	—	+1.0	°C	V_{DD} = 3.3V
-10°C < T_A ≤ +125°C	T_{ACY}	-2.0	—	+2.0	°C	V_{DD} = 3.3V
-55°C < T_A ≤ +125°C	T_{ACY}	-3.0	—	+3.0	°C	V_{DD} = 3.3V
Internal ΣΔ ADC						
Conversion Time:						
9-bit Resolution	t_{CONV}	—	30	75	ms	33 samples/sec (typ.)
10-bit Resolution	t_{CONV}	—	60	150	ms	17 samples/sec (typ.)
11-bit Resolution	t_{CONV}	—	120	300	ms	8 samples/sec (typ.)
12-bit Resolution	t_{CONV}	—	240	600	ms	4 samples/sec (typ.)
Alert Output (Open-drain)						
High-level Current	I_{OH}	—	—	1	µA	V_{OH} = 5V
Low-level Voltage	V_{OL}	—	—	0.4	V	I_{OL}= 3 mA
Thermal Response						
Response Time	t_{RES}	—	1.4	—	s	Time to 63% (88°C) 27°C (Air) to 125°C (oil bath)

SLLS100A – JUNE 1984 – REVISED MAY 1995

- **Bidirectional Transceiver**
- **Meets or Exceeds the Requirements of ANSI Standards EIA/TIA-422-B and ITU Recommendation V.11**
- **Designed for Multipoint Transmission on Long Bus Lines in Noisy Environments**
- **3-State Driver and Receiver Outputs**
- **Individual Driver and Receiver Enables**
- **Wide Positive and Negative Input/Output Bus Voltage Ranges**
- **Driver Output Capability . . . ±60 mA Max**
- **Thermal-Shutdown Protection**
- **Driver Positive- and Negative-Current Limiting**
- **Receiver Input Impedance . . . 12 kΩ Min**
- **Receiver Input Sensitivity . . . ±200 mV**
- **Receiver Input Hysteresis . . . 50 mV Typ**
- **Operates From Single 5-V Supply**
- **Low Power Requirements**

D OR P PACKAGE
(TOP VIEW)



description

The SN75176A differential bus transceiver is a monolithic integrated circuit designed for bidirectional data communication on multipoint bus-transmission lines. It is designed for balanced transmission lines and meets ANSI Standard EIA/TIA-422-B and ITU Recommendation V.11.

The SN75176A combines a 3-state differential line driver and a differential input line receiver, both of which operate from a single 5-V power supply. The driver and receiver have active-high and active-low enables, respectively, that can be externally connected together to function as a direction control. The driver differential outputs and the receiver differential inputs are connected internally to form differential input/output (I/O) bus ports that are designed to offer minimum loading to the bus whenever the driver is disabled or V_{CC} = 0. These ports feature wide positive and negative common-mode voltage ranges making the device suitable for party-line applications.

The driver is designed to handle loads up to 60 mA of sink or source current. The driver features positive- and negative-current limiting and thermal shutdown for protection from line fault conditions. Thermal shutdown is designed to occur at a junction temperature of approximately 150°C. The receiver features a minimum input impedance of 12 kΩ, an input sensitivity of ±200 mV, and a typical input hysteresis of 50 mV.

The SN75176A can be used in transmission-line applications employing the SN75172 and SN75174 quadruple differential line drivers and SN75173 and SN75175 quadruple differential line receivers.

The SN75176A is characterized for operation from 0°C to 70°C.

Function Tables

DRIVER

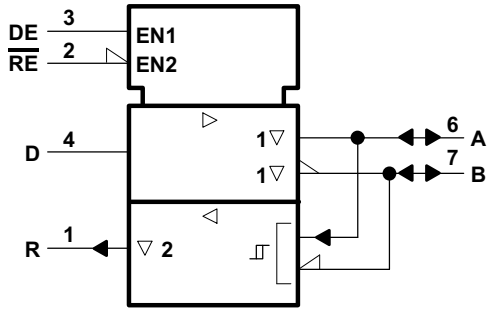
INPUT D	ENABLE DE	OUTPUTS A	OUTPUTS B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A – B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V < V_{ID} < 0.2 V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	?

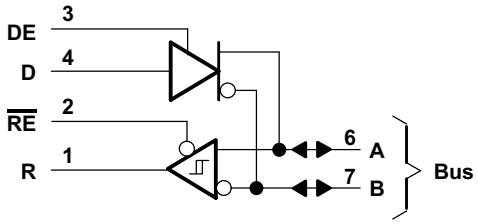
H = high level, L = low level, ? = indeterminate,
X = irrelevant, Z = high impedance (off)

logic symbol†

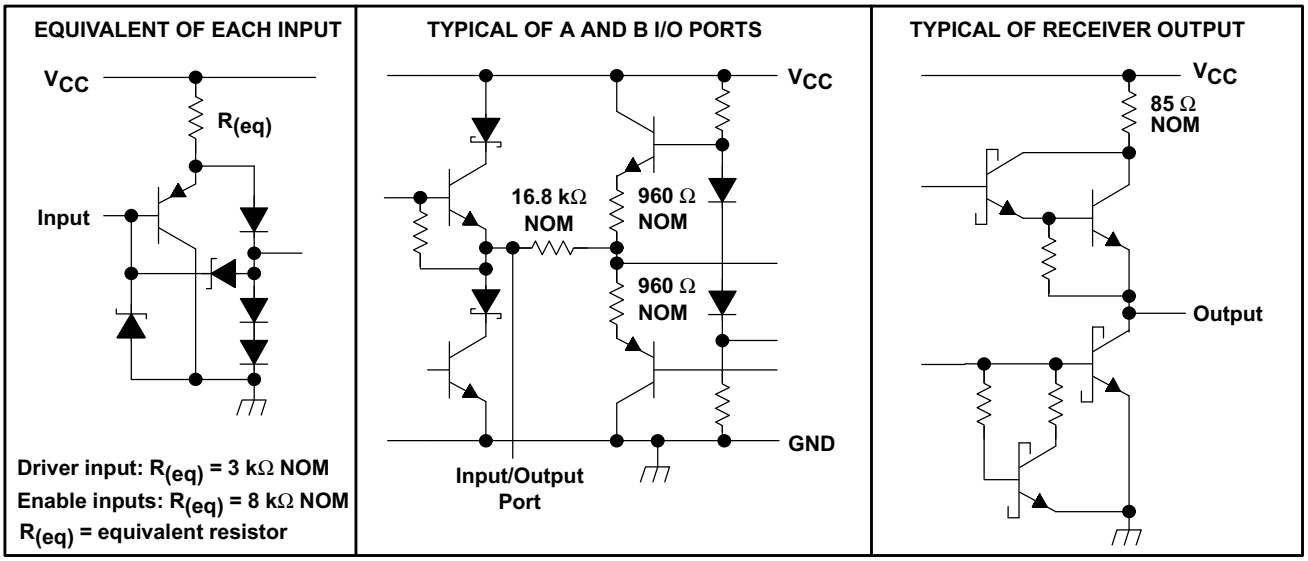


† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)



schematics of inputs and outputs



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Voltage range at any bus terminal . . . . . –10 V to 15 V
Enable input voltage, V_I . . . . . 5.5 V
Continuous total power dissipation . . . . . See Dissipation Rating Table
Operating free-air temperature range, T_A . . . . . 0°C to 70°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds . . . . . 260°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

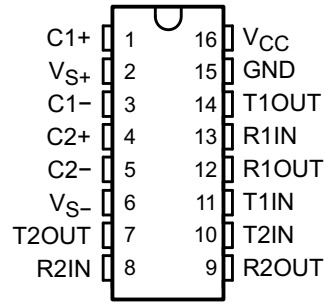
NOTE 1: All voltage values, except differential input/output bus voltage, are with respect to network ground terminal.

DISSIPATION RATING TABLE

PACKAGE	$T_A \le 25°C$ POWER RATING	DERATING FACTOR ABOVE $T_A = 25°C$	$T_A = 70°C$ POWER RATING	$T_A = 105°C$ POWER RATING
D	725 mW	5.8 mW/°C	464 mW	261 mW
P	1100 mW	8.8 mW/°C	704 mW	396 mW

- **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

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
−40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

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

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

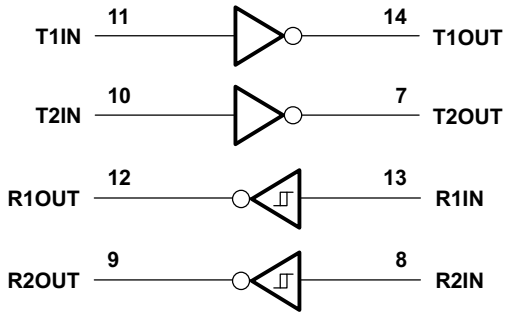
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) . . . . . −0.3 V to 6 V
Positive output supply voltage range, V_{S+} . . . . . V_{CC} − 0.3 V to 15 V
Negative output supply voltage range, V_{S-} . . . . . −0.3 V to −15 V
Input voltage range, V_I: Driver . . . . . −0.3 V to V_{CC} + 0.3 V
Receiver . . . . . ±30 V
Output voltage range, V_O: T1OUT, T2OUT . . . . . V_{S-} − 0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT . . . . . −0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT . . . . . Unlimited
Package thermal impedance, θ_{JA} (see Notes 2 and 3): D package . . . . . 73°C/W
DW package . . . . . 57°C/W
N package . . . . . 67°C/W
NS package . . . . . 64°C/W
Operating virtual junction temperature, T_J . . . . . 150°C
Storage temperature range, T_{stg} . . . . . −65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltages are with respect to network GND.
2. Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
3. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	−40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 4 and Figure 4)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, All outputs open, T_A = 25°C		8	10	mA

‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.

NOTE 4: Test conditions are C1−C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.

Single-Supply, Rail-to-Rail Low Power FET-Input Op Amp

AD822

FEATURES

- True single-supply operation
  - Output swings rail-to-rail
  - Input voltage range extends below ground
  - Single-supply capability from 5 V to 30 V
  - Dual-supply capability from ±2.5 V to ±15 V
- High load drive
  - Capacitive load drive of 350 pF, G = +1
  - Minimum output current of 15 mA
- Excellent ac performance for low power
  - 800 µA maximum quiescent current per amplifier
  - Unity-gain bandwidth: 1.8 MHz
  - Slew rate of 3 V/µs
- Good dc performance
  - 800 µV maximum input offset voltage
  - 2 µV/°C typical offset voltage drift
  - 25 pA maximum input bias current
- Low noise
  - 13 nV/√Hz @ 10 kHz
  - No phase inversion

APPLICATIONS

- Battery-powered precision instrumentation
- Photodiode preamps
- Active filters
- 12-bit to 14-bit data acquisition systems
- Medical instrumentation
- Low power references and regulators

CONNECTION DIAGRAM

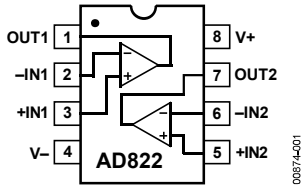


Figure 1. 8-Lead PDIP (N Suffix); 8-Lead MSOP (RM Suffix); and 8-Lead SOIC_N (R Suffix)

GENERAL DESCRIPTION

The AD822 is a dual precision, low power FET input op amp that can operate from a single supply of 5 V to 30 V or dual supplies of ±2.5 V to ±15 V. It has true single-supply capability with an input voltage range extending below the negative rail, allowing the AD822 to accommodate input signals below ground in the single-supply mode. Output voltage swing extends to within 10 mV of each rail, providing the maximum output dynamic range.

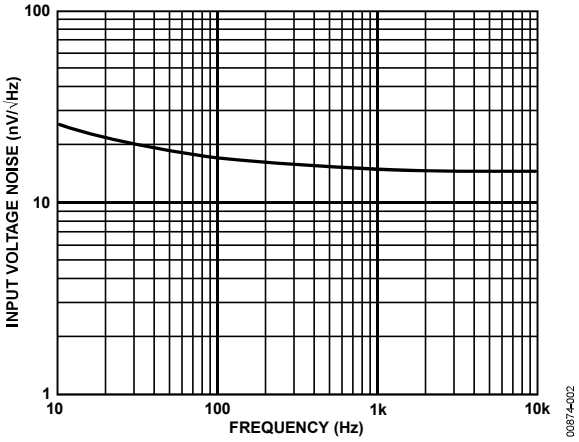


Figure 2. Input Voltage Noise vs. Frequency

Offset voltage of 800 µV maximum, offset voltage drift of 2 µV/°C, input bias currents below 25 pA, and low input voltage noise provide dc precision with source impedances up to a gigaohm. The 1.8 MHz unity-gain bandwidth, –93 dB THD at 10 kHz, and 3 V/µs slew rate are provided with a low supply current of 800 µA per amplifier.

Rev. I

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

SPECIFICATIONS

V_S = 0 V, 5 V @ T_A = 25°C, V_{CM} = 0 V, V_{OUT} = 0.2 V, unless otherwise noted.

Table 1.

		A Grade			B Grade			
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
DC PERFORMANCE								
Initial Offset			0.1	0.8		0.1	0.4	mV
Maximum Offset Over Temperature			0.5	1.2		0.5	0.9	mV
Offset Drift			2			2		µV/°C
Input Bias Current	V_{CM} = 0 V to 4 V		2	25		2	10	pA
At T_{MAX}			0.5	5		0.5	2.5	nA
Input Offset Current			2	20		2	10	pA
At T_{MAX}			0.5			0.5		nA
Open-Loop Gain	V_{OUT} = 0.2 V to 4 V							
	R_L = 100 kΩ	500	1000		500	1000		V/mV
T_{MIN} to T_{MAX}		400			400			V/mV
	R_L = 10 kΩ	80	150		80	150		V/mV
T_{MIN} to T_{MAX}		80			80			V/mV
	R_L = 1 kΩ	15	30		15	30		V/mV
T_{MIN} to T_{MAX}		10			10			V/mV
NOISE/HARMONIC PERFORMANCE								
Input Voltage Noise								
f = 0.1 Hz to 10 Hz			2			2		µV p-p
f = 10 Hz			25			25		nV/√Hz
f = 100 Hz			21			21		nV/√Hz
f = 1 kHz			16			16		nV/√Hz
f = 10 kHz			13			13		nV/√Hz
Input Current Noise								
f = 0.1 Hz to 10 Hz			18			18		fA p-p
f = 1 kHz			0.8			0.8		fA/√Hz
Harmonic Distortion	R_L = 10 kΩ to 2.5 V							
f = 10 kHz	V_{OUT} = 0.25 V to 4.75 V		−93			−93		dB
DYNAMIC PERFORMANCE								
Unity-Gain Frequency			1.8			1.8		MHz
Full Power Response	V_{OUT} p-p = 4.5 V		210			210		kHz
Slew Rate			3			3		V/µs
Settling Time								
To 0.1%	V_{OUT} = 0.2 V to 4.5 V		1.4			1.4		µs
To 0.01%	V_{OUT} = 0.2 V to 4.5 V		1.8			1.8		µs
MATCHING CHARACTERISTICS								
Initial Offset				1.0			0.5	mV
Maximum Offset Over Temperature				1.6			1.3	mV
Offset Drift			3			3		µV/°C
Input Bias Current				20			10	pA
Crosstalk @ f = 1 kHz	R_L = 5 kΩ		−130			−130		dB
Crosstalk @ f = 100 kHz	R_L = 5 kΩ		−93			−93		dB
INPUT CHARACTERISTICS								
Input Voltage Range[1], T_{MIN} to T_{MAX}		−0.2		+4	−0.2		+4	V
Common-Mode Rejection Ratio (CMRR)	V_{CM} = 0 V to 2 V	66	80		69	80		dB
T_{MIN} to T_{MAX}	V_{CM} = 0 V to 2 V	66			66			dB

Parameter	Conditions	A Grade Min	A Grade Typ	A Grade Max	B Grade Min	B Grade Typ	B Grade Max	Unit
Input Impedance								
Differential			$10^{13}\|\|0.5$			$10^{13}\|\|0.5$		Ω||pF
Common Mode			$10^{13}\|\|2.8$			$10^{13}\|\|2.8$		Ω||pF
OUTPUT CHARACTERISTICS								
Output Saturation Voltage[2]								
$V_{OL} - V_{EE}$	I_{SINK} = 20 µA		5	7		5	7	mV
T_{MIN} to T_{MAX}				10			10	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 20 µA		10	14		10	14	mV
T_{MIN} to T_{MAX}				20			20	mV
$V_{OL} - V_{EE}$	I_{SINK} = 2 mA		40	55		40	55	mV
T_{MIN} to T_{MAX}				80			80	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 2 mA		80	110		80	110	mV
T_{MIN} to T_{MAX}				160			160	mV
$V_{OL} - V_{EE}$	I_{SINK} = 15 mA		300	500		300	500	mV
T_{MIN} to T_{MAX}				1000			1000	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 15 mA		800	1500		800	1500	mV
T_{MIN} to T_{MAX}				1900			1900	mV
Operating Output Current		15			15			mA
T_{MIN} to T_{MAX}		12			12			mA
Capacitive Load Drive			350			350		pF
POWER SUPPLY								
Quiescent Current, T_{MIN} to T_{MAX}			1.24	1.6		1.24	1.6	mA
Power Supply Rejection	V+ = 5 V to 15 V	66	80		70	80		dB
T_{MIN} to T_{MAX}		66			70			dB

[1] This is a functional specification. Amplifier bandwidth decreases when the input common-mode voltage is driven in the range (V+ – 1 V) to V+. Common-mode error voltage is typically less than 5 mV with the common-mode voltage set at 1 V below the positive supply.

[2] $V_{OL} - V_{EE}$ is defined as the difference between the lowest possible output voltage (V_{OL}) and the negative voltage supply rail (V_{EE}). $V_{CC} - V_{OH}$ is defined as the difference between the highest possible output voltage (V_{OH}) and the positive supply voltage (V_{CC}).

V_S = ±5 V @ T_A = 25°C, V_{CM} = 0 V, V_{OUT} = 0 V, unless otherwise noted.

Table 2.

		A Grade			B Grade			
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
DC PERFORMANCE								
Initial Offset			0.1	0.8		0.1	0.4	mV
Maximum Offset Over Temperature			0.5	1.5		0.5	1	mV
Offset Drift			2			2		µV/°C
Input Bias Current	V_{CM} = −5 V to +4 V		2	25		2	10	pA
At T_{MAX}			0.5	5		0.5	2.5	nA
Input Offset Current			2	20		2	10	pA
At T_{MAX}			0.5			0.5		nA
Open-Loop Gain	V_{OUT} = −4 V to +4 V							
	R_L = 100 kΩ	400	1000		400	1000		V/mV
T_{MIN} to T_{MAX}		400			400			V/mV
	R_L = 10 kΩ	80	150		80	150		V/mV
T_{MIN} to T_{MAX}		80			80			V/mV
	R_L = 1 kΩ	20	30		20	30		V/mV
T_{MIN} to T_{MAX}		10			10			V/mV
NOISE/HARMONIC PERFORMANCE								
Input Voltage Noise								
f = 0.1 Hz to 10 Hz			2			2		µV p-p
f = 10 Hz			25			25		nV/√Hz
f = 100 Hz			21			21		nV/√Hz
f = 1 kHz			16			16		nV/√Hz
f = 10 kHz			13			13		nV/√Hz
Input Current Noise								
f = 0.1 Hz to 10 Hz			18			18		fA p-p
f = 1 kHz			0.8			0.8		fA/√Hz
Harmonic Distortion	R_L = 10 kΩ							
f = 10 kHz	V_{OUT} = ±4.5 V		−93			−93		dB
DYNAMIC PERFORMANCE								
Unity-Gain Frequency			1.9			1.9		MHz
Full Power Response	V_{OUT} p-p = 9 V		105			105		kHz
Slew Rate			3			3		V/µs
Settling Time								
to 0.1%	V_{OUT} = 0 V to ±4.5 V		1.4			1.4		µs
to 0.01%	V_{OUT} = 0 V to ±4.5 V		1.8			1.8		µs
MATCHING CHARACTERISTICS								
Initial Offset				1.0			0.5	mV
Maximum Offset Over Temperature				3			2	mV
Offset Drift			3			3		µV/°C
Input Bias Current				25			10	pA
Crosstalk @ f = 1 kHz	R_L = 5 kΩ		−130			−130		dB
Crosstalk @ f = 100 kHz	R_L = 5 kΩ		−93			−93		dB
INPUT CHARACTERISTICS								
Input Voltage Range[1], T_{MIN} to T_{MAX}		−5.2		+4	−5.2		+4	V
Common-Mode Rejection Ratio (CMRR)	V_{CM} = −5 V to +2 V	66	80		69	80		dB
T_{MIN} to T_{MAX}	V_{CM} = −5 V to +2 V	66			66			dB
Input Impedance								
Differential			10^{13}‖0.5			10^{13}‖0.5		Ω‖pF
Common Mode			10^{13}‖2.8			10^{13}‖2.8		Ω‖pF

Parameter	Conditions	A Grade			B Grade			Unit
		Min	Typ	Max	Min	Typ	Max	
OUTPUT CHARACTERISTICS								
Output Saturation Voltage[2]								
$V_{OL} - V_{EE}$	I_{SINK} = 20 µA		5	7		5	7	mV
T_{MIN} to T_{MAX}				10			10	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 20 µA		10	14		10	14	mV
T_{MIN} to T_{MAX}				20			20	mV
$V_{OL} - V_{EE}$	I_{SINK} = 2 mA		40	55		40	55	mV
T_{MIN} to T_{MAX}				80			80	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 2 mA		80	110		80	110	mV
T_{MIN} to T_{MAX}				160			160	mV
$V_{OL} - V_{EE}$	I_{SINK} = 15 mA		300	500		300	500	mV
T_{MIN} to T_{MAX}				1000			1000	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 15 mA		800	1500		800	1500	mV
T_{MIN} to T_{MAX}				1900			1900	mV
Operating Output Current		15			15			mA
T_{MIN} to T_{MAX}		12			12			mA
Capacitive Load Drive			350			350		pF
POWER SUPPLY								
Quiescent Current, T_{MIN} to T_{MAX}			1.3	1.6		1.3	1.6	mA
Power Supply Rejection	V_{SY} = ±5 V to ±15 V	66	80		70	80		dB
T_{MIN} to T_{MAX}		66			70			dB

[1] This is a functional specification. Amplifier bandwidth decreases when the input common-mode voltage is driven in the range (V+ – 1 V) to V+. Common-mode error voltage is typically less than 5 mV with the common-mode voltage set at 1 V below the positive supply.

[2] $V_{OL} - V_{EE}$ is defined as the difference between the lowest possible output voltage (V_{OL}) and the negative voltage supply rail (V_{EE}). $V_{CC} - V_{OH}$ is defined as the difference between the highest possible output voltage (V_{OH}) and the positive supply voltage (V_{CC}).

V_S = ±15 V @ T_A = 25°C, V_{CM} = 0 V, V_{OUT} = 0 V, unless otherwise noted.

Table 3.

		A Grade			B Grade			
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
DC PERFORMANCE								
Initial Offset			0.4	2		0.3	1.5	mV
Maximum Offset Over Temperature			0.5	3		0.5	2.5	mV
Offset Drift			2			2		µV/°C
Input Bias Current	V_{CM} = 0 V		2	25		2	12	pA
	V_{CM} = −10 V		40			40		pA
At T_{MAX}	V_{CM} = 0 V		0.5	5		0.5	2.5	nA
Input Offset Current			2	20		2	12	pA
At T_{MAX}			0.5			0.5		nA
Open-Loop Gain	V_{OUT} = −10 V to +10 V							
	R_L = 100 kΩ	500	2000		500	2000		V/mV
T_{MIN} to T_{MAX}		500			500			V/mV
	R_L = 10 kΩ	100	500		100	500		V/mV
T_{MIN} to T_{MAX}		100			100			V/mV
	R_L = 1 kΩ	30	45		30	45		V/mV
T_{MIN} to T_{MAX}		20			20			V/mV
NOISE/HARMONIC PERFORMANCE								
Input Voltage Noise								
f = 0.1 Hz to 10 Hz			2			2		µV p-p
f = 10 Hz			25			25		nV/√Hz
f = 100 Hz			21			21		nV/√Hz
f = 1 kHz			16			16		nV/√Hz
f = 10 kHz			13			13		nV/√Hz
Input Current Noise								
f = 0.1 Hz to 10 Hz			18			18		fA p-p
f = 1 kHz			0.8			0.8		fA/√Hz
Harmonic Distortion	R_L = 10 kΩ							
f = 10 kHz	V_{OUT} = ±10 V		−85			−85		dB
DYNAMIC PERFORMANCE								
Unity-Gain Frequency			1.9			1.9		MHz
Full Power Response	V_{OUT} p-p = 20 V		45			45		kHz
Slew Rate			3			3		V/µs
Settling Time								
to 0.1%	V_{OUT} = 0 V to ±10 V		4.1			4.1		µs
to 0.01%	V_{OUT} = 0 V to ±10 V		4.5			4.5		µs
MATCHING CHARACTERISTICS								
Initial Offset				3			2	mV
Maximum Offset Over Temperature				4			2.5	mV
Offset Drift			3			3		µV/°C
Input Bias Current				25			12	pA
Crosstalk @ f = 1 kHz	R_L = 5 kΩ		−130			−130		dB
Crosstalk @ f = 100 kHz	R_L = 5 kΩ		−93			−93		dB
INPUT CHARACTERISTICS								
Input Voltage Range[1], T_{MIN} to T_{MAX}		−15.2		+14	−15.2		+14	V
Common-Mode Rejection Ratio (CMRR)	V_{CM} = −15 V to +12 V	70	80		74	90		dB
T_{MIN} to T_{MAX}	V_{CM} = −15 V to +12 V	70			74			dB
Input Impedance								
Differential			10^{13}||0.5			10^{13}||0.5		Ω||pF
Common Mode			10^{13}||2.8			10^{13}||2.8		Ω||pF

Parameter	Conditions	A Grade Min	A Grade Typ	A Grade Max	B Grade Min	B Grade Typ	B Grade Max	Unit
OUTPUT CHARACTERISTICS								
Output Saturation Voltage[2]								
$V_{OL} - V_{EE}$	I_{SINK} = 20 µA		5	7		5	7	mV
T_{MIN} to T_{MAX}				10			10	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 20 µA		10	14		10	14	mV
T_{MIN} to T_{MAX}				20			20	mV
$V_{OL} - V_{EE}$	I_{SINK} = 2 mA		40	55		40	55	mV
T_{MIN} to T_{MAX}				80			80	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 2 mA		80	110		80	110	mV
T_{MIN} to T_{MAX}				160			160	mV
$V_{OL} - V_{EE}$	I_{SINK} = 15 mA		300	500		300	500	mV
T_{MIN} to T_{MAX}				1000			1000	mV
$V_{CC} - V_{OH}$	I_{SOURCE} = 15 mA		800	1500		800	1500	mV
T_{MIN} to T_{MAX}				1900			1900	mV
Operating Output Current		20			20			mA
T_{MIN} to T_{MAX}		15			15			mA
Capacitive Load Drive			350			350		pF
POWER SUPPLY								
Quiescent Current, T_{MIN} to T_{MAX}			1.4	1.8		1.4	1.8	mA
Power Supply Rejection	V_{SY} = ±5 V to ±15 V	70	80		70	80		dB
T_{MIN} to T_{MAX}		70			70			dB

[1] This is a functional specification. Amplifier bandwidth decreases when the input common-mode voltage is driven in the range (V+ – 1 V) to V+. Common-mode error voltage is typically less than 5 mV with the common-mode voltage set at 1 V below the positive supply.

[2] $V_{OL} - V_{EE}$ is defined as the difference between the lowest possible output voltage (V_{OL}) and the negative voltage supply rail (V_{EE}). $V_{CC} - V_{OH}$ is defined as the difference between the highest possible output voltage (V_{OH}) and the positive supply voltage (V_{CC}).

ABSOLUTE MAXIMUM RATINGS

Table 4.

Parameter	Rating
Supply Voltage	±18 V
Internal Power Dissipation	
8-Lead PDIP (N)	Observe derating curves
8-Lead SOIC_N (R)	Observe derating curves
8-Lead MSOP (RM)	Observe derating curves
Input Voltage[1]	((V+) + 0.2 V) to ((V−) − 20 V)
Output Short-Circuit Duration	Indefinite
Differential Input Voltage	±30 V
Storage Temperature Range (N)	−65°C to +125°C
Storage Temperature Range (R, RM)	−65°C to +150°C
Operating Temperature Range	
A Grade and B Grade	−40°C to +85°C
Lead Temperature (Soldering, 60 sec)	260°C

[1] See the Input Characteristics section.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

THERMAL RESISTANCE

θ_{JA} is specified for the worst-case conditions, that is, a device soldered in a circuit board for surface-mount packages.

Table 5. Thermal Resistance

Package Type	θ_{JA}	Unit
8-lead PDIP (N)	90	°C/W
8-lead SOIC_N (R)	160	°C/W
8-lead MSOP (RM)	190	°C/W

MAXIMUM POWER DISSIPATION

The maximum power that can be safely dissipated by the AD822 is limited by the associated rise in junction temperature. For plastic packages, the maximum safe junction temperature is 145°C. If these maximums are exceeded momentarily, proper circuit operation is restored as soon as the die temperature is reduced. Leaving the device in the overheated condition for an extended period can result in device burnout. To ensure proper operation, it is important to observe the derating curves shown in Figure 27.

While the AD822 is internally short-circuit protected, this may not be sufficient to guarantee that the maximum junction temperature is not exceeded under all conditions. With power supplies ±12 V (or less) at an ambient temperature of 25°C or less, if the output node is shorted to a supply rail, then the amplifier is not destroyed, even if this condition persists for an extended period.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

TYPICAL PERFORMANCE CHARACTERISTICS

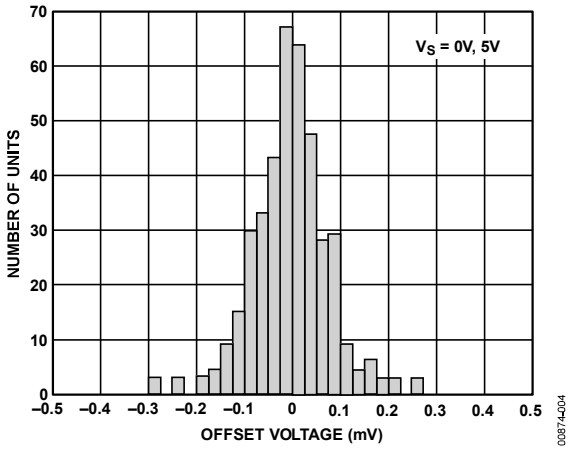


Figure 4. Typical Distribution of Offset Voltage (390 Units)

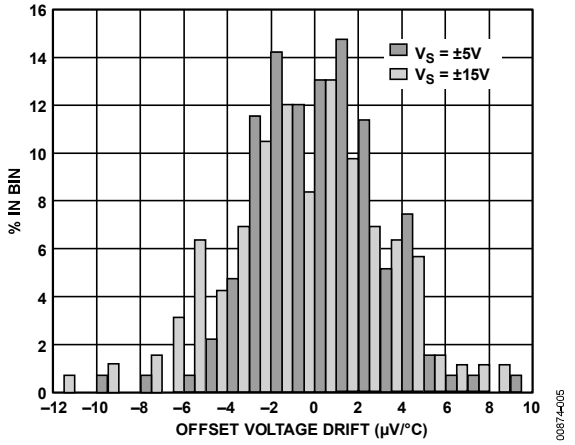


Figure 5. Typical Distribution of Offset Voltage Drift (100 Units)

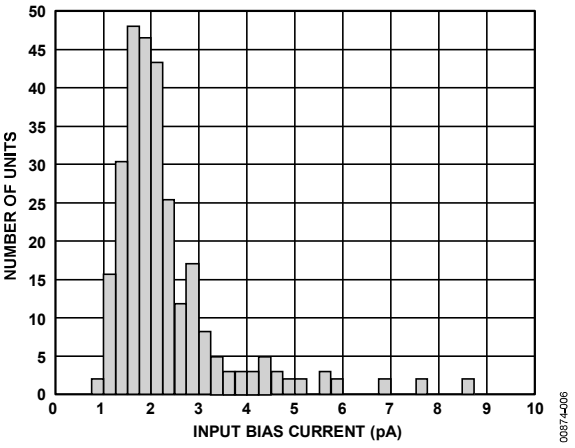


Figure 6. Typical Distribution of Input Bias Current (213 Units)

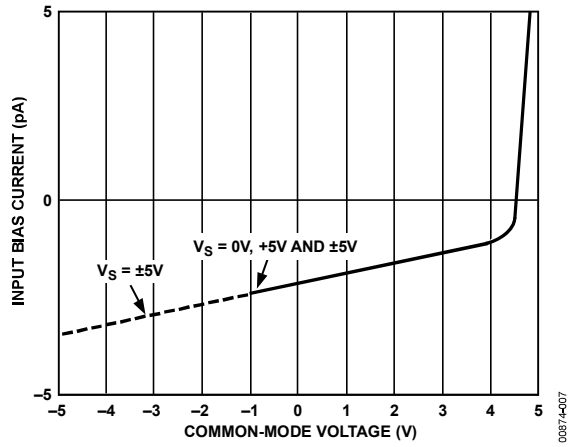


Figure 7. Input Bias Current vs. Common-Mode Voltage; V_S = 5 V, 0 V, and V_S = ±5 V

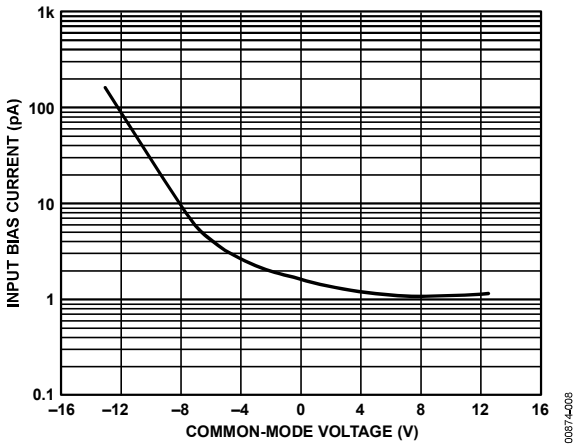


Figure 8. Input Bias Current vs. Common-Mode Voltage; V_S = ±15 V

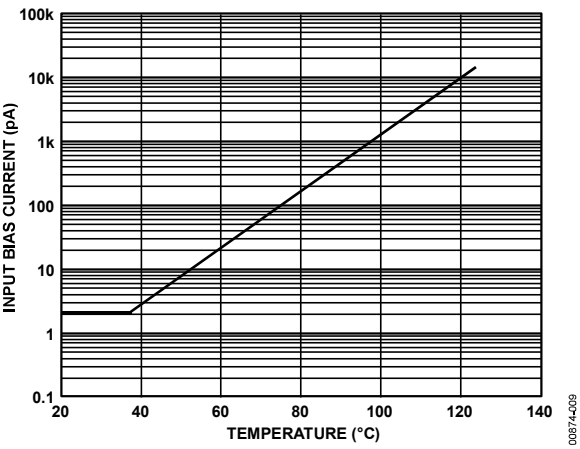


Figure 9. Input Bias Current vs. Temperature; V_S = 5 V, V_{CM} = 0 V

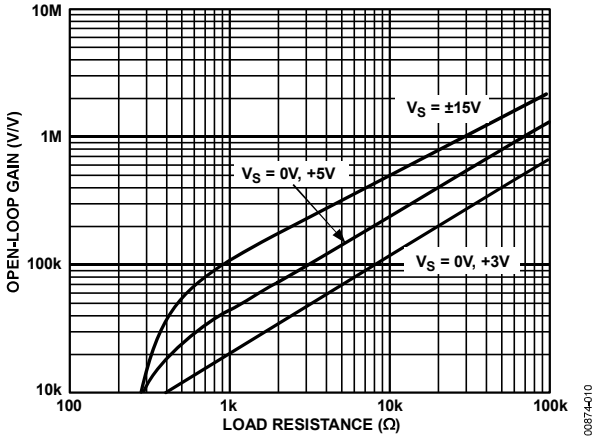


Figure 10. Open-Loop Gain vs. Load Resistance

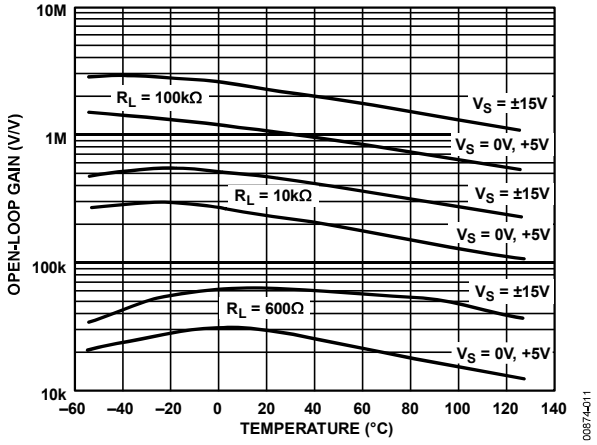


Figure 11. Open-Loop Gain vs. Temperature

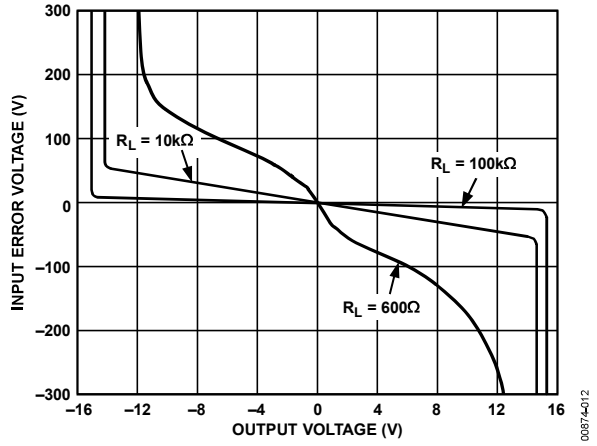


Figure 12. Input Error Voltage vs. Output Voltage for Resistive Loads

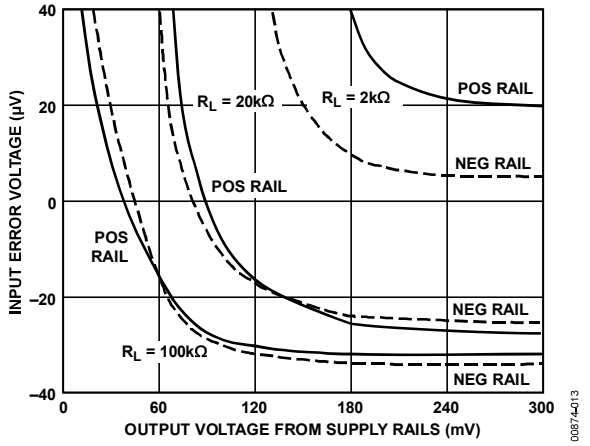


Figure 13. Input Error Voltage with Output Voltage Within 300 mV of Either Supply Rail for Various Resistive Loads; V_S = ±5 V

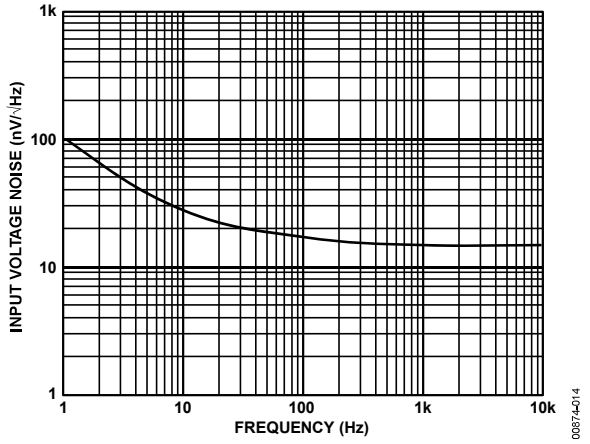


Figure 14. Input Voltage Noise vs. Frequency

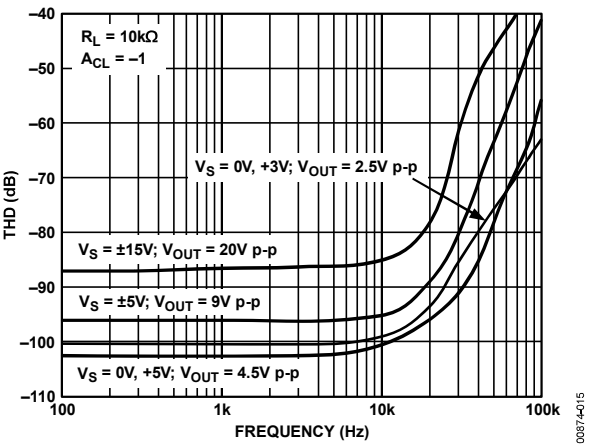


Figure 15. Total Harmonic Distortion (THD) vs. Frequency

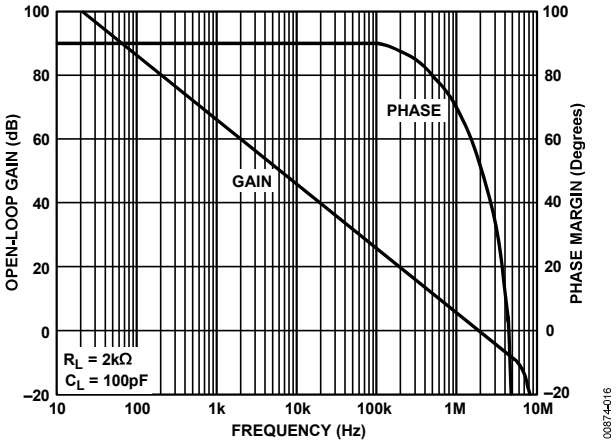


Figure 16. Open-Loop Gain and Phase Margin vs. Frequency

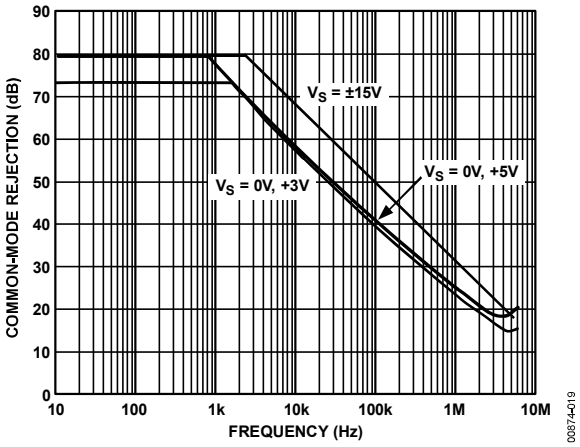


Figure 19. Common-Mode Rejection vs. Frequency

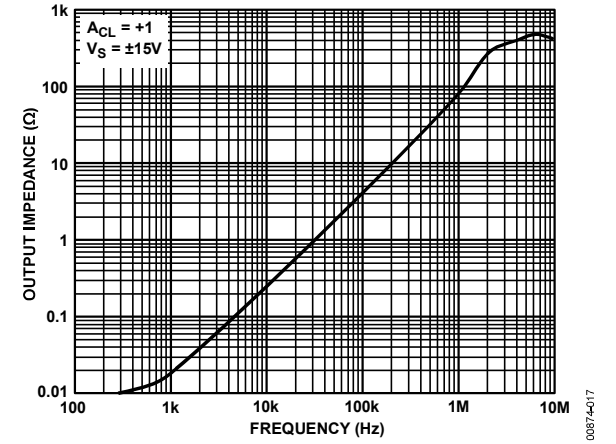


Figure 17. Output Impedance vs. Frequency

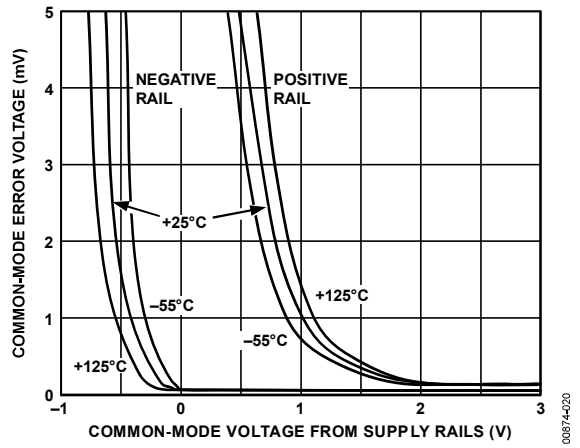


Figure 20. Absolute Common-Mode Error vs. Common-Mode Voltage from Supply Rails (V_S – V_{CM})

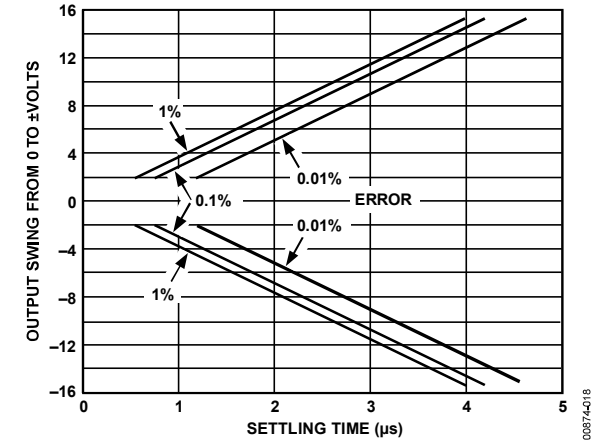


Figure 18. Output Swing and Error vs. Settling Time

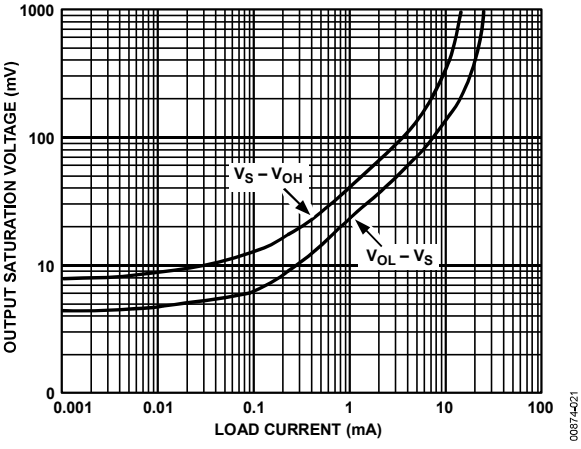


Figure 21. Output Saturation Voltage vs. Load Current

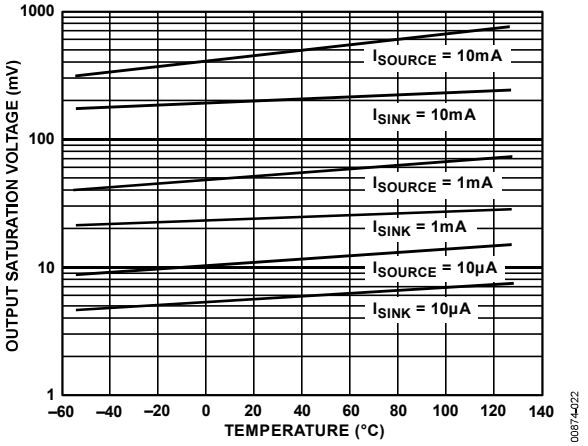


Figure 22. Output Saturation Voltage vs. Temperature

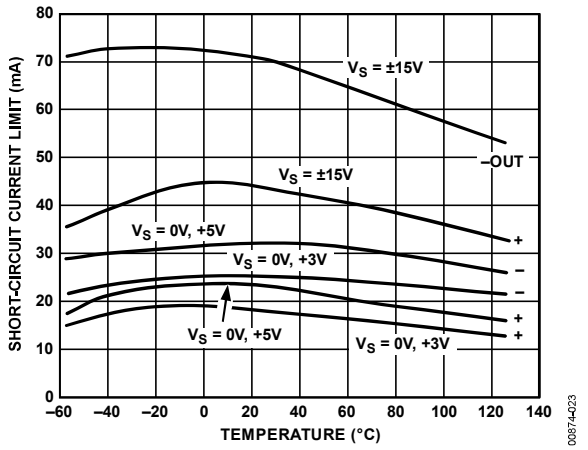


Figure 23. Short-Circuit Current Limit vs. Temperature

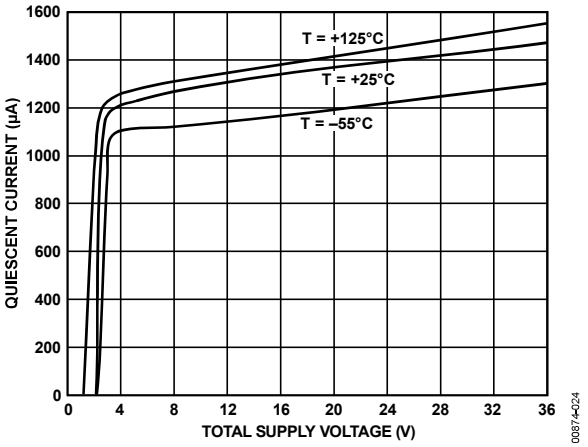


Figure 24. Quiescent Current vs. Supply Voltage vs. Temperature

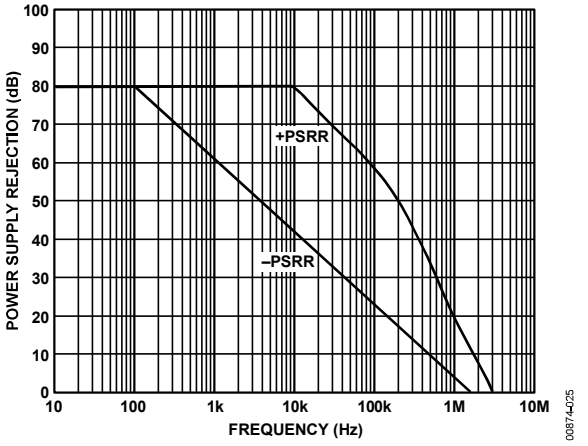


Figure 25. Power Supply Rejection vs. Frequency

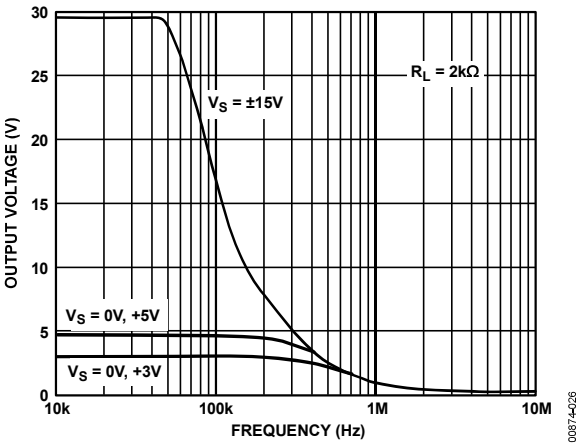


Figure 26. Large Signal Frequency Response

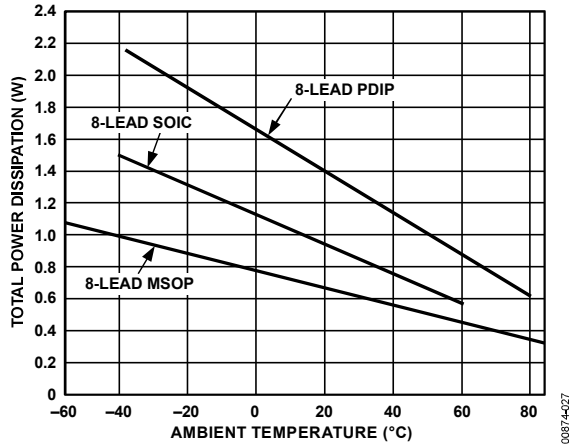
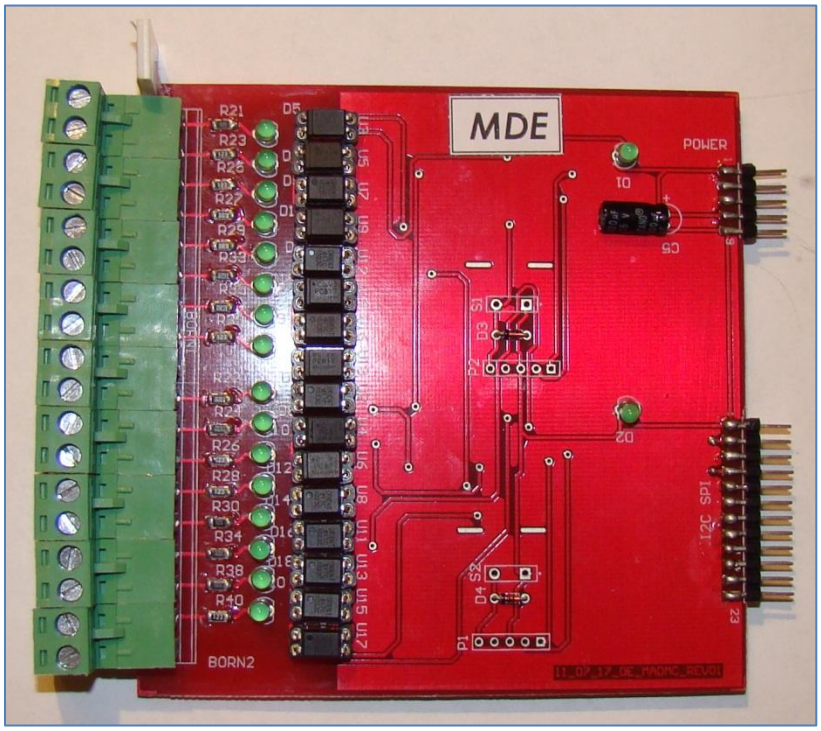


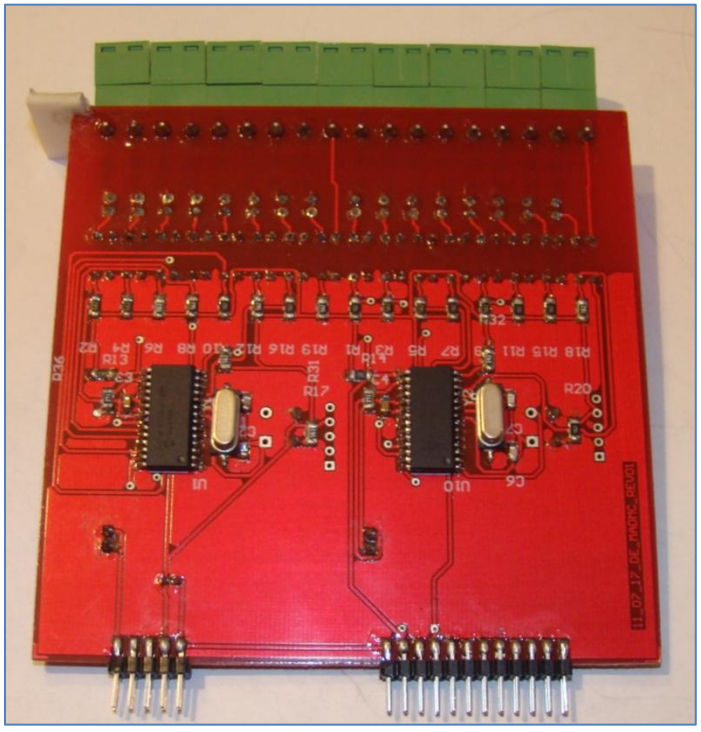
Figure 27. Maximum Power Dissipation vs. Temperature for Packages

ANEXO III

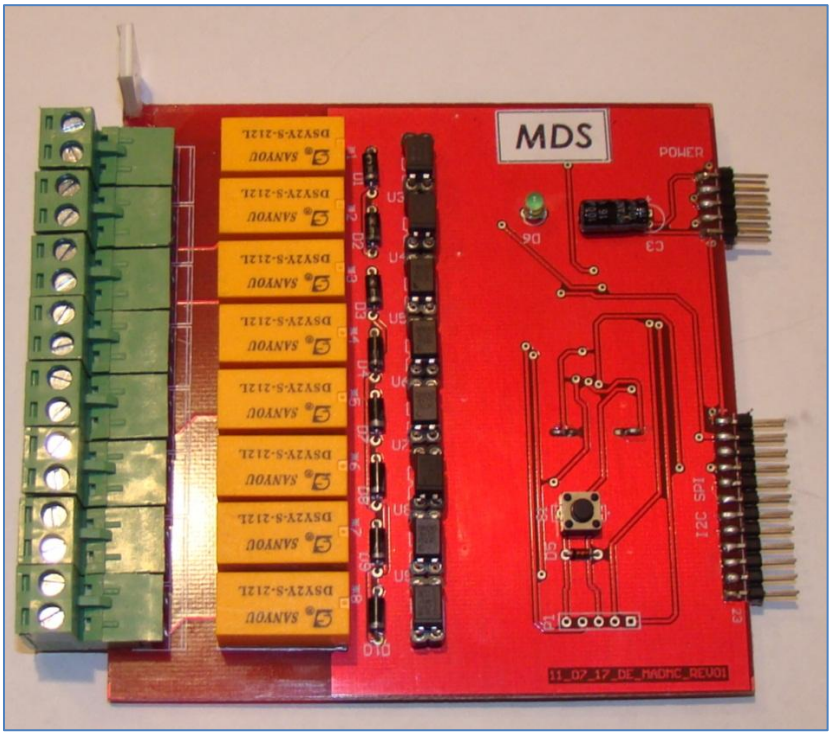
FOTOGRAFÍAS DEL MÓDULO

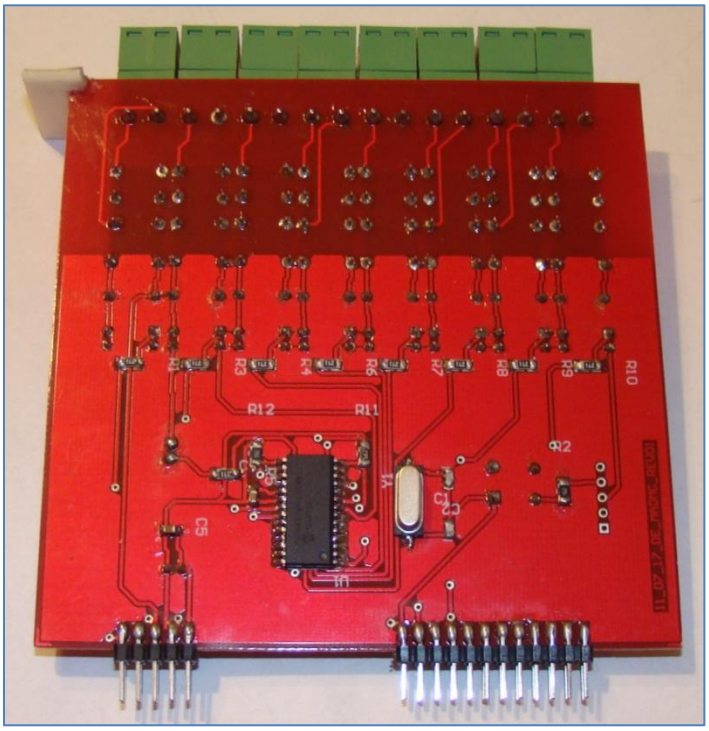
TARJETA DE ENTRADAS DIGITALES



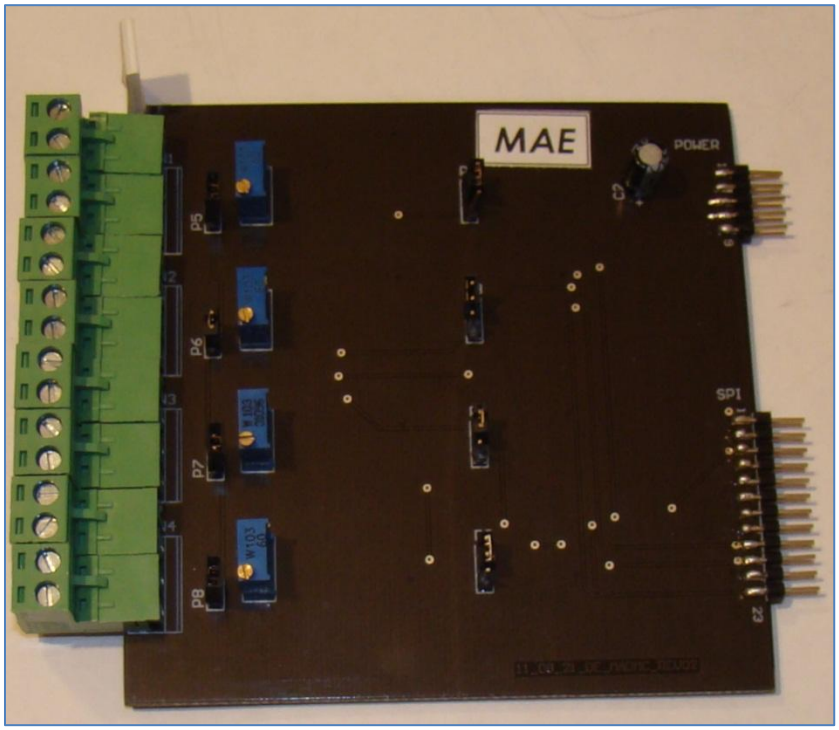


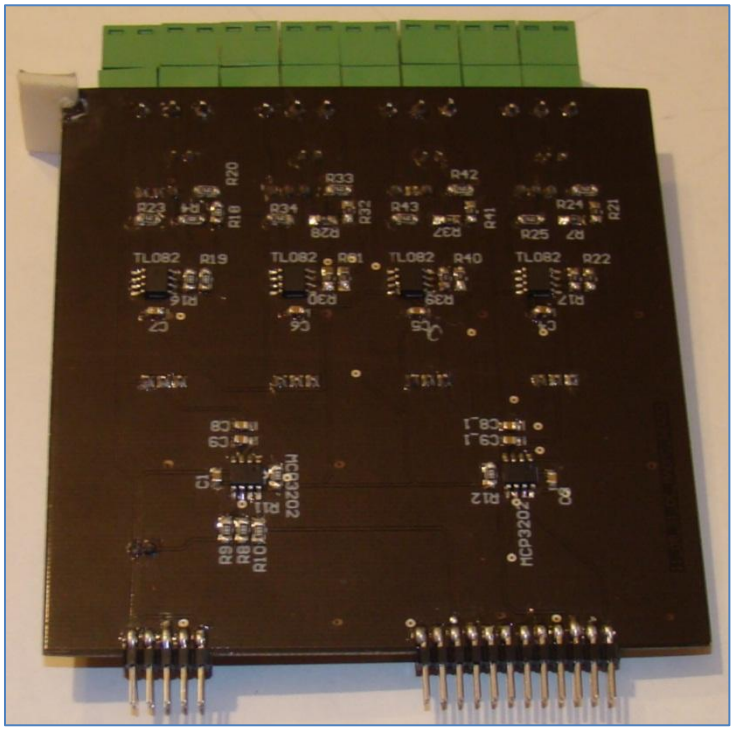
TARJETA DE SALIDAS DE RELÉ



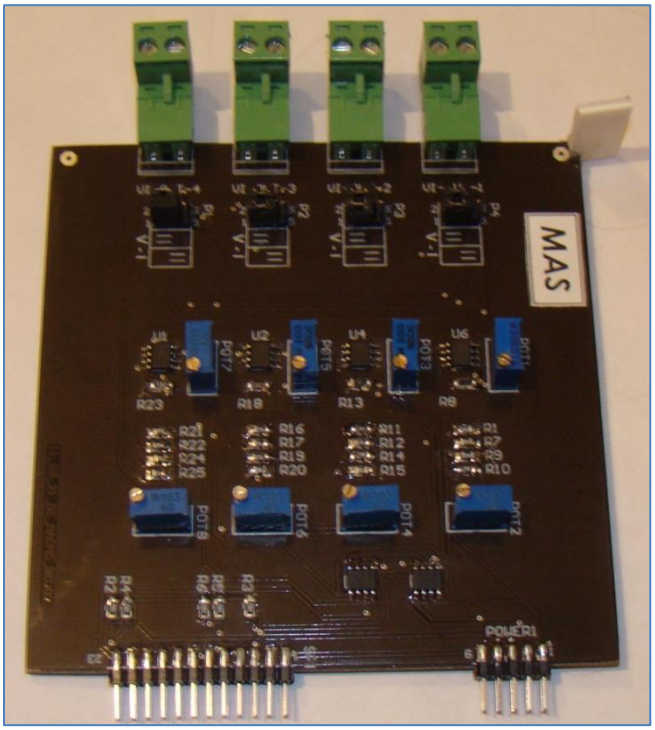


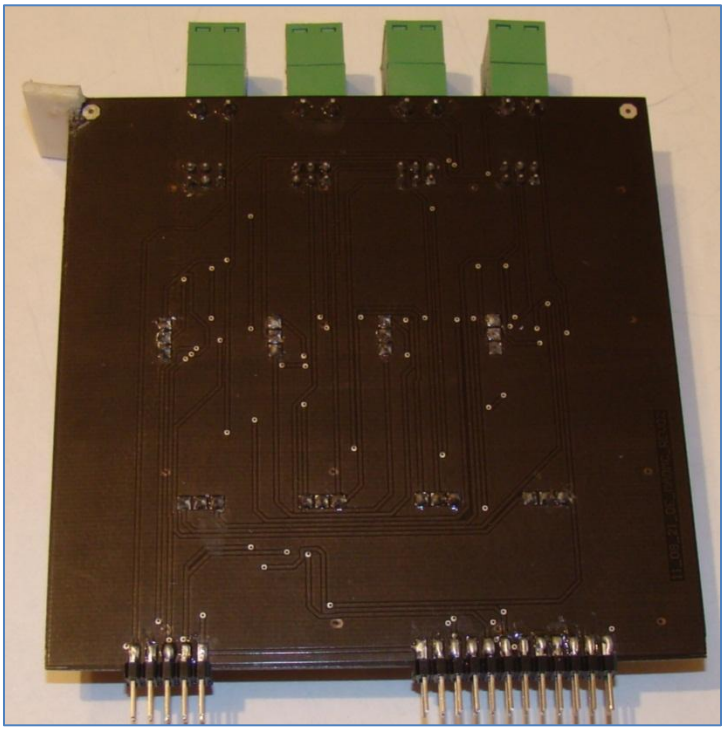
TARJETA DE ENTRADAS ANÁLOGAS



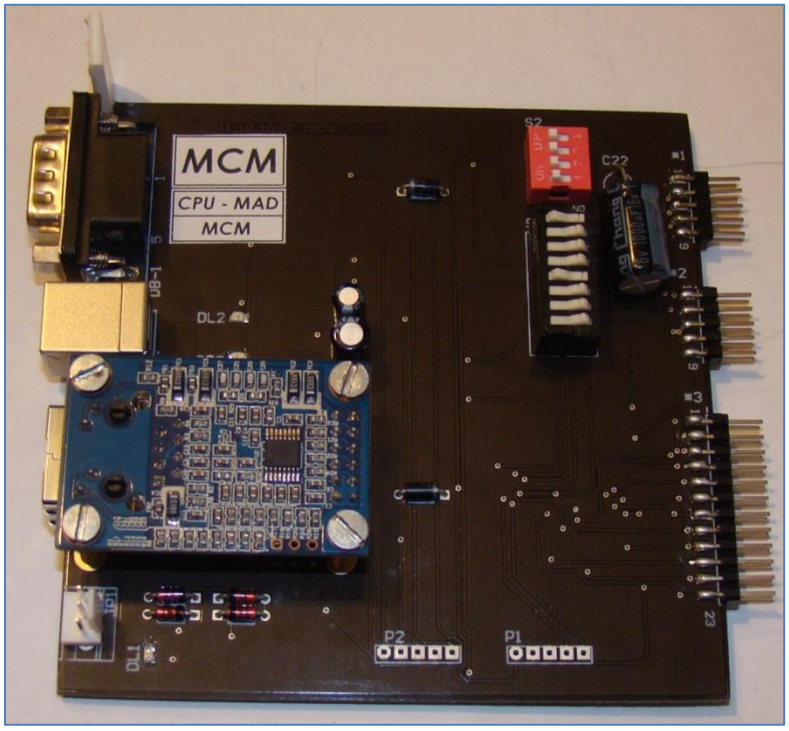


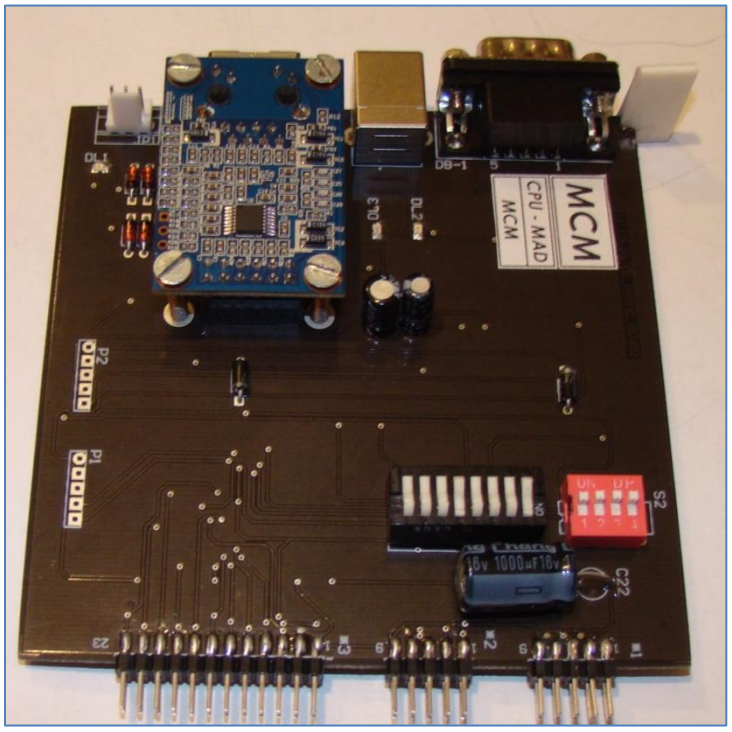
TARJETA DE SALIDAS ANÁLOGAS



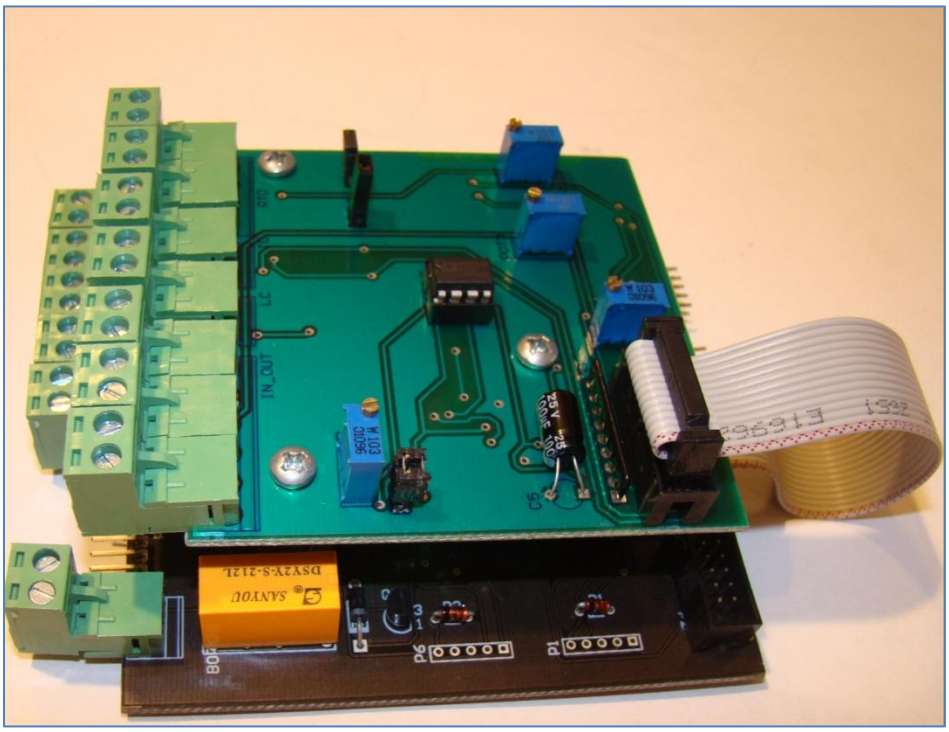


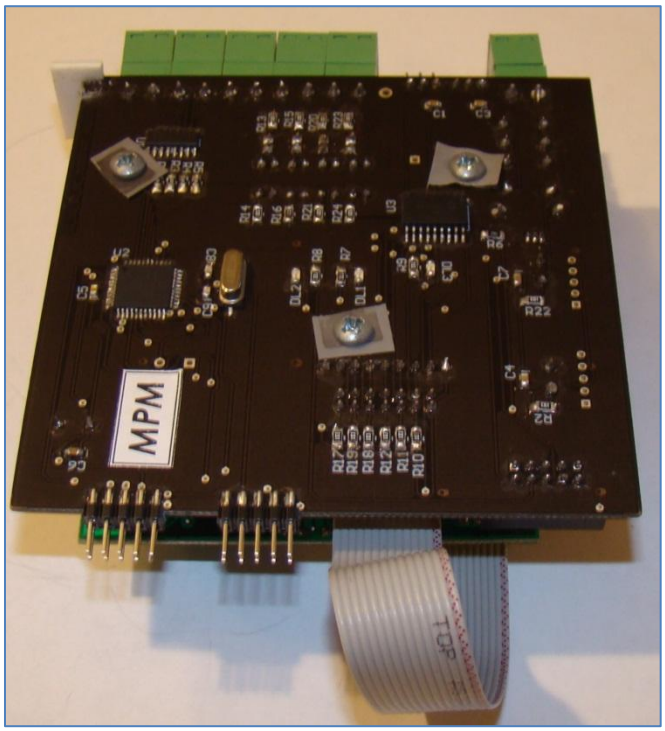
TARJETA MASTER DE COMUNICACIONES



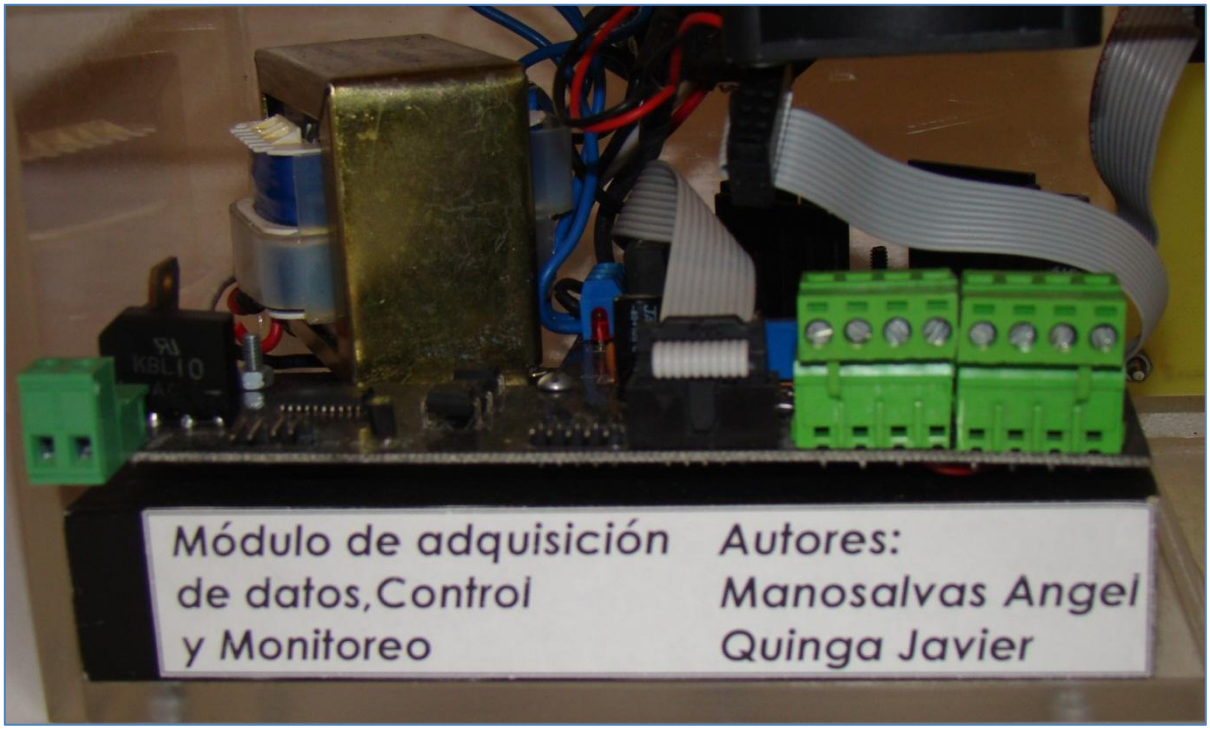


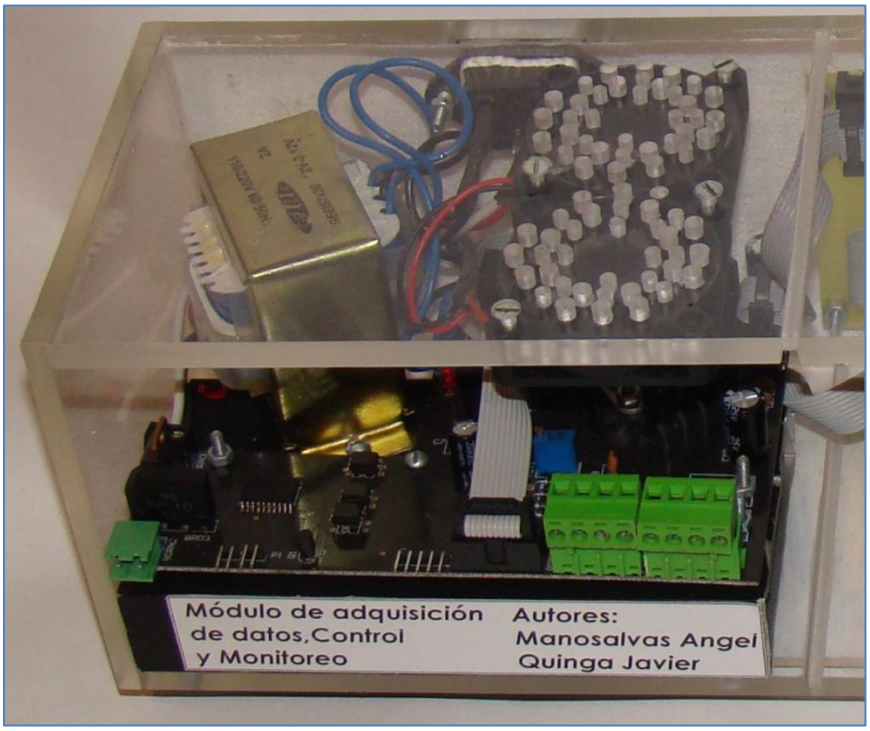
TARJETA MASTER DE PROCESOS



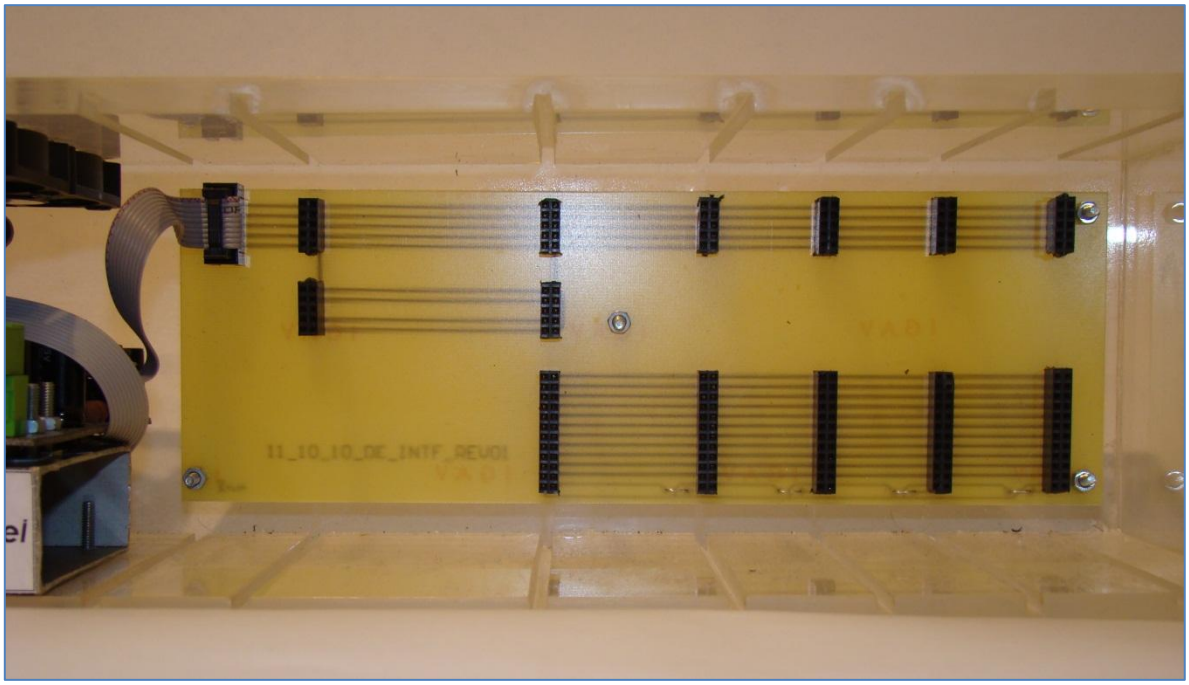


TARJETA FUENTE Y CIRCUITOS DE FUERZA





TARJETA BACK PANEL



ANEXO IV

FIRMWARES DE MICROCONTROLADORES

Programa del microcontrolar PIC16F887 esclavo Modbus de la tarjeta MCM.

```
////////////////////////////////////////////////////////////////////////
////                            VMOSBUS_I2C_SPI_MAD.c                ////
////                                                                 ////
////                                                                 ////
//// Realizado con el compilador CCS PCWH 4.014                     ////
////                                                                 ////
//// Javier Quinga Jarrín, Angel Manosalvas Benalcázar              ////
////////////////////////////////////////////////////////////////////////


//// Configuración del Microcontrolador
////////////////////////////////////////////////////////////////////////

//#define USE_WITH_PC 0
#include <16f887.h>
//#device *=16
#fuses INTRC, NOWDT, NOLVP, NOBROWNOUT, NOPROTECT, PUT
#use delay(clock=4000000)
//Configuración I2C
#use i2c(Master,sda=PIN_C4,scl=PIN_C3,FORCE_SW)
// RS232 Estándar
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
//#use spi(DI=PIN_C4, DO=PIN_C5, CLK=PIN_C3, BITS=16)
#use spi(MASTER,CLK=PIN_C1, DO=PIN_C5, DI=PIN_C2, BITS=8, baud = 10000, MODE=2, MSB_FIRST,
stream=DA_spi)
#byte PORTA = 5
byte rcv2;
BYTE data;
 byte pwm1, pwm0;
int8 dato1, dato2, dato3, dato4, dato5, dato6, dato7, dato8;
int8 dato11, dato21, dato31, dato41, dato51, dato61, dato71, dato81;

//// Configuración del Modbus
////////////////////////////////////////////////////////////////////////

#define OSC_4MHZ    0x60
#define MODBUS_TYPE MODBUS_TYPE_SLAVE
#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_C6   // Data transmit pin
#define MODBUS_SERIAL_RX_PIN PIN_C7   // Data receive pin
//The following should be defined for RS485 communication
#define MODBUS_SERIAL_ENABLE_PIN   PIN_E0   // Controls DE pin for RS485
//#define MODBUS_SERIAL_RX_ENABLE    PIN_A1   // Controls RE pin for RS485
#else
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
#endif
//#include "modbus.c"
#include "MI_MODBUS_485.c"
#define MODBUS_ADDRESS 0x0A
#define EEPROM_ADDRESS long int

//// Subrutinas para escribir vís SPI
////////////////////////////////////////////////////////////////////////

void write_SPI0()
{
//spi_out=make16(modbus_rx.data[2],modbus_rx.data[3]);
  output_low(PIN_D1);
    delay_ms(20);
   //spi_xfer(modbus_rx.data[3]);
   //spi_xfer(DA_spi,modbus_rx.data[2]);
   spi_xfer(DA_spi,pwm0);
    //spi_xfer(DA_spi, an1);
   //spi_xfer(DA_spi,an2 );
```

```
   //spi_xfer(DA_spi, modbus_rx.data[3]);
   spi_xfer(DA_spi,pwm1);
   //spi_xfer(0x01700);/* |4 bits dummy|10bits DA|2 bits x */
   //spi_xfer(DA_spi,spi_out);/* |4 bits dummy|10bits DA|2 bits x */
    delay_ms(20);
   output_high(PIN_D1);
}
void write_SPI1()
{
//spi_out=make16(modbus_rx.data[2],modbus_rx.data[3]);
  output_low(PIN_D0);
    delay_ms(20);
   //spi_xfer(modbus_rx.data[3]);
   //spi_xfer(DA_spi,modbus_rx.data[2]);
   spi_xfer(DA_spi,pwm0);
    //spi_xfer(DA_spi, an1);
   //spi_xfer(DA_spi,an2 );
   //spi_xfer(DA_spi, modbus_rx.data[3]);
   spi_xfer(DA_spi,pwm1);
   //spi_xfer(0x01700);/* |4 bits dummy|10bits DA|2 bits x */
   //spi_xfer(DA_spi,spi_out);/* |4 bits dummy|10bits DA|2 bits x */
    delay_ms(20);
   output_high(PIN_D0);
}

//// Subrutinas para leer vía SPI
////////////////////////////////////////////////////////////////////////

void read_SPI0()
{
      output_low(PIN_D4);
      delay_ms(20);
      spi_xfer(DA_spi,1);
      dato1 = spi_xfer(DA_spi,0x80);
      dato2 = spi_xfer(DA_spi,0);
      dato3 = dato1&15;
      delay_ms(20);
      output_high(PIN_D4);
}

void read_SPI1()
{
      output_low(PIN_D4);
      delay_ms(20);
      spi_xfer(DA_spi,1);
      dato5 = spi_xfer(DA_spi,0xC0);
      dato6 = spi_xfer(DA_spi);
      dato7 = dato5&15;
      delay_ms(20);
      output_high(PIN_D4);
}

 void read_SPI01()
{
      output_low(PIN_D5);
      delay_ms(20);
      spi_xfer(DA_spi,1);
      dato11 = spi_xfer(DA_spi,0x80);
      dato21 = spi_xfer(DA_spi,0);
      dato31 = dato11&15;
      delay_ms(20);
      output_high(PIN_D5);
}

void read_SPI11()
{
      output_low(PIN_D5);
      delay_ms(20);
      spi_xfer(DA_spi,1);
```

```
      dato51 = spi_xfer(DA_spi,0xC0);
      dato61 = spi_xfer(DA_spi);
      dato71 = dato51&15;
      delay_ms(20);
      output_high(PIN_D5);
}

//// Subrutinas para escribir vía I2C
////////////////////////////////////////////////////////////////////////

void write_ext_eeprom(long int address, BYTE data)
{
   short int status;
  // output_b (modbus_rx.data[3]);
   i2c_start();                     //Inicializa la transmisión
   i2c_write(248);                  //Escribe la palabra de control (dirección 0h + 0 para
escritura)
   delay_ms(10);
   i2c_write(modbus_rx.data[3]);    //Dato a escribir
   delay_ms(10);
   i2c_stop();                      //Finalización de la transmisión.
   i2c_start();                     //Reinicio
   status=i2c_write(248);           //Lectura del bit ACK, para evitar escrituras incorrectas


  while(status==1)                  //Si es 1 esperar a que responda el esclavo
   {
      i2c_start();
      status=i2c_write(248);
   }
}


//// Subrutinas para leer vía I2C
////////////////////////////////////////////////////////////////////////

BYTE read_ext_eeprom1(long int address) {

   i2c_start();                     //Inicializa la transmisión
   i2c_write(120);                  //Escribe la palabra de control (dirección 0h + 0 para
escritura)
   i2c_start();                     //Reinicio
   i2c_write(121);                  //Escribe la palabra de control (dirección 0h + 1 para lectura)
   data=i2c_read(0);                //lectura del dato
   i2c_stop();                      //Finalización de la transmisión.

   return(data);
}

BYTE read_ext_eeprom2(long int address) {

   i2c_start();                     //Inicializa la transmisión
   i2c_write(124);                  //Escribe la palabra de control (dirección 0h + 0 para
escritura)
   i2c_start();                     //Reinicio
   i2c_write(125);                  //Escribe la palabra de control (dirección 0h + 1 para lectura)
   data=i2c_read(0);                //lectura del dato
   i2c_stop();                      //Finalización de la transmisión.

   return(data);
}

/*This function may come in handy for you since MODBUS uses MSB first.*/
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);
}
```

```
//// Programa principal, responde  a las funciones Modbus
////////////////////////////////////////////////////////////////////////

void main()
{
   int8 coils = 0b00000101;
   int8 inputs = 0b00001001;
   int16 hold_regs[] = {0x8800,0x7700,0x6600,0x5500,0x4400,0x3300,0x2200,0x1100};
   int16 input_regs[] = {0x1100,0x2200,0x3300,0x4400,0x5500,0x6600,0x7700,0x8800};
   int16 event_count = 0;

   modbus_init();

   while(TRUE)
   {


   int8 valor=0, dato1_I2C, dato2_I2C;
   EEPROM_ADDRESS address;;

     if (modbus_rx.func==4)
           {

              delay_ms (10);
              read_SPI0();
              delay_ms (10);
              read_SPI1();
              delay_ms (10);
              read_SPI01();
              delay_ms (10);
              read_SPI11();
             }

      dato1_I2C=READ_EXT_EEPROM1( address);

      dato2_I2C=READ_EXT_EEPROM2( address);

      //WRITE_EXT_EEPROM(248, modbus_rx.data[3]);

//// Registros Modbus
////////////////////////////////////////////////////////////////////////

   input_regs[0]=make16(00,00);
   input_regs[1]=make16(dato2,dato3);
   input_regs[2]=make16(dato6,dato7);
   input_regs[3]=make16(dato21,dato31);
   input_regs[4]=make16(dato61,dato71);
   input_regs[5]=make16(00,00);
   input_regs[6]=make16(00,00);
   input_regs[7]=make16(00,00);

   hold_regs[0]=make16(PORTA,00);
   hold_regs[1]=make16(dato1_I2C,dato2_I2C);
   hold_regs[2]=0x1200;
   hold_regs[3]=0x1200;
   hold_regs[4]=0x1200;
   hold_regs[5]=0x1600;
   hold_regs[6]=0x2000;
   hold_regs[7]=0x2400;

//// Funciones Modbus
////////////////////////////////////////////////////////////////////////

      while(!modbus_kbhit());

      //check address against our address, 0 is broadcast
      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)
      {
         switch(modbus_rx.func)
```

```
         {
            case FUNC_READ_COILS:    //read coils
            case FUNC_READ_DISCRETE_INPUT:    //read inputs
               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
               {
                  int8 data;

                  if(modbus_rx.func == FUNC_READ_COILS)
                     data = coils>>(modbus_rx.data[1]);      //move to the starting coil
                  else
                     data = inputs>>(modbus_rx.data[1]);      //move to the starting input

                  data = data & (0xFF>>(8-modbus_rx.data[3]));  //0 out values after quantity

                  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] >= 8 || modbus_rx.data[3]+modbus_rx.data[1] > 8)
                  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:      //write 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
               {
                  //coils are stored msb->lsb so we must use 7-address
                  if(modbus_rx.data[2] == 0xFF)
                     bit_set(coils,7-modbus_rx.data[1]);
                  else
                     bit_clear(coils,7-modbus_rx.data[1]);


modbus_write_single_coil_rsp(MODBUS_ADDRESS,modbus_rx.data[1],((int16)(modbus_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
               {
                  //the registers are stored in little endian format
```

```
                  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[0],modbus_rx.data[1]),
                              make16(modbus_rx.data[2],modbus_rx.data[3]));

                              if(modbus_rx.data[1]== 0)
                               {
                               pwm0 = modbus_rx.data[2]|144;
                               pwm1 = modbus_rx.data[3];
                                write_SPI0();
                              output_high(PIN_D1);
                              delay_ms(10);
                                }

                                else
                               {
                                if(modbus_rx.data[1]== 1)
                               {
                               pwm0 = modbus_rx.data[2];
                               pwm1 = modbus_rx.data[3];
                                WRITE_EXT_EEPROM(248, modbus_rx.data[3]);
                              delay_ms(10);
                                }
                                else
                                {
                                if(modbus_rx.data[1]== 2)
                               {
                               pwm0 = modbus_rx.data[2]|16;
                               pwm1 = modbus_rx.data[3];
                                write_SPI1();
                                output_high(PIN_D0);
                              delay_ms(10);
                                }
                                 }
                                }

                              //output_d (modbus_rx.data[3]);//
                              //WRITE_EXT_EEPROM(248, modbus_rx.data[3]);

//_____________________________________________________________________________//
                              //HASTA AQUI TODO BIEN//
               }
               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);
                     else
                        bit_clear(coils,7-i);
                  }

                  modbus_write_multiple_coils_rsp(MODBUS_ADDRESS,
                                 make16(modbus_rx.data[0],modbus_rx.data[1]),
                                 make16(modbus_rx.data[2],modbus_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:    //We don't support the function, so return exception
               modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
         }
      }
   }
}


//// Fin del programa
////////////////////////////////////////////////////////////////////////
```

Programa del microcontrolar PIC16F887 esclavo Modbus de la tarjeta MPM.

```
////////////////////////////////////////////////////////////////////////
////                            VMOSBUS_I2C_SPI_PRC.c                ////
////                                                                 ////
////                                                                 ////
//// Realizado con el compilador CCS PCWH 4.014                     ////
////                                                                 ////
//// Javier Quinga Jarrín, Angel Manosalvas Benalcázar              ////
////////////////////////////////////////////////////////////////////////

//// Configuración del Microcontrolador
////////////////////////////////////////////////////////////////////////

//#define USE_WITH_PC 0
#include <16f887.h>
//#device *=16
#fuses XT, NOWDT, NOLVP, NOBROWNOUT, NOPROTECT, PUT
#use delay(clock=4000000)
#use i2c(Master,sda=PIN_C4,scl=PIN_C3,FORCE_SW,fast=10000)          //Configuración I2C
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7
#use spi(MASTER,CLK=PIN_B6, DO=PIN_C5, DI=PIN_A5, BITS=8, baud = 10000, MODE=2, MSB_FIRST,
stream=DA_spi)
byte rcv2;
BYTE data;
 byte XXXX, YYYY, pwm1, pwm0, discr_out, an_out01, an_out02, an_out11, an_out12, an_out21, an_out22,
an_out31, an_out32, an_out1, anout1, anout2;
int8 dato1, dato2, dato3, dato4, dato5, dato6, dato7, dato8;
int8 dato11, dato21, dato31, dato41, dato51, dato61, dato71, dato81;
int8 datoRTD1, datoRTD2, datoRTD3, datoRTD4, datoRTD5, datoRTD6;
int8 datoLC1, datoLC2, datoLC3, datoLC4, datoLC5, datoLC6;
int8 dato_I2C;
int8 out_q;
int8 dato_TIT;
int8 in_q, in_q1, servo;
INT8 CONTADOR0;
INT16 CONTADOR1;
#define D4     PIN_D4
#define D5     PIN_D5
#define D6     PIN_D6
#define D7     PIN_D7
#byte T1CON = 0x010

//// Configuración Modbus
////////////////////////////////////////////////////////////////////////

#define OSC_4MHZ    0x60
#define MODBUS_TYPE MODBUS_TYPE_SLAVE
#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_C6
#define MODBUS_SERIAL_RX_PIN PIN_C7
//PARA RS485
#define MODBUS_SERIAL_ENABLE_PIN     PIN_E0
//#define MODBUS_SERIAL_RX_ENABLE    PIN_A1
#else
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
#endif
#include "MI_MODBUS_485_PRC.c"
#define MODBUS_ADDRESS 0x0B
#define EEPROM_ADDRESS long int
#bit RE1=0x73.1
```

```
//// Subrutinas de lectura SPI sensores
////////////////////////////////////////////////////////////////////////

void read_SPI_RTD()
{
  output_low(PIN_B1);
    delay_ms(5);
   datoRTD1 = spi_xfer();
      delay_ms(5);
  datoRTD2 = spi_xfer();
   delay_ms(5);
   datoRTD3 = spi_xfer();
   delay_ms(5);
  if (datoRTD1==255)
           {
              datoRTD1=datoRTD4;
             }
  if (datoRTD2==255)
           {
              datoRTD2=datoRTD5;
             }
  if (datoRTD3==255)
           {
              datoRTD3=datoRTD6;
             }
 datoRTD4=datoRTD1;
 datoRTD5=datoRTD2;
 datoRTD6=datoRTD3;
   output_high(PIN_B1);
}

void read_SPI_LC()
{
  output_low(PIN_B2);
    delay_ms(5);
   datoLC1 = spi_xfer();
      delay_ms(5);
  datoLC2 = spi_xfer();
   delay_ms(5);
   datoLC3 = spi_xfer();
   delay_ms(5);
   output_high(PIN_B2);

}

//// Subrutinas de escritura SPI
////////////////////////////////////////////////////////////////////////

void write_SPI0()
{
      output_low(PIN_D1);
      delay_ms(10);
      spi_xfer(DA_spi,anout1);
      spi_xfer(DA_spi,anout2);
      delay_ms(10);
      output_high(PIN_D1);
}


void write_SPI1()
{
      output_low(PIN_D0);
      delay_ms(10);
      spi_xfer(DA_spi,anout1);
      spi_xfer(DA_spi,anout2);
      delay_ms(10);
      output_high(PIN_D0);
}
```

```
//// Subrutinas de lectura SPI
////////////////////////////////////////////////////////////////////////

void read_SPI0()
{
      output_low(PIN_B4);
      delay_ms(20);
      spi_xfer(DA_spi,1);
        dato1 = spi_xfer(DA_spi,0x80);
      dato2 = spi_xfer(DA_spi,0);
      dato3 = dato1&15;
      delay_ms(20);
      output_high(PIN_B4);
}

void read_SPI1()
{
      output_low(PIN_B4);
      delay_ms(20);
      spi_xfer(DA_spi,1);
      dato5 = spi_xfer(DA_spi,0xC0);
      dato6 = spi_xfer(DA_spi);
      dato7 = dato5&15;
      delay_ms(20);
      output_high(PIN_B4);
}

//// Subrutinas de escritura I2C
////////////////////////////////////////////////////////////////////////

void write_ext_eeprom(long int address, BYTE data)
{
   short int status;
   i2c_start();                     //Inicializa la transmisión
   i2c_write(248);
   delay_ms(10);
   i2c_write(modbus_rx.data[3]);    //Dato a escribir
   delay_ms(10);
   i2c_stop();                      //Finalización de la transmisión.
   i2c_start();                     //Reinicio
   status=i2c_write(248);           //Lectura del bit ACK, para evitar escrituras incorrectas
  while(status==1)                  //Si es 1 esperar a que responda el esclavo
   {
      i2c_start();
      status=i2c_write(248);
   }
}

//// Subrutinas de lectura I2C
////////////////////////////////////////////////////////////////////////

BYTE read_ext_eeprom(long int address) {

   i2c_start();                     //Inicializa la transmisión
   i2c_write(144);                  //Escribe la palabra de control (dirección 0h + 0 para
escritura)
    //delay_ms(10);
   i2c_start();                     //Reinicio
   i2c_write(145);                  //Escribe la palabra de control (dirección 0h + 1 para lectura)
   data=i2c_read(0);                //lectura del dato
   i2c_stop();                      //Finalización de la transmisión.
   //delay_ms(10);
   return(data);
}
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);
}
```

```
//// Programa Principal Modbus
////////////////////////////////////////////////////////////////////////

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;
    set_tris_d(0b00000000);
    set_tris_a(0b11111111);
    set_tris_e(0b00000000);
    set_tris_b(0b00000001);
    set_tris_c(0b01000001);
   output_d (0);
   output_high(PIN_E1);
   output_high(PIN_B1);
   output_high(PIN_B2);
   output_high(PIN_B3);
   output_high(PIN_B4);
   output_high(PIN_B5);
   output_high(PIN_C3);
   output_high(PIN_C4);
   set_timer0(0);
   set_timer1(0);
   delay_ms(50);
   modbus_init();
   while(TRUE)
   {
   int8 valor=0, dato1_I2C, dato2_I2C, dato;
   EEPROM_ADDRESS address;

 //// Para leer los input registers
////////////////////////////////////////////////////////////////////////


     if (modbus_rx.func==4)
           {
               output_high(PIN_C3);
               output_high(PIN_C4);
               delay_ms (5);
                 dato=READ_EXT_EEPROM( address);
             delay_ms (5);
              output_high(PIN_C3);
               output_high(PIN_C4);
              read_SPI_RTD();
              delay_ms (5);
               read_SPI0();
              delay_ms (5);
              read_SPI1();
              delay_ms (5);
              read_SPI_LC();
              delay_ms (10);

             }
             output_high(PIN_C3);
               output_high(PIN_C4);
               delay_ms (5);

   output_low (D7);
   output_high (D4);
   delay_ms(50);

   output_low (D4);
   output_high (D5);
   delay_ms(50);

   output_low (D5);
```

```
   output_high (D6);
   delay_ms(50);

   output_low (D6);
   output_high (D7);
   delay_ms(50);

     in_q=input_a()&15;
     in_q1=in_q^15;
     //CONTADOR0=GET_TIMER0();
     //CONTADOR1=GET_TIMER1();

//// Registros Modbus
////////////////////////////////////////////////////////////////////////

  input_regs[0]=make16(in_q1,00);
   input_regs[1]=make16(dato,00);
   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(CONTADOR0,00);
   input_regs[7]=CONTADOR1;
   input_regs[8]=make16(00,00);
   input_regs[9]=make16(00,00);

   hold_regs[0]=make16(an_out02,an_out01);
   hold_regs[1]=make16(an_out12,an_out11);
   hold_regs[2]=make16(an_out22,an_out21);
   hold_regs[3]=make16(an_out32,an_out31);
   hold_regs[4]=make16(dato1_I2C,00);
   hold_regs[5]=make16(00,00);
   hold_regs[6]=make16(YYYY,00);
   hold_regs[7]=make16(servo,00);
   hold_regs[8]=make16(XXXX,00);
   hold_regs[9]=make16(out_q,00);

//// Funciones Modbus
////////////////////////////////////////////////////////////////////////

      while(!modbus_kbhit());

      check address against our address, 0 is broadcast
      if((modbus_rx.address == MODBUS_ADDRESS) || modbus_rx.address == 0)
      {
         switch(modbus_rx.func)
         {
            case FUNC_READ_COILS:    //read coils
            case FUNC_READ_DISCRETE_INPUT:    //read inputs
               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
                {
                   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]);
                  set_timer1(0);
                  delay_ms(100);
               event_count++;
            }
            break;
         case FUNC_WRITE_SINGLE_COIL:      //write 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
            {
               //coils are stored msb->lsb so we must use 7-address
               if(modbus_rx.data[2] == 0xFF)
                  bit_set(coils,7-modbus_rx.data[1]);
               else
                  bit_clear(coils,7-modbus_rx.data[1]);


modbus_write_single_coil_rsp(MODBUS_ADDRESS,modbus_rx.data[1],((int16)(modbus_rx.data[2]))<<8);

               event_count++;
            }
            break;
         case FUNC_WRITE_SINGLE_REGISTER:
            if(modbus_rx.data[0] || modbus_rx.data[1] >= 10)
               modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_DATA_ADDRESS);
            else
            {
               //the registers are stored in little endian format
               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[0],modbus_rx.data[1]),
                           make16(modbus_rx.data[2],modbus_rx.data[3]));

                           if(modbus_rx.data[1]== 3)
                            {
                            an_out31 = modbus_rx.data[2];
                            an_out32 = modbus_rx.data[3];
                            anout1 = modbus_rx.data[2]|144;
                            anout2 = modbus_rx.data[3];
                             write_SPI0();
                           output_high(PIN_D1);
                           delay_ms(5);
                             }

                            else
                           {
                            if(modbus_rx.data[1]== 9)
                           {
                           out_q = modbus_rx.data[3];
                           output_d (out_q&255);
```

```
  //WRITE_EXT_EEPROM(248, modbus_rx.data[3]);
  delay_ms(5);
  }
  else
  {
  if(modbus_rx.data[1]== 0)
 {
 an_out01 = modbus_rx.data[2];
 an_out02 = modbus_rx.data[3];
 anout1 = modbus_rx.data[2]|16;
 anout2 = modbus_rx.data[3];

  write_SPI1();
  output_high(PIN_D0);
delay_ms(5);
  }
  else
  {
  if(modbus_rx.data[1]== 1)
 {
 an_out11 = modbus_rx.data[2];
 an_out12 = modbus_rx.data[3];
 anout1 = modbus_rx.data[2]|144;
 anout2 = modbus_rx.data[3];
  //write_SPI2();
  output_high(PIN_D0);
delay_ms(5);
  }
  else
  {
  if(modbus_rx.data[1]== 2)
 {
 an_out21 = modbus_rx.data[2];
 an_out22 = modbus_rx.data[3];
 anout1 = modbus_rx.data[2]|16;
 anout2 = modbus_rx.data[3];
  //write_SPI3();
  output_high(PIN_D1);
delay_ms(5);
  }
  else
  {

  if(modbus_rx.data[1]== 7)
 {
 servo = modbus_rx.data[3];

  WRITE_EXT_EEPROM(248, modbus_rx.data[3]);

  delay_ms(5);
  }
  else
   {
    if(modbus_rx.data[1]== 8)
 {

   XXXX = set_pwm1_duty (modbus_rx.data[3]);
  if (XXXX  == 0)
  {

     output_low(PIN_E2);
  }
  else
  {
     output_high(PIN_E2);
  }
  delay_ms(5);
  }
  else
```

```
                                {
                                    if(modbus_rx.data[1]== 6)
                                {
                                YYYY = set_pwm2_duty (modbus_rx.data[3]);
                                 delay_ms(5);
                                 }
                                 }
                                 }
                                 }
                                 }
                                 }
                                 }
                                 }
                               //output_d (modbus_rx.data[3]);//
                               //WRITE_EXT_EEPROM(248, modbus_rx.data[3]);

//_____________________________________________________________________________//
                               //HASTA AQUI TODO BIEN//
               }
               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);
                     else
                        bit_clear(coils,7-i);
                  }

                  modbus_write_multiple_coils_rsp(MODBUS_ADDRESS,
                                 make16(modbus_rx.data[0],modbus_rx.data[1]),
                                 make16(modbus_rx.data[2],modbus_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:    //We don't support the function, so return exception
               modbus_exception_rsp(MODBUS_ADDRESS,modbus_rx.func,ILLEGAL_FUNCTION);
         }
      }
   }
```

Programa del microcontrolar PIC18F2550 Master Modbus de la tarjeta MCM.

```
////////////////////////////////////////////////////////////////////////
////                          VISA_MOSBUS.c                         ////
////                                                                ////
////                                                                ////
//// Realizado con el compilador CCS PCWH 4.014                     ////
////                                                                ////
//// Javier Quinga Jarrín, Angel Manosalvas Benalcázar              ////
////////////////////////////////////////////////////////////////////////

//ºººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººº
//ºººººººººººCONFIGURACION DEL MICROCONTROLADORººººººººººººººººººººº
#include <18F2550.h>
#device adc=8
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL1,CPUDIV1,VREGEN
#use delay(clock=48000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
//ºººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººº

//ºººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººººº
//ºººººººººººDEFINICION DEL DISPOSITIVOººººººººººººººººººººººººººººº
#define MODBUS_TYPE MODBUS_TYPE_MASTER
#define MODBUS_SERIAL_RX_BUFFER_SIZE 64
#define MODBUS_SERIAL_BAUD 9600

#define USE_WITH_PC 0

#ifndef USE_WITH_PC
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, stream=PC, errors)
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_EXT
#define MODBUS_SERIAL_TX_PIN PIN_C6   // Data transmit pin
#define MODBUS_SERIAL_RX_PIN PIN_C7   // Data receive pin
//The following should be defined for RS485 communication
//#define MODBUS_SERIAL_ENABLE_PIN   0   // Controls DE pin for RS485
//#define MODBUS_SERIAL_RX_ENABLE    0   // Controls RE pin for RS485
#define DEBUG_MSG(msg) fprintf(PC, msg)
#define DEBUG_DATA(msg,data) fprintf(PC, msg, data)
#else
#define MODBUS_SERIAL_INT_SOURCE MODBUS_INT_RDA
#define DEBUG_MSG(msg) if(0)
#define DEBUG_DATA(msg,data) if(0)
#endif

#include "MI_MODBUS.c"

#define MODBUS_SLAVE_ADDRESS 0x0A

#define LED_ON     PIN_C1
#define LED_COM    PIN_C2

//// Configuración USB
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
#define USB_HID_DEVICE     FALSE             //deshabilitamos el uso de las directivas HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for IN bulk/interrupt
transfers
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for OUT bulk/interrupt
transfers
#define USB_EP1_TX_SIZE    15                 //size to allocate for the tx endpoint 1 buffer
#define USB_EP1_RX_SIZE    15                 //size to allocate for the rx endpoint 1 buffer

#include <pic18_usb.h>
#include <descriptor.h>
#include <usb_MDB.c>

#define LEDV    PIN_B6
#define LEDR    PIN_B7
```

```
//#define LEDA    PIN_B0
#define ENCENDER  output_high
#define APAGAR    output_low

#define control   bufferI[0]
#define leds      bufferI[1]
#define pwm_o     bufferI[2]
#define medida1    bufferO[0]
#define medida2    bufferO[1]
#define medida3    bufferO[2]
#define term1    bufferO[3]
#define term2    bufferO[4]

int16 out_regs[] = {0x1100,0x2200,0x3300,0x4400,0x5500,0x6600,0x7700,0x8800};


//// Funciones Modbus
////////////////////////////////////////////////////////////////////////


int i;

/*This function may come in handy for you since MODBUS uses MSB first.*/
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 print_menu()
{
   DEBUG_MSG("\r\nPick command to send\r\n1. Read all coils.\r\n");
   DEBUG_MSG("2. Read all inputs.\r\n3. Read all holding registers.\r\n");
   DEBUG_MSG("4. Read all input registers.\r\n5. Turn coil 6 on.\r\n6.");
   DEBUG_MSG("Write 0x4444 to register 0x03\r\n7. Set 8 coils using 0x50 as mask\r\n");
   DEBUG_MSG("8. Set 2 registers to 0x1111, 0x2222\r\n9. Send unknown command\r\n");
}

void read_all_coils()
{
   DEBUG_MSG("Coils:\r\n");
   if(!(modbus_read_coils(MODBUS_SLAVE_ADDRESS,0,8)))
   {
      DEBUG_MSG("Data: ");
      /*Started at 1 since 0 is quantity of coils*/
      for(i=1; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void read_all_inputs()
{
   DEBUG_MSG("Inputs:\r\n");
   if(!(modbus_read_discrete_input(MODBUS_SLAVE_ADDRESS,0,8)))
   {
      DEBUG_MSG("Data: ");
      /*Started at 1 since 0 is quantity of coils*/
      for(i=1; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
```

```
   }
}

void read_all_holding()
{
   DEBUG_MSG("Holding Registers:\r\n");
   if(!(modbus_read_holding_registers(MODBUS_SLAVE_ADDRESS,0,10)))
   {
      DEBUG_MSG("Data: ");
      /*Started at 1 since 0 is quantity of coils*/
      for(i=1; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void read_all_input_reg()
{
   //DEBUG_MSG("Input Registers:\r\n");
   if(!(modbus_read_input_registers(MODBUS_SLAVE_ADDRESS,0,10)))
   {
      //DEBUG_MSG("Data: ");
      /*Started at 1 since 0 is quantity of coils*/
      for(i=1; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
     // DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void write_coil()
{
   DEBUG_MSG("Writing Single Coil:\r\n");
   if(!(modbus_write_single_coil(MODBUS_SLAVE_ADDRESS,6,TRUE)))
   {
      DEBUG_MSG("Data: ");
      for(i=0; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void write_reg()
{
  // DEBUG_MSG("Writing Single Register:\r\n");

   if(!(modbus_write_single_register(MODBUS_SLAVE_ADDRESS,9,out_regs[0])))

   {
      //DEBUG_MSG("Data: ");
      for(i=0; i < (modbus_rx.len); ++i)
        // DEBUG_DATA("%X ", modbus_rx.data[i]);
         DEBUG_DATA("%X ", 255);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
```

```
   }
}

void write_coils()
{
   int8 coils[1] = { 0x50 };
   DEBUG_MSG("Writing Multiple Coils:\r\n");
   if(!(modbus_write_multiple_coils(MODBUS_SLAVE_ADDRESS,0,8,coils)))
   {
      DEBUG_MSG("Data: ");
      for(i=0; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void write_regs()
{
   int16 reg_array[2] = {0x1111, 0x2222};
   DEBUG_MSG("Writing Multiple Registers:\r\n");
   if(!(modbus_write_multiple_registers(MODBUS_SLAVE_ADDRESS,0,2,reg_array)))
   {
      DEBUG_MSG("Data: ");
      for(i=0; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void unknown_func()
{
   DEBUG_MSG("Trying unknown function\r\n");
   DEBUG_MSG("Diagnostic:\r\n");
   if(!(modbus_diagnostics(MODBUS_SLAVE_ADDRESS,0,0)))
   {
      DEBUG_MSG("Data:");
      for(i=0; i < (modbus_rx.len); ++i)
         DEBUG_DATA("%X ", modbus_rx.data[i]);
      DEBUG_MSG("\r\n\r\n");
   }
   else
   {
      DEBUG_DATA("<-**Exception %X**->\r\n\r\n", modbus_rx.error);
   }
}

void parse_read(char c)
{
   switch(c)
   {
      case '1':
         read_all_coils();
         break;
      case '2':
         read_all_inputs();
         break;
      case '3':
         read_all_holding();
         break;
      case '4':
         read_all_input_reg();
```

```
         break;
   }
}

void parse_write(char c)
{
   switch(c)
   {
      case '5':
         write_coil();
         break;
      case '6':
         write_reg();
         break;
      case '7':
         write_coils();
         break;
      case '8':
         write_regs();
         break;
      case '9':
         unknown_func();
         break;
   }
}


//// Programa Principal USB - Modbus
////////////////////////////////////////////////////////////////////////

void main(void) {

char c;


   //declaración de variables
   int8 iBuff[15];                  //buffers
   int8 oBuff[15];


   //enable_interrupts(global);
   //enable_interrupts(int_rda);

  BYTE sec;
  BYTE min;
  BYTE hrs;
  BYTE day;
  BYTE month;
  BYTE yr;
  BYTE dow;
  BYTE FLAG;

//   lcd_init();
//   lcd_putc("\f    ...Javier...\n");
//   delay_ms( 500 );

//  ds1307_init();


   usb_init();

   usb_task();
   usb_wait_for_enumeration();


 delay_ms(100);

//   DEBUG_MSG("\r\nInitializing...");
   modbus_init();
```

```
//   DEBUG_MSG("...ready\r\n");

#ifndef USE_WITH_PC
   do{
      print_menu();
      c = getc(PC);

//      fprintf(PC,"\r\n");
      parse_read(c);
      parse_write(c);  //Split between two functions due to segment size issues

   } while(TRUE);
#else
   //read_all_coils();
  // read_all_inputs();
   //read_all_holding();
   //read_all_input_reg();
   //write_coils();
   //write_regs();
   //write_coil();
   write_reg();
   //read_all_coils();
   //read_all_holding();
   unknown_func();
#endif


    while (TRUE)
   {

  // ds1307_get_date(day,month,yr,dow);
  //   delay_ms(50);
  // lcd_gotoxy(1,2);
  // printf(lcd_putc,"\fFecha: %2d:%2d:%2d",day,month,dow);

  //  ds1307_get_time(hrs,min,sec);

  // lcd_gotoxy(1,1);
   //printf(lcd_putc,"\f  Fecha:%2d/0%1d/20%2d",day,month,yr);
   //delay_ms(5);
   //lcd_gotoxy(1,2);
   //printf(lcd_putc,"\n  Hora:%2d:%2d:%2d",hrs,min,sec);
   //delay_ms(5);

   //oBuff[0]=hrs;
   //oBuff[1]=min;
   //oBuff[2]=sec;


//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°asi se recibe datos via modbus slave a modbus master°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°

// read_all_input_reg();
//printf(lcd_putc,"\f%4d%4d%4d%4d",modbus_rx.data[0],modbus_rx.data[2],modbus_rx.data[4],modbus_rx.d
ata[6]);
//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°asi se recibe datos via modbus slave a modbus master°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°

//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°asi se envia datos via modbus master a modbus slave°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
  // delay_ms(1000);
   //oBuff[3]=modbus_rx.data[4];
   //out_regs[0]=make16(00,sec);
     //      write_reg();
```

```
//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°asi se envia datos via modbus master a modbus slave°°°°°°°°°°°°°°°°°°°
//°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°


      if(usb_enumerated())          // PicWinUSB is enumerated?
      {
         if (usb_kbhit(1))          // EP has data?
         {
            usb_get_packet(1, iBuff, 3); // Read 2 byte packet from EP 1


           //printf(lcd_putc,"\n  Dato:%2d:%2d",iBuff[0],iBuff[1]);

            delay_ms(50);
            if (iBuff[0] == '0') // Led_Mode
            {
              //lcd_putc("\f   ...CONTROL...\n");
               // delay_ms(50);
                if (iBuff[1] == 'A')
                {
                //output_high (LED_ON);
                read_all_input_reg();
          //
printf(lcd_putc,"\f%4d%4d%4d%4d",modbus_rx.data[0],modbus_rx.data[2],modbus_rx.data[4],modbus_rx.dat
a[6]);
                oBuff[0]= modbus_rx.data[1];
                 oBuff[1]= modbus_rx.data[2];
                  oBuff[2]= modbus_rx.data[3];
                   oBuff[3]= modbus_rx.data[4];
                    oBuff[4]= modbus_rx.data[5];
                    oBuff[5]= modbus_rx.data[6];
                     oBuff[6]= modbus_rx.data[7];
                    oBuff[7]= modbus_rx.data[8];
                     oBuff[8]= modbus_rx.data[9];
                    oBuff[9]= modbus_rx.data[10];

                }
                 if (iBuff[1] == 'B')
                {
                //output_low (LED_ON);

                  out_regs[0]=make16(00,iBuff[2]);
                  write_reg();
                }


            }

            if (iBuff[0] == '1') // ADC_Mode
            {
               //lcd_putc("\f  ...LECTURA...\n");
                delay_ms(50);
               usb_put_packet(1, oBuff, 10, USB_DTS_TOGGLE); // Send 2 byte packet to EP 1
            }



         }
      }
      }

}
```

Programa del microcontrolar PIC16F876A esclavo I2C de entradas discretas

```
////////////////////////////////////////////////////////////////////////
////                          MLE.c                                 ////
////                                                                ////
////                                                                ////
//// Realizado con el compilador CCS PCWH 4.014                     ////
////                                                                ////
//// Javier Quinga Jarrín, Angel Manosalvas Benalcázar              ////
////////////////////////////////////////////////////////////////////////


#include <16F876A.h>
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz)
#FUSES PUT                      //Power Up Timer
#FUSES PROTECT                  //Code protected from reads
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES BROWNOUT                 //Brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected

#use delay(clock=4M)

#use i2c(SLAVE, sda=PIN_C4, scl=PIN_C3, address=0x60, force_hw, stream=I2CS)

#use fast_io(D)

int rcv_buf[0x10];
int wrt_buf[0x10];
int cmd=0xFF;

#int_SSP // interrupt on i2c activity
void  i2c_isr(void)
{
    int state, incoming;
    state = i2c_isr_state();

    if(state == 120)
    {
        incoming = i2c_read(I2CS);
        if (state == 1)
        {
            cmd = incoming;
        }

      }
    else
    {
        i2c_write(I2CS,wrt_buf[]);
    }
}

void main()
{

    enable_interrupts(INT_SSP);
    enable_interrupts(GLOBAL);

    printf("\n\rbegin");

    while (TRUE)
    {
   wrt_buf = input_b;
    }
}
```

Programa del microcontrolar PIC16F876A esclavo I2C de salidas discretas

```
////////////////////////////////////////////////////////////////////////
////                           MLS.c                                ////
////                                                                ////
////                                                                ////
//// Realizado con el compilador CCS PCWH 4.014                     ////
////                                                                ////
//// Javier Quinga Jarrín, Angel Manosalvas Benalcázar              ////
////////////////////////////////////////////////////////////////////////


#include <16F876A.h>
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz)
#FUSES PUT                      //Power Up Timer
#FUSES PROTECT                  //Code protected from reads
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES BROWNOUT                 //Brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected

#use delay(clock=4M)

#use i2c(SLAVE, sda=PIN_C4, scl=PIN_C3, address=0x60, force_hw, stream=I2CS)

#use fast_io(D)

int rcv_buf[0x10];
int wrt_buf[0x10];
int cmd=0xFF;

#int_SSP // interrupt on i2c activity
void  i2c_isr(void)
{
    int state, incoming;
    state = i2c_isr_state();

    if(state == 248)
    {
        incoming = i2c_read(I2CS);
        if (state == 1)
        {
            cmd = incoming;
        }

      }
    else
    {
        i2c_write(I2CS,wrt_buf[]);
    }
}

void main()
{

    enable_interrupts(INT_SSP);
    enable_interrupts(GLOBAL);

    printf("\n\rbegin");

    while (TRUE)
    {
   output_b = cmd;
    }
}
```

Programa del microcontrolar PIC16F819 esclavo I2C de salida PWM

```
////////////////////////////////////////////////////////////////////////
////                          servo.c                               ////
////                                                                ////
////                                                                ////
//// Realizado con el compilador CCS PCWH 4.014                     ////
////                                                                ////
//// Javier Quinga Jarrín, Angel Manosalvas Benalcázar              ////
////////////////////////////////////////////////////////////////////////
#include <16F819.h>

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz)
#FUSES PUT                      //Power Up Timer
#FUSES PROTECT                  //Code protected from reads
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES BROWNOUT                 //Brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected

#use delay(clock=4M)

#use i2c(SLAVE, sda=PIN_C4, scl=PIN_C3, address=0x60, force_hw, stream=I2CS)

#use fast_io(D)
SETUP_TIMER_2(T2_DIV_BY_16,1,1);
SETUP_CCP1(CCP_PWM);
int rcv_buf[0x10];
int wrt_buf[0x10];
int cmd=0xFF;

#int_SSP // interrupt on i2c activity
void  i2c_isr(void)
{
    int state, incoming;
    state = i2c_isr_state();

    if(state == 248)
    {
        incoming = i2c_read(I2CS);
        if (state == 1)
        {
            cmd = incoming;
        }

      }
    else
    {
        i2c_write(I2CS,wrt_buf[]);
    }
}

void main()
{

    enable_interrupts(INT_SSP);
    enable_interrupts(GLOBAL);

    printf("\n\rbegin");

    while (TRUE)
    {
set_pwm1_duty (cmd);
    }
}
```

ANEXO V

COTIZACIONES

PROFORMA No.: 2048

Av.Colón Oe3-31 y Versalles
Quito - Ecuador
(593) 2 2502-124

Fecha de Emisión : **23-JUN-11 15:46:18**
Cliente : **JAVIER QUINGA**
Dirección : LAS CASAS
Teléfono : 098966786
Atención : **JAVIER QUINGA**

Bodega: Principal
Vendedor: CHAVEZDE
Fecha: 23/Jun/2011

Código	Descripción	Cantidad	P.V.P.	Total
001088	CC 100NF- SMD	1,00	0,2800	0,2800
000593	CC 22 PF/50V	1,00	0,0714	0,0714
000801	CRY 20 MHZ - SMD	1,00	0,5804	0,5804
001339	PIC16F877A SMD	1,00	5,8036	5,8036
002345	PIC16F887 - SMD	1,00	7,0089	7,0089
001154	PIC16F628A SMD	1,00	3,1250	3,1250
002780	PIC16F870 - SMD	1,00	5,4464	5,4464
001634	PIC18F2550 SMD	1,00	8,7054	8,7054
001557	ATMEGA 324 - SMD	1,00	13,0804	13,0804
001559	ATMEGA 16 SMD	1,00	5,0446	5,0446
002327	ATMEGA 32 - SMD	1,00	7,1429	7,1429
001142	ATMEGA 128 SMD	1,00	16,6071	16,6071
001470	LED RO - SMD	1,00	0,7143	0,7143
002453	LED RGB SMD	1,00	0,5357	0,5357
001635	1N4148 SMD	1,00	0,0893	0,0893
000427	R 1 KOHM SMD	1,00	0,0300	0,0300

Valides de la Proforma : **8 días**
Forma de Pago : **Contado**
Observaciones :

Subtotal:	**74,27**
Descuento realizado:	**,00**
IVA 12 %	**8,91**
Valor Total de la Proforma :	**83,18**

Atentamente
A.P.M.

Recibí Conforme

PROFORMA No.: 2049

Av.Colón Oe3-31 y Versalles
Quito - Ecuador
(593) 2 2502-124

Fecha de Emisión : **23-JUN-11 15:49:49**
Cliente : **JAVIER QUINGA**
Dirección : LAS CASAS
Teléfono : 098966786
Atención : **JAVIER QUINGA**

Bodega: Principal
Vendedor: CHAVEZDE
Fecha: 23/Jun/2011

Código	Descripción	Cantidad	P.V.P.	Total
001340	2N3904 - SMD	1,00	0,0900	0,0900
000787	MOSFET FDN335N - SMD	1,00	0,6250	0,6250
000139	ZENER 5.1 V	1,00	0,0982	0,0982
002355	DS1307 - SMD	1,00	4,1700	4,1700
001243	MAX 232 SMD	1,00	2,0536	2,0536
000861	SN75176 - SMD	1,00	1,3393	1,3393

Valides de la Proforma : **8 días**
Forma de Pago : **Contado**
Observaciones :

Subtotal:	**8,38**
Descuento realizado:	**,00**
IVA 12 %	**1,01**
Valor Total de la Proforma :	**9,38**

Atentamente
A.P.M.

Recibí Conforme

Order Shipment Notification

**** Your order has been shipped ****

If you have any questions about this order, please contact our sales office at 866-433-5722.

Order Information

Web Order Number:	**201110171643**
Allied Order Number:	76875Y-00
Your PO Number:	JAVIER
Ordered By:	JAVIER QUINGA
Telephone Number:	098966786
Ship Method:	UPS

Items Ordered

Qty. Ordered	Qty. Shipped	Stock #	Mfr. Part #	Your Part #	Price	Total
4	4	383-2105	MCP4822-E/SN		$3.41	$13.64
DUAL 12-BIT DAC WITH SPI INTERACE AND INTERNAL VREF, SOIC-8						
10	10	630-0489	AD822ARZ		$6.73	$67.27
IC OPAMP FET R-R LP 8-SOIC						
5	5	383-0214	MCP3202-BI/SN		$3.28	$16.42
12-BIT ADC, SPI, DUAL CHANNEL						
1	1	383-1743	PIC18F2550-I/SO		$4.69	$4.69
28 PIN, 32 KB FLASH, 2048 RAM, FS-USB 2.0						
2	2	383-0441	PIC16F819-I/SO		$2.42	$4.84
18 PIN, 3.5 KB FLASH, 256 RAM, 16 I/O						
10	10	735-2166	ULN2003ADR		$0.34	$3.39
DARLINGTON TRANSISTOR ARRAY, 16 PIN SOIC						
4	4	383-1108	MCP3551-E/SN		$4.52	$18.08
A/D Converter, 22-Bit Delta-Sigma, 1-Ch, 15 SPS, SOIC-8						

10	10	383-1081	MCP1702T-3302E/CB	$0.48	$4.80
LOW IQ 250MA LDO, VIN 10V MAX, VOUT=3.3V					
3	3	630-0233	AD620ARZ	$8.54	$25.62
Amplifier; IC Instrument Amp LP 8-SOIC					
6	6	735-2167	ULN2803ADW	$0.72	$4.33
DARLINGTON TRANSISTOR ARRAY, 18 PIN SOIC					
4	4	248-0413	L293DD	$2.88	$11.54
PUSH-PULL FOUR CHANNEL DRIVER, 20 PIN SOIC					
4	4	383-2473	MCP3421A1T-E/CH	$2.72	$10.88
18-bit delta-sigma ADC, single channel, 4sps					
4	4	383-1869	MCP9800A0T-M/OT	$1.36	$5.45
HIGH-ACCURACY, 12-BIT THERMAL SENSOR WITH SERIAL INTERFACE (A0 ADDRESS)					
2	2	383-0156	DSPIC30F4013-20E/P	$6.92	$13.84
16 BIT MCU/DSP 40LD 20MIPS 48 KB FLASH					
2	2	383-1606	MCP4725A0T-E/CH	$1.09	$2.18
Single, 12-bit NV DAC with I2C interface					
4	4	383-0420	PIC16F690-I/P	$2.17	$8.68
20 PIN, 7 KB FLASH, 256 RAM, 18 I/O					
				Tax:	$15.15
				Freight:	$9.75
				Order Total:	**$240.55**

Tracking Numbers:

Track this order

Soluciones en Manufactura y Electrónica

SMELEKTRONIK CIA. LTDA. **RUC Nº 0190361896001**

Mariscal Lamar 4-19 y Vargas Machuca / Celular: 096472455

Tel. 07 2 835922 / Telefax: 07 2 833834 E-mail: bolipeza@cue.satnet.net

E- mail: raul@smelektronik.com.ec / Web: www.smelektronik.com.ec

PRICE LIST FEBRUARY 2011

PRECIOS DE SERVICIOS EN PLACAS DE CIRCUITO IMPRESO

PRECIO DE CENTIMETROS PROCESADOS en formatos para prototipos

FORMATO	Medidas (mm) *en bruto*	Medidas (mm) *en netto max.*	Centimetros cuadrados	Simple lado	Simple lado con SM	Doble lado	Doble lado con SM	FLEXIBLE SS0.3	FLEXIBLE DS0.2
F1	*107 x 145*	*97 x 135 SS* *90 x 135 DS* *97 x 130 DS*	*155,15*	16,55	24,17	18,85	26,44	26,5	29,25
F2	*215 x 145*	*205 x 135 SS* *205 x 130 DS* *200 x 135 DS*	*311,75*	30,72	39,65	35,26	44,22	41,8	45,17
F3	*107 x 290*	*97 x 280 SS* *90 x 280 DS*	*310,3*	30,18	38,12	34,75	42,65	41,5	43,88
F4	*215 x 290*	*205 x 280 SS* *200 x 280 DS*	*620,6*	58,82	70,28	67,95	79,42	73,1	79,92
F5	*177 x 133*	*167 x 125 SS* *167 x 119 DS* *163 x 125 DS*	*235,41*	23,86	32,35	27,32	35,80	33,4	35,45
F6	*233 x 133*	*223 x 125 SS* *223 x 120 DS*	*309,9*	30,42	39,05	34,95	43,58	41,5	44,66
F7	*260 x 200*	*250 x 190 SS* *250 x 185 DS*	*520*	49,56	60,42	57,18	68,04	64,58	71,14
F8	*260 x 133*	*250 x 125 SS* *250 x 120 DS*	*345,8*	33,57	42,26	38,64	47,33	45,05	49,73
F9	*266 x 125*	*256 x 115 SS* *256 x 110 DS*	*332,5*	32,32	40,77	37,19	45,64	44,23	47,80
F10	*233 x 177*	*223 x 169 SS* *223 x 164 DS*	*412,41*	39,87	49,90	45,92	55,94	53,6	59,32
F11	*300 x 200*	*290 x 190 SS* *290 x 185 DS*	*600*	56,59	67,60	65,38	76,40	71,96	79,27
F12	*300 x 135*	*290 x 125 SS* *290 x 120 DS*	*405*	38,8	47,65	44,73	53,60	49,73	55,58
FE13	*310 x 107*	*300 x 97 SS* *300 x 90 DS*	*331,7*	32,05	40,02	36,92	44,88	42,7	46,63
FE14	*310 x 220*	*300 x 210 SS* *300 x 205 DS*	*682*	64,00	75,72	74,00	85,72	80,96	88,75

Descuentos por volumenes en formatos de producción

Equivalente a:	Medidas (mm)	*Cantidad*	*Descuento*
1 Formato F13	*581 x 268*	*1557 cm²*	*5%*
1 Formato F14	*581 x 536*	*3114 cm²*	*10%*
1 Formato F15	*1162 x 536*	*6228 cm²*	*20%*
1 Formato F16	*1162 x 536 x 2*	*12457 cm²*	*25%*
1 Formato F17	*1162 x 536 x 4*	*24913 cm²*	*30%*

Nota: Para cantidades intermedias y mayores por favor solicite nuestros precios especiales y le cotizaremos personalizadamente su trabajo. Todos los precios de ésta lista no incluyen IVA.

VALORES FIJOS DE TOOLING		VALOR EN $ USD
PLOTTER DOBLE SIDE		6
PLOTTER SIMPLE SIDE		3
PLOTTER DOBLE SIDE + SOLDER MASK		12
PLOTTER SIMPLE SIDE + SOLDER MASK		6
PLOTTER SCREEN x LAYER		3
DISEÑO DE MASCARA GRAFICA EN ILUSTRADOR - Valor x arte		12
IMPRESIÓN DE MASCARA GRAFICA + Plastificado FORMATO A4		6
IMPRESIÓN DE MASCARA GRAFICA + Plastificado FORMATO A3		15
ARMADO DE TECLADO DE MEMBRANA - Valor x cm²		0,025
SCREEN SERIGRAFICO - MATERIALES PINTURA/LIMPIEZA - 1-20 paneles		7
PREPARACION DE MALLA		35
FRESADO - ROUTERING PLACAS	LONGITUD (cm)	0,035
Fresado plásticos, metálicos, etc SERVICIO CNC - Valor por minuto		0,25
DISEÑO DEL FRESADO CNC - CADCAM - VALOR POR HORA		10
SCREEN LAMINADO - VALOR POR FORMATO		3,6
DISEÑO DEL PCB EN ALTIUM DESIGNER - VALOR POR HORA		12
RE-RUTEO DEL PCB - CORRECCIONES - VALOR POR HORA		6

ELEMENTOS	P.Unit
Bateria + socalo	4
Bornera 2-3pin azul	0,4
Buje bronce 10mm	0,4
Buje plástico 4mm	0,2
Buje plástico 8mm	0,35
Cable plano 10 pin - 25uni	0,12
Cable plano 10 pin c/cm	0,2
Cond. 0.1uf 0603	0,15
Diodo tipo 1N4007	0,1
Fuente 3A 24-12 resinado	15
Fuente RS 25-12	32,5
Fuente RS 25-12 - 25unid	28
Fuente RS 35-12	35,5
Fuente RS 35-12 - 25unid	30,6
HIDROCAR	120
LED BICOLOR	0,2
Led0805 SMD	0,1
Ponchado pin met. c/pin	0,03
Pulsante membrana	1
Pulsante memb - 25unid	0,5
Pulsante normal	1
Pulsante normal - 25unid	0,5
Rele 12V 5pin	1
Resist0805 SMD	0,1
Resistecia tipo 1/4W	0,03
Riel DIN c/acople plastico	1
Robot + kit USB	125
Robot ARM	80
Tornillos cst >5mm	0,05
Transistor tipo 2N3904	0,12

DEPOSITOS:

C.CTE. Nº 02072004416 PRODUBANCO a nombre de RAUL PESANTEZ CABRERA Y BOLIVAR PESANTEZ CABRERA.

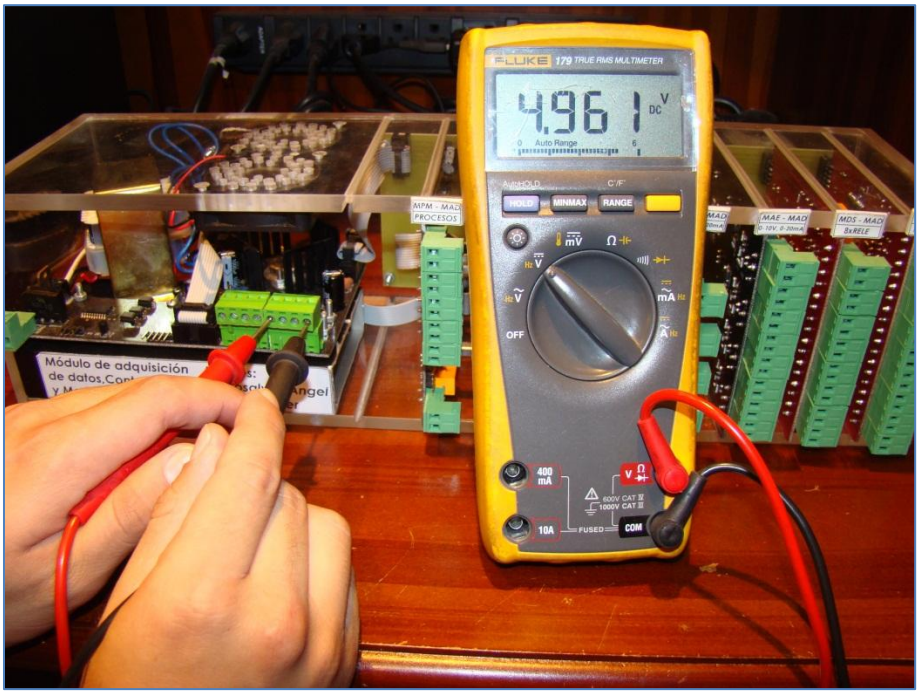
C.CTE. Nº 4015008486 BOLIVARIANO a nombre de RAUL PESANTEZ CABRERA Y BOLIVAR PESANTEZ CABRERA.

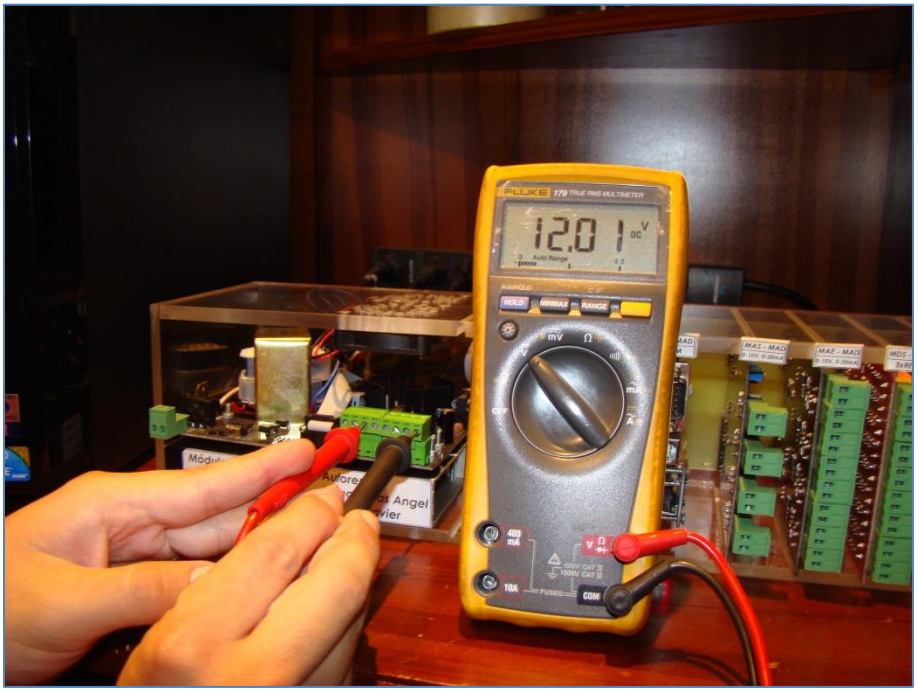
C.CTE. Nº 02072009035 PRODUBANCO a nombre de SMELEKTRONIK CIA. LTDA. * solo en efectivo o transferencia *

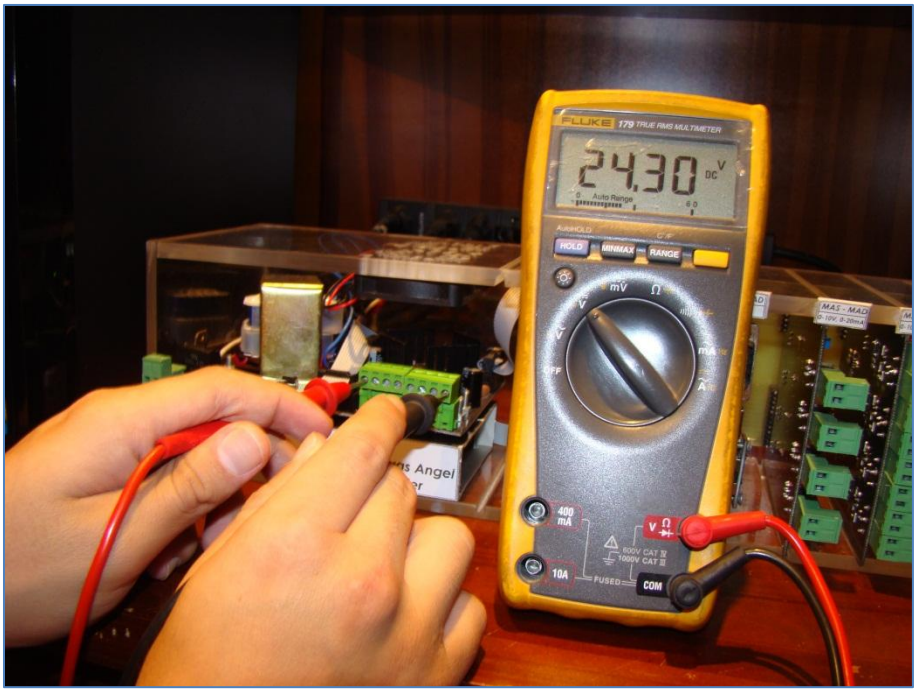
ANEXO VI

FOTOGRAFÍAS DEL PROTOCOLO DE PRUEBAS

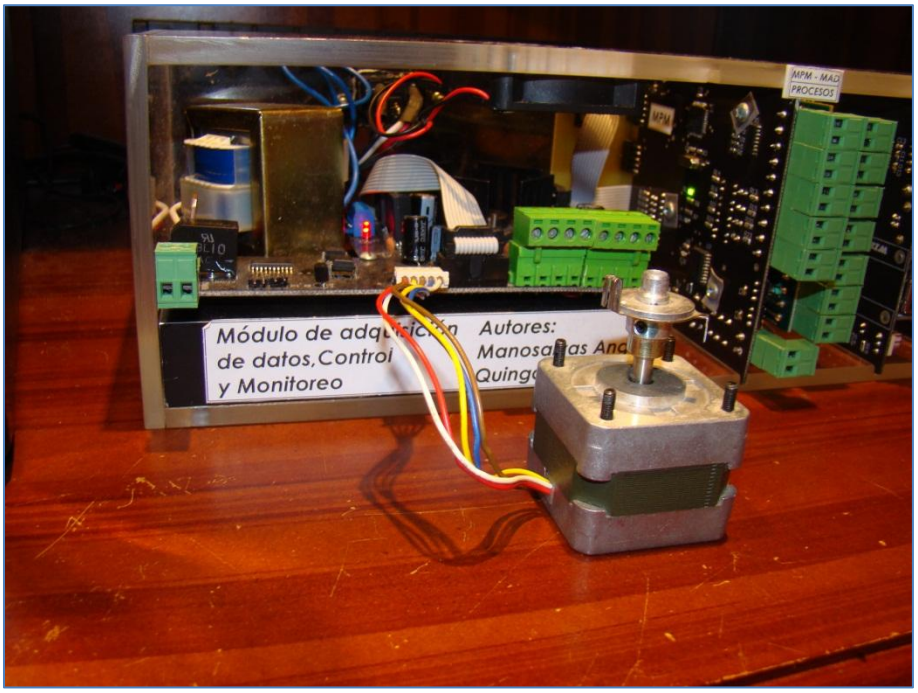
MEDICIÓN DE VALORES DE TENSIÓN EN LA FUENTE DE 5 – 12 Y 24 VDC

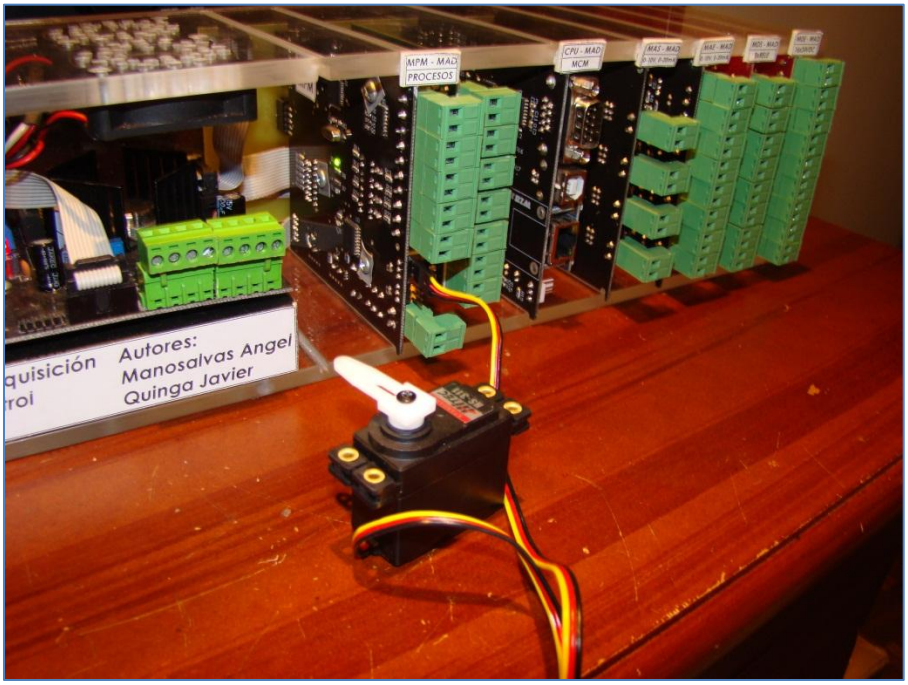


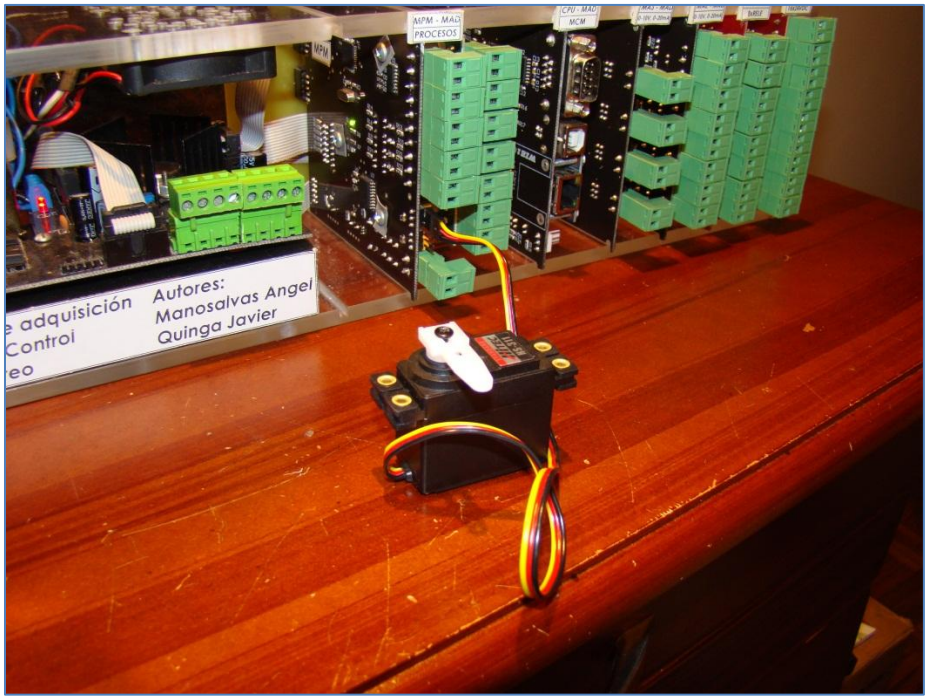


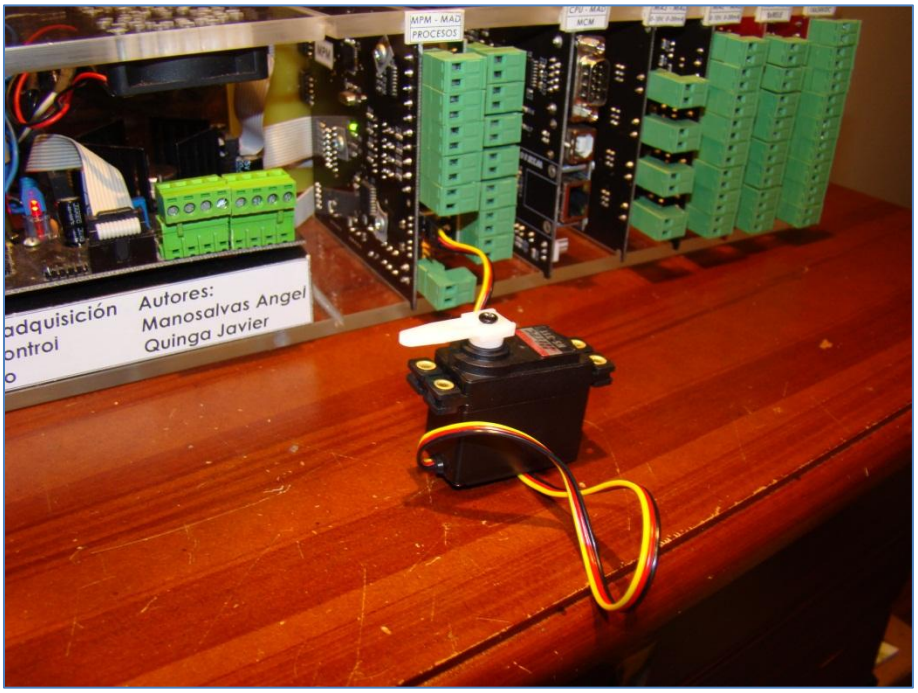

FLUKE 179 TRUE RMS MULTIMETER
24.30 V DC
Auto Range
HOLD
MIN MAX
RANGE
400 mA
10A
COM
FUSED

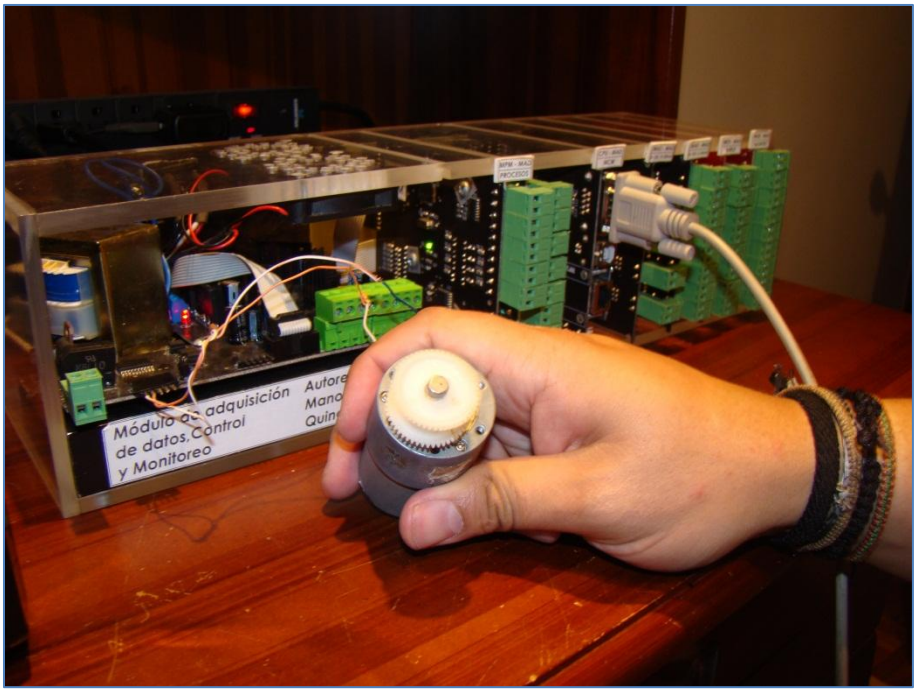
CONEXIÓN DE MOTORES DE TIPO PAP, SERVO MOTOR Y DC



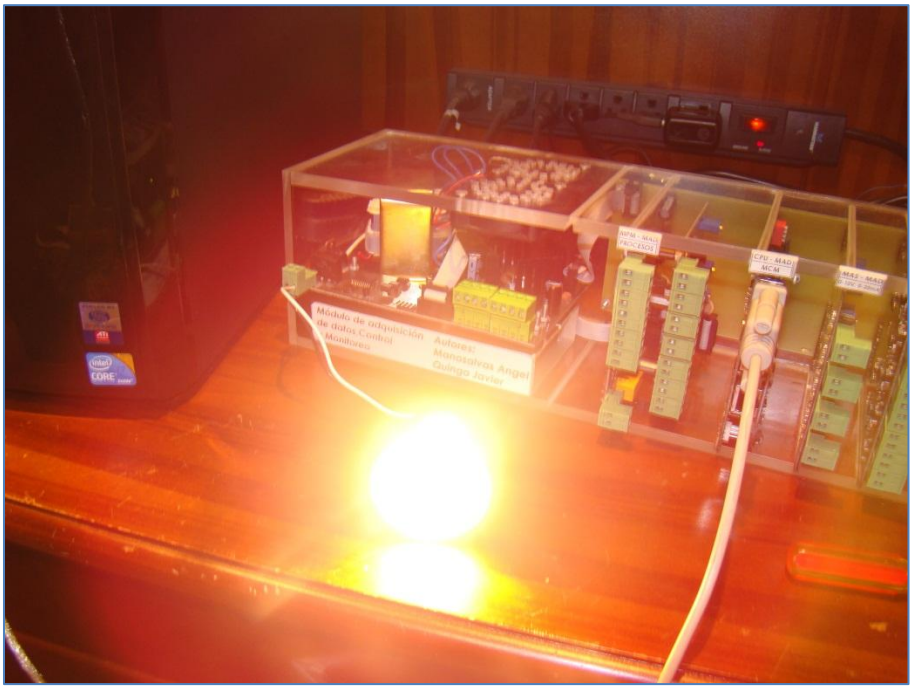


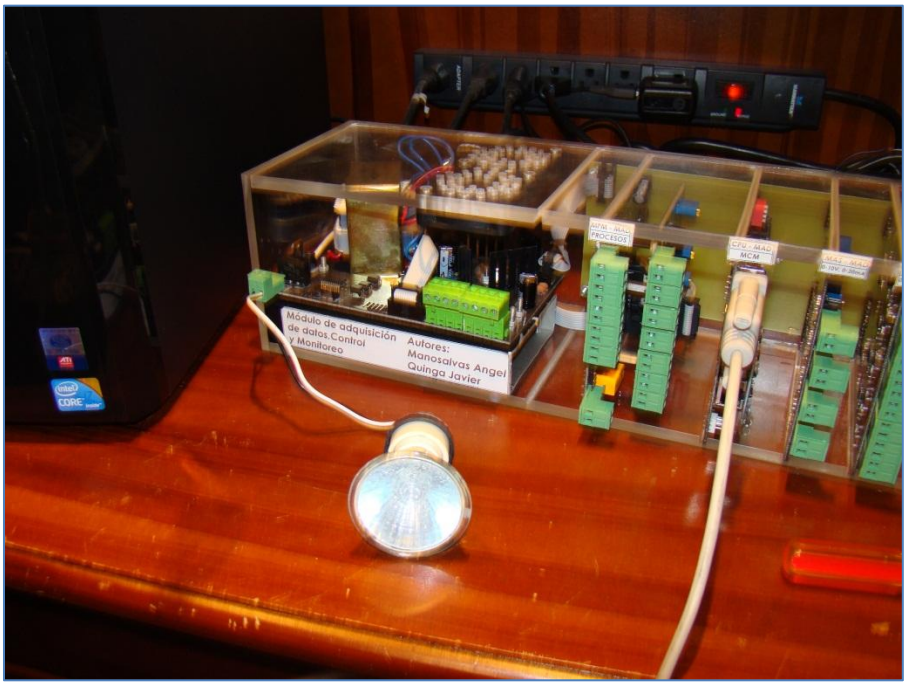

MPM - MAD
PROCESOS
CPU - MAD
MCM
adquisición
Control
eo
Autores:
Manosalvas Angel
Quinga Javier


MPM - MAD
PROCESOS
CPU - MAD
MCM
adquisición
ontroi
o
Autores:
Manosalvas Angel
Quinga Javier

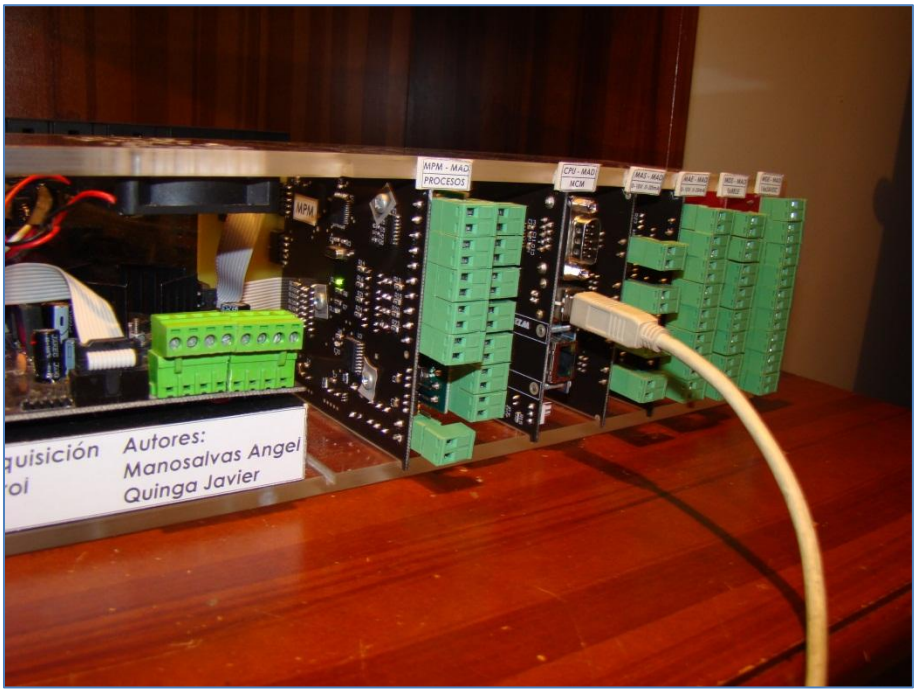

Módulo de adquisición
de datos, Control
y Monitoreo

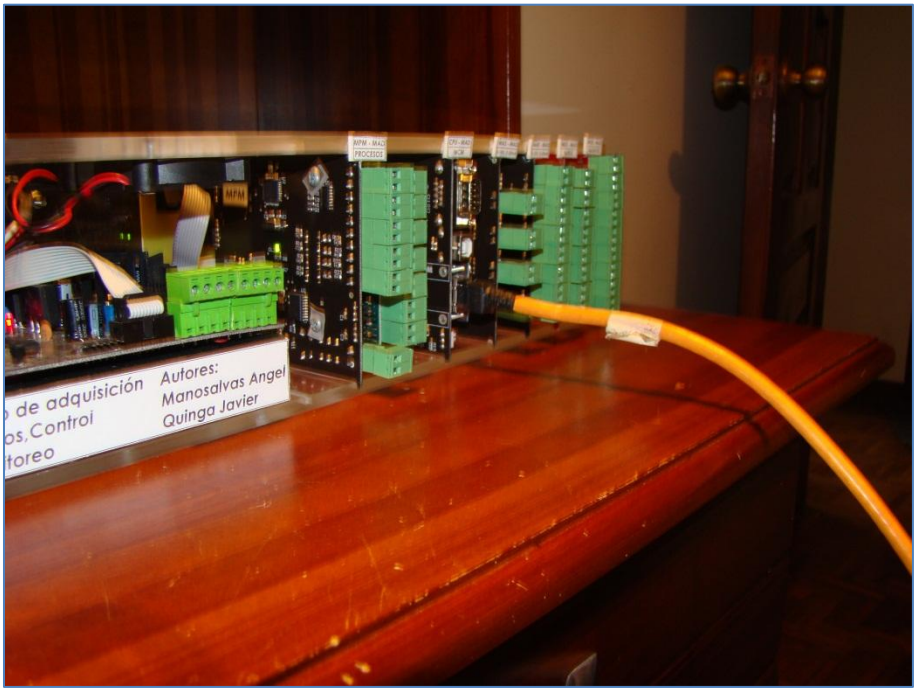
CONEXIÓN DE LÁMPARA DE TIPO DICROICA DE 50W.

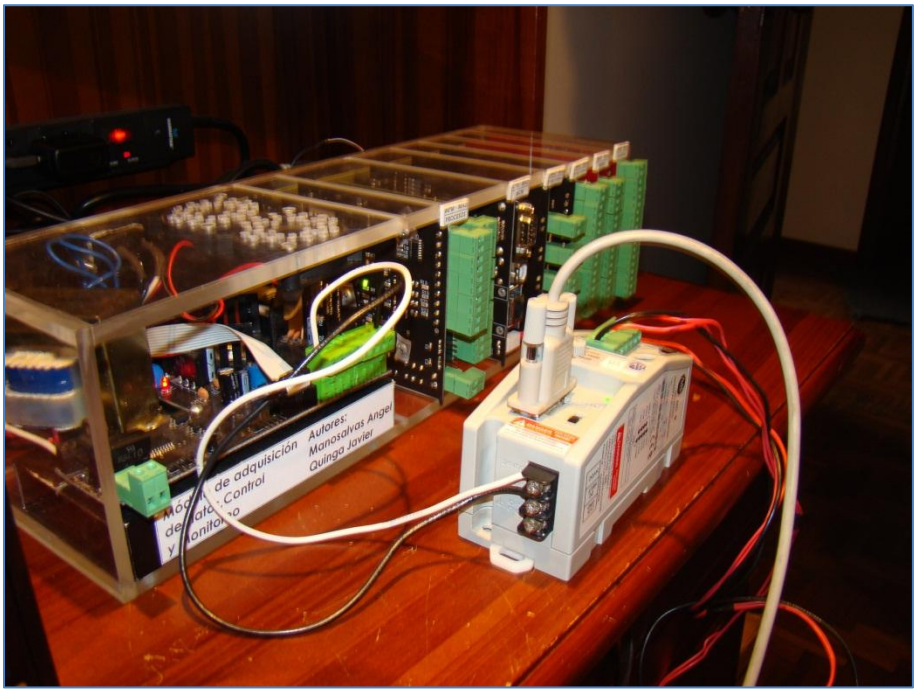


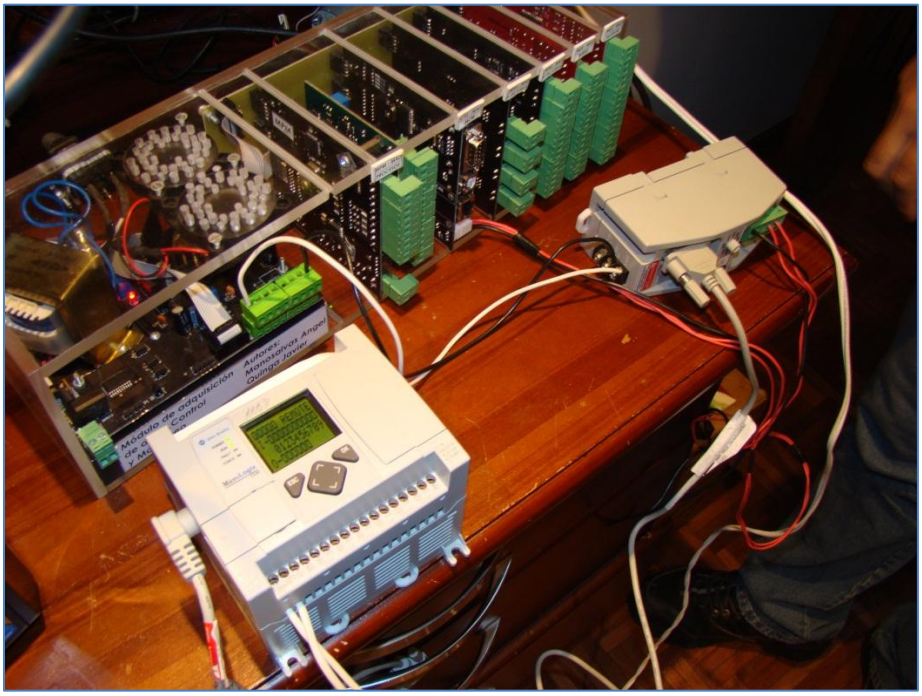


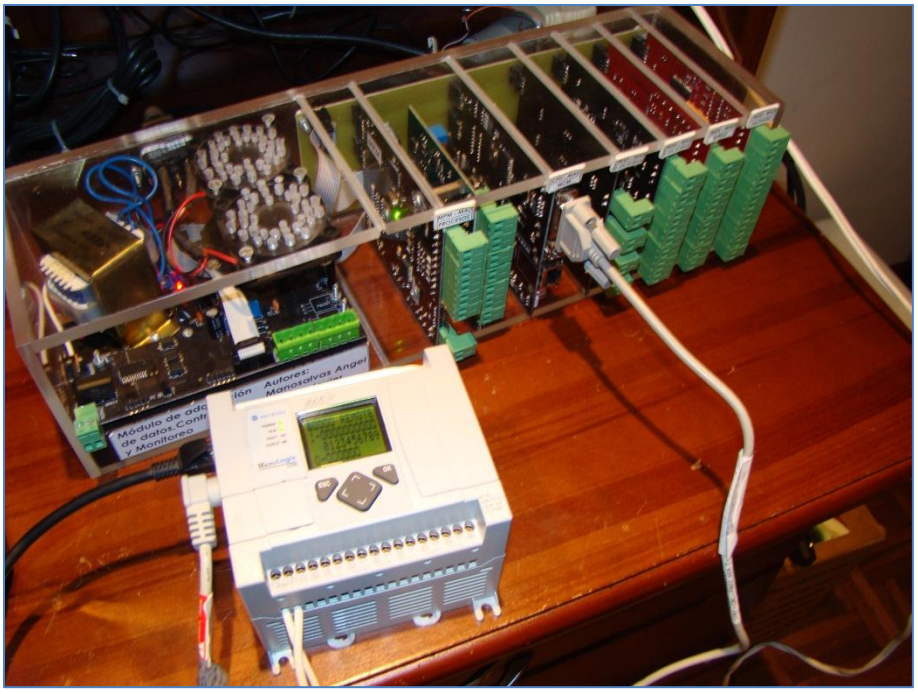
CONEXIÓN DEL MÓDULO POR USB, RS-232, RS-485, ETHERNET.

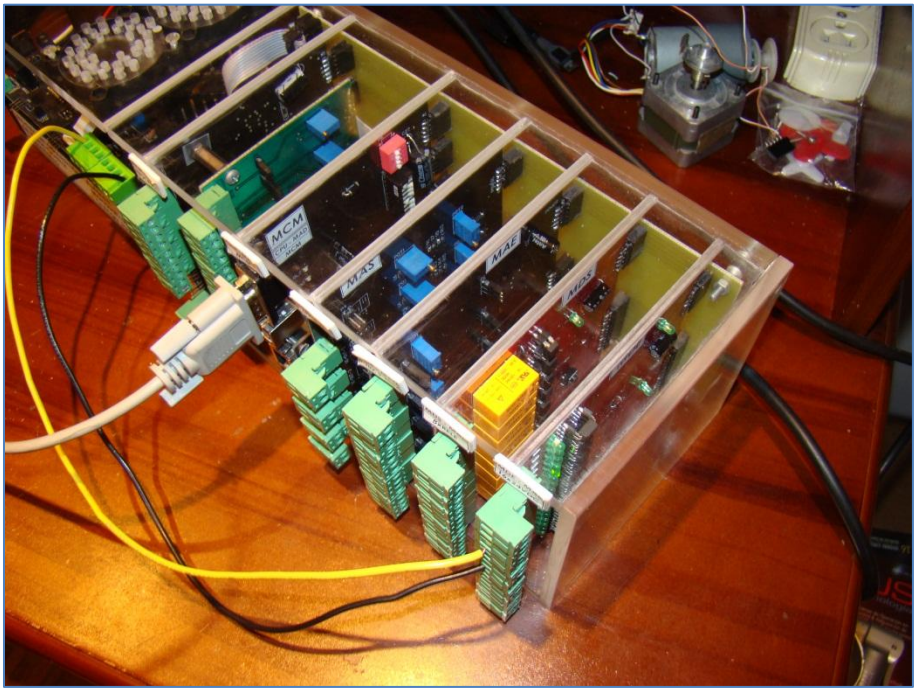




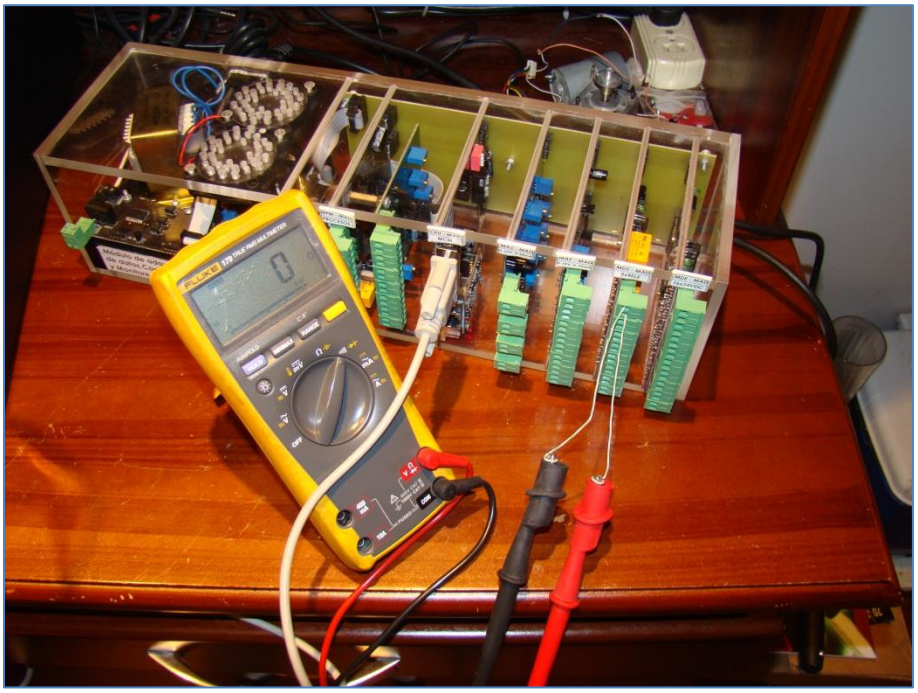

Autores:
Manosalvas Angel
Quinga Javier

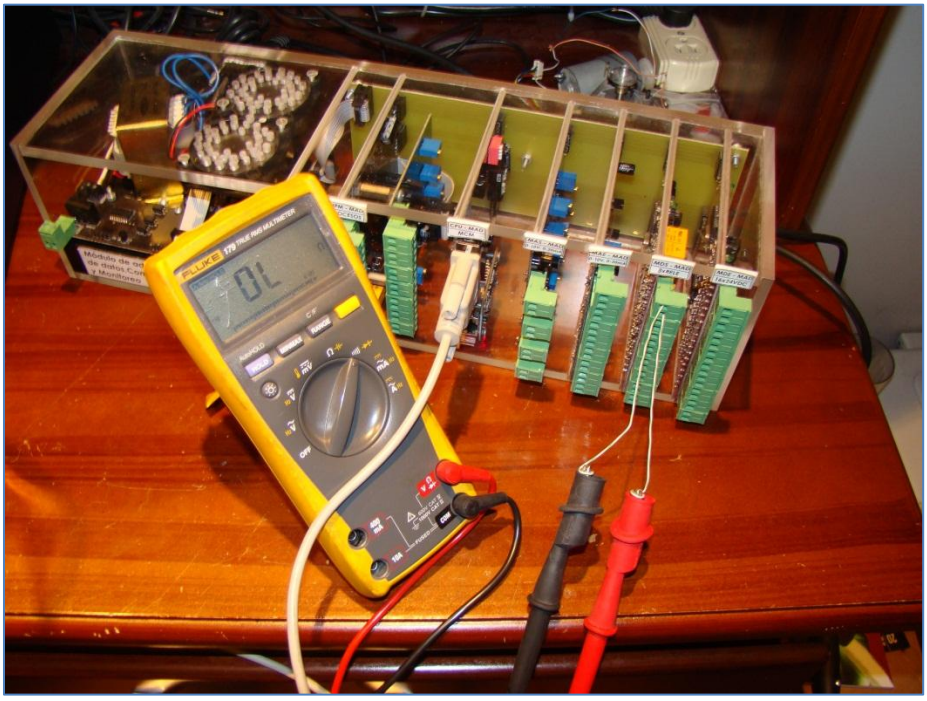

Autores:
Manosalvas Angel
Quinga Javier


Autores:
Manosalvas Angel

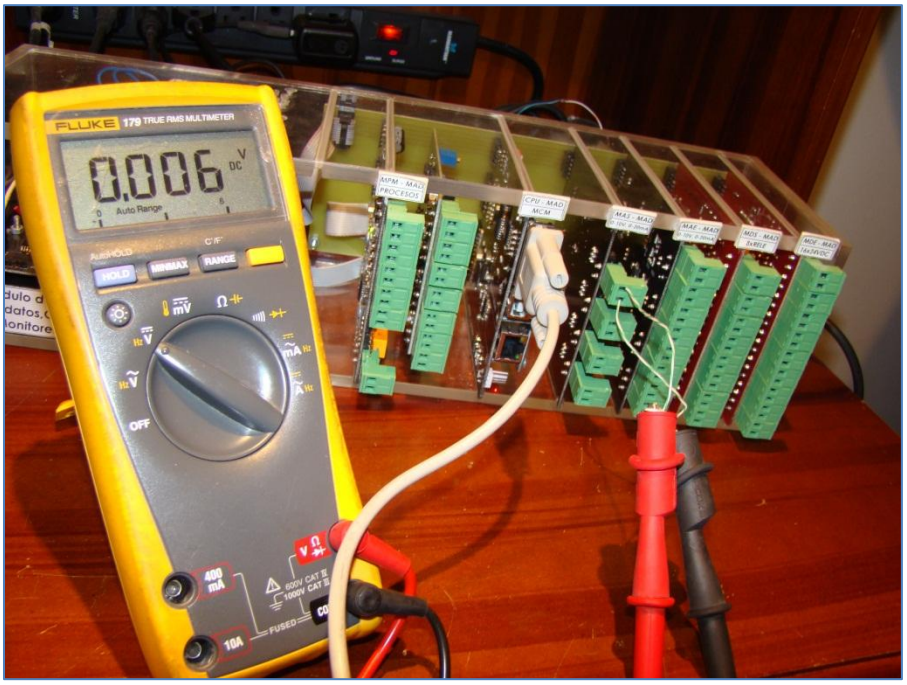

MCM
MAS
MAF

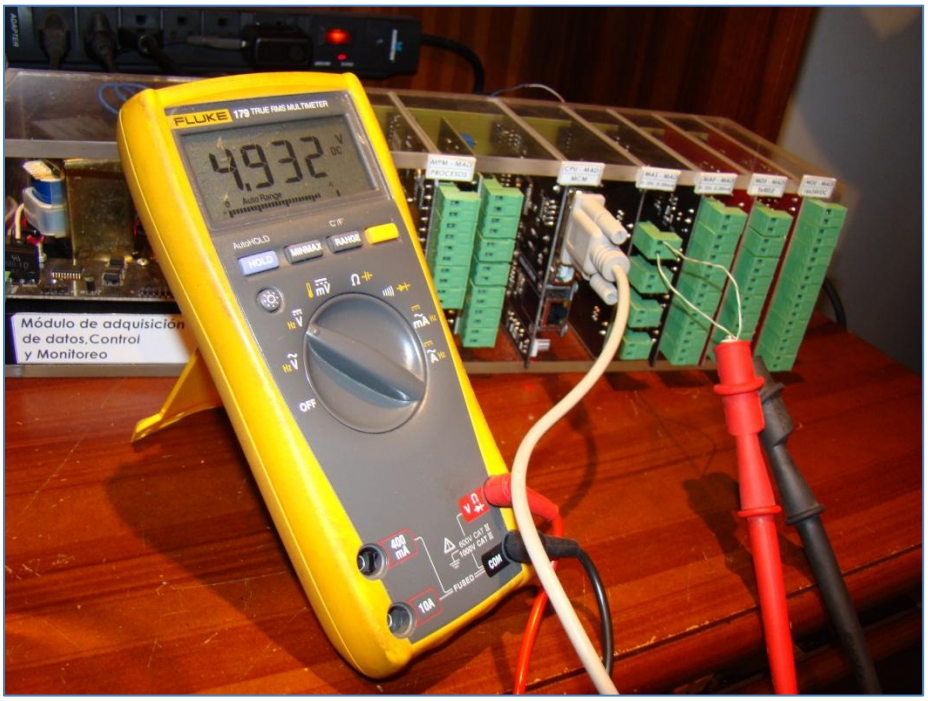
PRUEBA DE SALIDAS DE REÉ

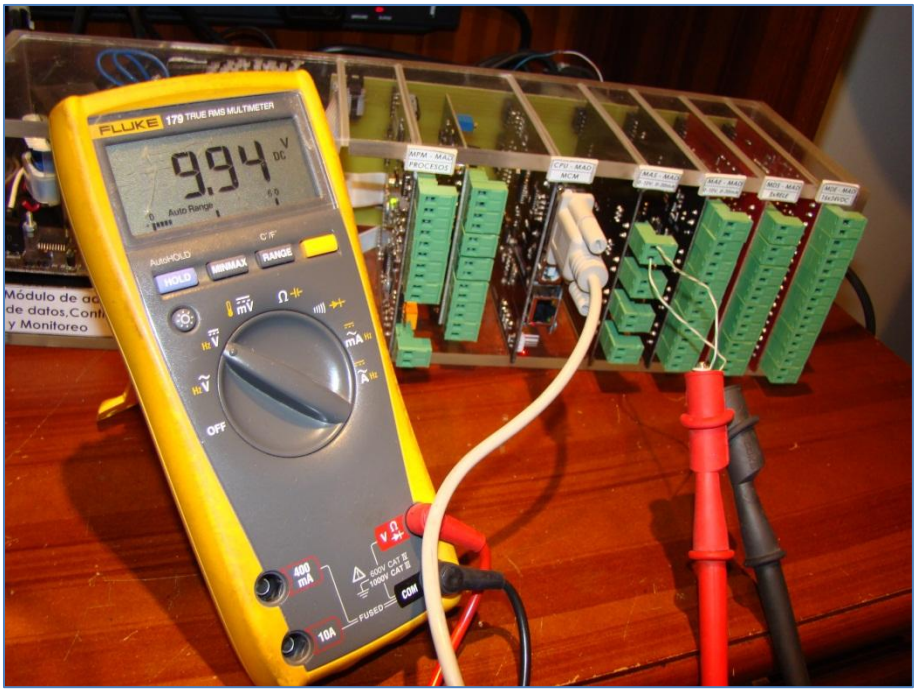




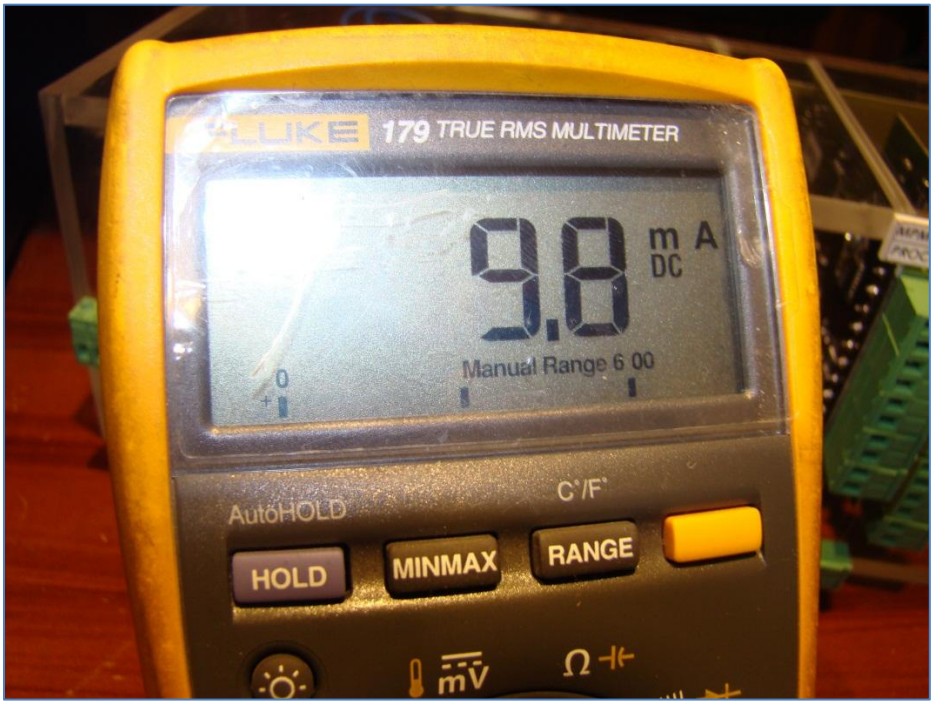
PRUEBA DE SALIDAS ANÁLOGAS EN MODO TENSIÓN DE 0 A 10 VDC





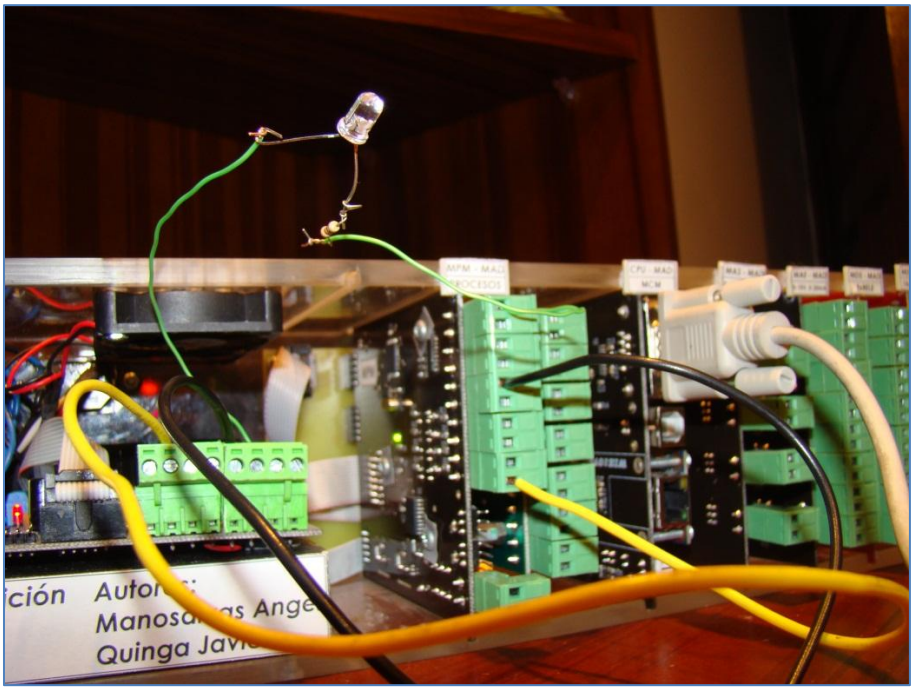

FLUKE 179 TRUE RMS MULTIMETER
9.94 V DC
Auto Range
AutoHOLD
HOLD
MIN MAX
RANGE
OFF
400 mA
10A
FUSED
COM
Módulo de ac
de datos,Cont
y Monitoreo

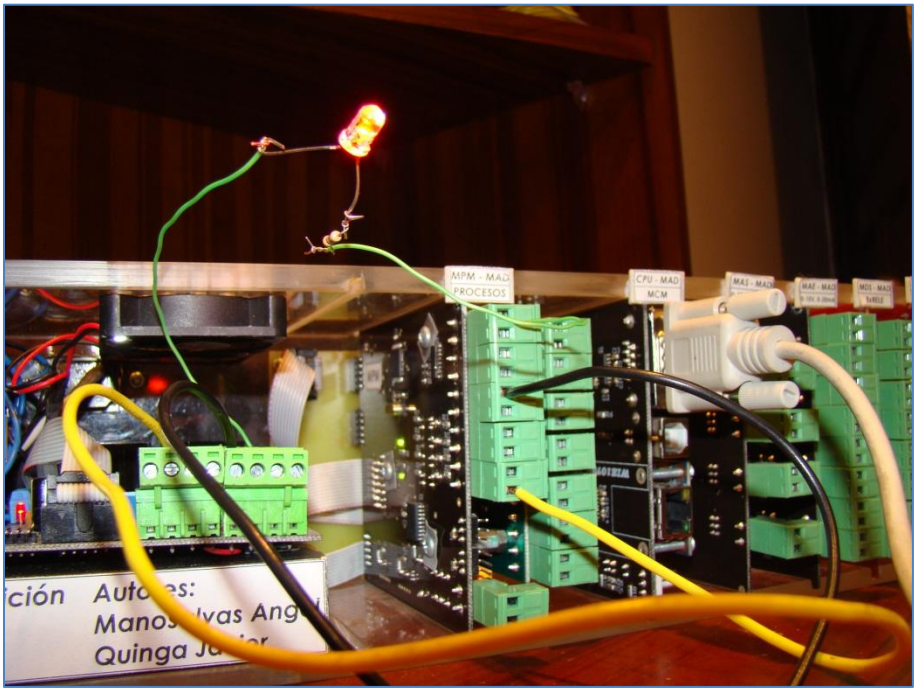
PRUEBA DE SALIDAS ANÁLOGAS EN MODO CORRIENTE 0 A 20 mA



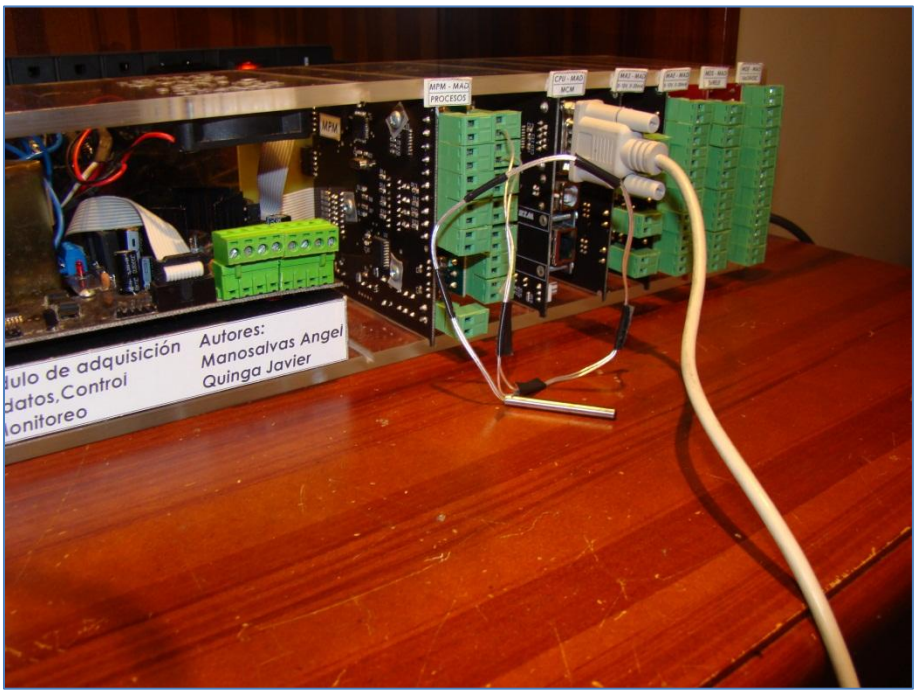

FLUKE 179 TRUE RMS MULTIMETER
19.8
HOLD
MINMAX
RANGE
OFF
Módulo de adq
de datos,Contro
y Monitoreo

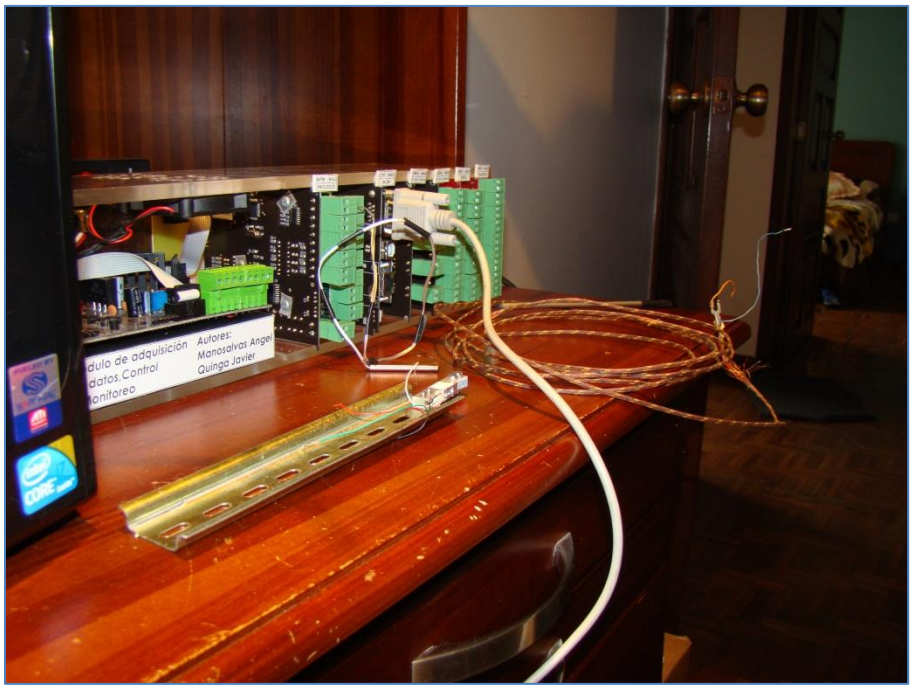
PRUEBA DE SALIDAS DIGITALES A 24 VDC





CONEXIÓN DE UNA RTD Y TERMOCUPLA AL MÓDULO





ANEXO VII

PROCESO PARA OBTENER LA PATENTE EN EL IEPI

GUIA PARA LOS SOLICITANTES DE PATENTES DE INVENCIÓN Y

MODELOS DE UTILIDAD

Este manual tiene carácter meramente informativo, por lo que para aclaraciones mas precisas se deberá consultar la Decisión 486 de la Comisión del Acuerdo de Cartagena, el Convenio de París para la Protección de la Propiedad Industrial (R.O. # 244 del 29 de Julio de 1999), así como la Ley de Propiedad Intelectual Ecuatoriana (R.O. # 320 del 19 de Mayo de 1998) con su respectivo reglamento de aplicación (R.O. # 120 del 1 de Febrero de 1999).

DOCUMENTO ELABORADO POR LA DIRECCIÓN DE PATENTES DEL IEPI
Fuentes utilizadas, las indicadas: Decisión Andina 486, Ley de Propiedad Intelectual del Ecuador y su reglamento, Manual del Inventor.

¿QUÉ ES UNA PATENTE?

Es un derecho que el Estado confiere en forma exclusiva a las invenciones.

Una patente provee a su titular el derecho a explotar industrial y/o comercialmente en forma exclusiva su invento.

La patente tendrá un plazo de duración de veinte años, contados a partir de la fecha de presentación de la respectiva solicitud si es de invención y de diez años si es modelo de utilidad.

¿QUÉ SE PUEDE PATENTAR?

Las invenciones de productos o de procedimientos en cualquier campo de la tecnología.

REQUISITOS DE PATENTABILIDAD DE LAS INVENCIONES

Para que una invención pueda ser objeto de patente debe reunir tres requisitos:

a. Novedad

Se considera que una invención es **nueva** cuando no forma parte del "estado de la técnica" anterior. Por tanto, no debe hacerse pública de ninguna manera y en ninguna parte antes de la fecha de presentación de la solicitud de Patente; en caso contrario, dicha publicidad habría incorporado la invención al "estado de la técnica" y destruiría la novedad

b. Nivel Inventivo

Se considera que una invención tiene **nivel inventivo**, cuando no se deriva de manera evidente del estado de la técnica es decir, cuando para una persona versada en la materia no resulta obvia.(Manual del Inventor- Oficina Cubana 1999)

c. Aplicación Industrial

Se considera que una invención es susceptible de **aplicación industrial** cuando la invención pueda ser fabricada o utilizada ventajosamente en cualquier industria, entendida ésta en el sentido más amplio.(Manual del Inventor- Oficina Cubana 1999)

Los requisitos de patentabilidad se juzgan con relación al **Estado de la Técnica**, el cual está constituido por todo aquello que antes de a fecha de presentación de la solicitud se ha hecho accesible al público.

¿QUÉ NO SE PUEDE PATENTAR?

No se considerarán invenciones:

a) Los descubrimientos, principios y teorías científicas y los métodos Matemáticos.
b) El todo o parte de seres vivos tal como se encuentra en la naturaleza, los procesos biológicos naturales, el material biológico existente en la naturaleza o aquel que pueda ser aislado, inclusive genoma o genoplasma de cualquier ser vivo natural.
c) Las obras literarias y artísticas o cualquier otra protegida por el derecho de autor;
d) Los planes, reglas y métodos para el ejercicio de actividades intelectuales, juegos o actividades económico-comerciales.
e) Los programas de ordenadores o el soporte lógico, como tales, y
f) Las formas de presentar información.

Se excluye de la patentabilidad expresamente:

a) Las invenciones cuya explotación comercial deba impedirse necesariamente para proteger el orden público o la moral.
b) Las invenciones cuya explotación comercial deba impedirse para proteger la salud o la vida de las personas o de los animales o para preservar los vegetales o el medio ambiente.
c) Las plantas, los animales y los procedimientos esencialmente biológicos para producción de plantas o animales que no sean procedimientos no biológicos o microbiológicos.
d) Los métodos de terapéuticos o quirúrgicos para el tratamiento humano o animal, así como los métodos de diagnóstico aplicados a los seres humanos o animales.

Los productos o procedimientos ya patentados, comprendidos en el estado de la técnica, no serán objeto de nueva patente, por el simple hecho de atribuirse un uso distinto al originalmente comprendido por la patente inicial (Art. 21 de la Decisión 486 del Acuerdo de Cartagena)

COMO SOLICITAR UNA PATENTE

DOCUMENTACIÓN NECESARIA:

La solicitud para obtener una patente de invención deberá presentarse en el formulario preparado, y puesto a disposición por la Dirección Nacional de Propiedad Industrial y deberá especificar:

a) Identificación del solicitante(s) con sus datos generales, e indicando el modo de obtención del derecho en caso de no ser él mismo el inventor.

b) Identificación del inventor(es) con sus datos generales.

c) Título o nombre de la invención

d) Identificación del lugar y fecha de depósito del material biológico vivo, cuando la invención se refiera a procedimiento microbiológico.

e) Identificación de la prioridad reivindicada, si fuere del caso o la declaración expresa de que no existe solicitud previa.

f) Identificación del representante o apoderado, con sus datos generales.

g) Identificación de los documentos que acompañan la solicitud.

A la solicitud se acompañara:

1) El título o nombre de la invención con la correspondiente memoria descriptiva que expliquen la invención de una manera clara y completa, de tal forma que una persona versada en la materia pueda ejecutarla.

2) Cuando la invención se refiera a materia viva, en las que la descripción no puedan detallarse en sí misma, se deberá incluir el depósito de la misma en una Institución depositaria autorizada por las oficinas nacionales competentes. El material depositado formará parte integrante de la descripción.

3) Una o más reivindicaciones que precisen la materia para la cual se solicita la protección mediante la patente.

4) Dibujos que fueren necesarios

5) Un resumen con el objeto y finalidad de la invención.

6) El Comprobante de Pago de la Tasa.

7) Copia certificada, traducida y legalizada de la primera solicitud de patente que se hubiere presentado en el exterior, en el caso de que se reivindique prioridad

8) El documento que acredite la Cesión de la invención o la relación laboral entre el solicitante y el inventor.

9) Nombramiento del Representante Legal, cuando el solicitante sea una persona jurídica.

10) Poder que faculte al apoderado el tramitar la solicitud de registro de la patente, en el caso de que el solicitante no lo haga el mismo.

Demás documentos necesarios.

Los documentos que se presenten para la obtención de la patente deberán estar redactados en idioma castellano y debidamente traducido si lo necesitara

Cabe aclarar que la Dirección Nacional de Propiedad Industrial no admitirá a trámite aquellas solicitudes que no contengan por lo menos los documentos mencionados en los numerales 1, 3, 4, 5 y 6. Es decir si a la solicitud se le acompañan por lo menos estos documentos se otorgará la fecha de presentación de la solicitud, que da origen al nacimiento del **derecho de prioridad** (Art. 33 de la Decisión 486 del Acuerdo de Cartagena)

LUGAR DE PRESENTACION

Las solicitudes de patente y demás documentos que deban acompañarse a las mismas deberán entregarse a la Dirección Nacional de Propiedad Industrial.

COMO PRESENTAR SU SOLICITUD DE PATENTE.

- La Solicitud de Patente debe presentarse por triplicado, en papel blanco, fuerte y duradero, de **formato A4** (29.7cm x 21cm).

- La memoria descriptiva y las reivindicaciones deben presentarse por duplicado.

- Las hojas no deben estar desgarradas, arrugadas ni dobladas. Solo deben utilizarse por una cara

- Las hojas estarán unidas de forma que puedan pasarse fácilmente durante su consulta y separarse y volverse a unir de nuevo sin dificultad.

- Cada hoja debe ser utilizada en sentido vertical, salvo lo dispuesto para dibujos.

- Cada uno de los documentos de la solicitud de patente (solicitud, descripción, reivindicaciones, resumen etc.) debe comenzar en una nueva hoja.

- Los márgenes deben estar en blanco.

- Las **hojas** de la descripción, reivindicaciones y dibujos deben estar numeradas **correlativamente** en cifras árabes. La numeración debe ir centrada en la parte superior de cada hoja, respetando el margen superior.

- Las **líneas** de cada hoja de la descripción y de las reivindicaciones deben ser numeradas de cinco en cinco, situándose esta numeración en la parte izquierda, a la derecha del margen y comenzando una nueva numeración en cada hoja.

- La solicitud de patente, la memoria descriptiva, el resumen, y las reivindicaciones deben estar mecanografiadas o impresas, con color negro e indeleble. Únicamente los símbolos y caracteres gráficos y las fórmulas químicas o matemáticas podrán estar manuscritos o dibujados.

- La terminología y los signos de la solicitud de patente deben ser uniformes.

- Las hojas deben estar razonablemente exentas de borraduras y no contener correcciones, tachaduras ni interlineaciones.

SOLICITUD DE PATENTE.

La solicitud por la que se solicita la patente va firmada por el solicitante o su apoderado.

- Datos que debe contener:

a) **Nombre**, apellidos, domicilio, ciudad, nacionalidad, teléfono, y demás datos **del solicitante**. Si se trata de una persona jurídica, se identificará por su razón social o de acuerdo con las disposiciones legales por las que se rija.

b) **Designación del inventor o inventores**. Si el solicitante no es el inventor o único inventor, debe señalarse en la casilla correspondiente de la solicitud el modo de adquisición del derecho.

c) **Título de la invención que se desea proteger**. Este título debe ser claro, conciso, debe designar técnicamente la invención y estar en congruencia con las reivindicaciones.

d) **Prioridad.** Si la solicitud de patente se basa en la prioridad de un depósito anterior en otro país de la Comunidad Andina, deberá incluir una Declaración de Prioridad. Esta Declaración de Prioridad, debe indicar la fecha de presentación de la solicitud anterior, el Estado en el cual se ha solicitado, así como el número que se le ha asignado. Se debe presentar también una copia de la solicitud anterior, certificada conforme por la Oficina de Origen, con indicación de su fecha de depósito y una traducción al castellano de la misma si la solicitud prioritaria no está redactada en este idioma.

e) De ser el caso, la **copia del contrato de acceso**, cuando los productos o procedimientos cuya patente se solicita han sido obtenidos o desarrollados a partir de recursos genéticos o de sus productos derivados de los que cualquiera de los Países Miembros es país de origen.

f) De ser el caso, la **copia del documento que acredite la licencia o autorización de uso de los conocimientos tradicionales** de las comunidades indígenas afro-americanas o locales de los Países Miembros, cuando los productos o procedimientos cuya protección se solicita han sido obtenidos o desarrollados a partir de dichos conocimientos de los que cualquiera de los Países Miembros es país de origen, de acuerdo a lo establecido en la Decisión 391 (del Medio Ambiente) y sus modificaciones y reglamentaciones vigentes.

g) De ser el caso el **certificado de depósito de material biológico**

h) **Representante Legal / Apoderado.** En cualquiera de los dos casos se deberá presentar los documentos pertinentes que acrediten tal nombramiento.

i) **Relación de documentos.** Por último, se indicará en la solicitud qué documentos se acompañan a la misma.

g) **Firmas** del solicitante y del abogado patrocinador.

RESUMEN:

Debe colocarse en la primera hoja normalizada entregada junto con la solicitud.

El **resumen** debe constituir en un instrumento eficaz de cara a efectuar búsquedas en un dominio técnico determinado. El objetivo del resumen es la información técnica, no pudiendo utilizarse para otro fin y en ningún caso para definir el alcance de la protección solicitada.

a. Deberá indicar el título de la invención.

b. Tendrá una extensión máxima de 150 palabras.

c. Deberá contener una exposición concisa del contenido de la descripción, reivindicaciones y en su caso, dibujo más característico que deberá situarse separadamente del texto; así mismo se podrá indicar la fórmula química que, entre las que figuran en la solicitud de patente caracterice mejor la invención.

d. No debe contener declaraciones sobre méritos, ventajas o valores de la invención.

e. Debe contener las siguientes partes bien diferenciadas:

- Objeto de la invención.
- Descripción de la invención.
- Aplicaciones, solución técnica que aporta la invención, concretando el problema que resuelve el dispositivo o procedimiento de dicha invención.
- Alternativas.
- Dibujo y/o fórmula.

Si el objeto de la invención es un dispositivo o un aparato, el resumen deberá contener los elementos más relevantes de que consta el mismo, con referencias entre paréntesis a las partes de la figura más representativa que acompañará el texto del resumen.

El texto del resumen y la figura que le acompaña deben ir en la primera hoja normalizada que se entrega con la solicitud.

MEMORIA DESCRIPTIVA

La memoria descriptiva estará redactada en la forma más concisa y clara posible, sin repeticiones inútiles y en congruencia con las reivindicaciones.

En la misma se indicarán los siguientes datos Art. 28 de la Decisión 486 del Acuerdo de Cartagena:

1. **Título** de la invención, tal como fue redactado en la solicitud.

2. Indicación del **sector de la técnica** al que se refiere la invención. (clase internacional)

3. Indicación del **estado de la técnica** anterior a la fecha de presentación, son los antecedentes de la invención conocidos por el solicitante.

4. **Explicación** de la invención de una manera clara y completa, que permita una comprensión del problema técnico planteado (problema solución) así como la solución del mismo, indicándose en su caso, las ventajas de la invención en relación al estado de la técnica anterior y la forma que indique al experto poder llevarla a la práctica. Esta descripción permitirá al especialista en el área, definir la novedad, el nivel inventivo y la aplicación industrial, para determinar el cumplimiento de los requisitos de patentabilidad. Deberá describirse por tanto y si es el caso ejemplos aclaratorios, fórmulas o referencias del invento.

5. **Descripción de los dibujos**, si los hubiera, los que deberán adjuntarse al final de la memoria y reivindicaciones, pero que deberán estar perfectamente señalizados en la memoria, al referirse a los mismos.

La descripción debe presentarse preferentemente en el orden mencionado.

Cuando la invención se refiera a material biológico, (Art. 29 de la Decisión 486 del Acuerdo de Cartagena) la descripción deberá cumplir los siguientes requisitos:

a. Que la descripción contenga las informaciones de que disponga el solicitante sobre las características del microorganismo.

b. Que el solicitante hubiere depositado no más tarde de la fecha de presentación de la solicitud un cultivo de microorganismos en una Institución autorizada para ello, conforme a los Convenios Internacionales.

Así mismo, el solicitante deberá indicar en la descripción el nombre de la Institución autorizada donde haya depositado una muestra del cultivo del microorganismo y consignar el número o clave de identificación de dicho microorganismo por la Institución autorizada.

COMO REDACTAR LAS REINVINDICACIONES.

Las reivindicaciones definen el objeto para el que se solicita la protección. Deben ser claras y concisas y han de fundarse en la descripción.

Debe contener:

- Un **preámbulo** o introducción en el que se indica cual es el objeto de la invención, que suele coincidir con el título y el propósito de la invención, y todas aquellas características técnicas que, aunque conocidas, son necesarias para la definición de los elementos que se van a proteger.

- Una parte **caracterizadora** precedida por la expresión "caracterizada por", "en el que la mejora comprende", ¨ que consiste en ¨ o una similar, en la que se exponen de manera concisa las características técnicas nuevas que se desean proteger.

Si la claridad y comprensión de la invención lo exigiera, la reivindicación esencial puede ir seguida de una o varias reivindicaciones dependientes que precisen las características adicionales que se deseen proteger, así como modos particulares o alternativos de realización de la invención. En este caso, se numerarán correlativamente. (Art. 30 de la Decisión 486 del Acuerdo de Cartagena)

Salvo en casos de absoluta necesidad, porque de otra forma no se entendieran las reivindicaciones, éstas deben tener carácter autónomo, es decir, no deben hacer referencia a la descripción y a los dibujos; se debe evitar, por lo tanto, expresiones del tipo ¨ como se describe en la parte... de la descripción ¨ o ¨ como se ilustra en la figura... de los dibujos ¨.

Únicamente si la solicitud contiene figuras, se indicarán entre paréntesis a continuación de las características técnicas mencionadas en las reivindicaciones los números correspondientes a dichas características.

DIBUJOS

La superficie útil de las hojas que contengan los dibujos no debe exceder de 26,2 cm x 17 cm. Las hojas no contendrán marco alrededor de su superficie útil ni alrededor de la superficie utilizada.

Los dibujos se realizarán de la forma siguiente:

- Ejecutado en líneas y trazos duraderos, negros, suficientemente densos y entintados, bien delimitados.

- Los cortes se indicarán mediante líneas oblicuas que no impidan la fácil lectura de los signos de referencia y de las líneas directrices.

- La escala de los dibujos y la claridad de su ejecución deberán ser tales que una reproducción fotográfica con reducción lineal a dos tercios permita distinguir sin dificultad todos los detalles. Si se emplea escala, ésta debe ser gráfica.

- Todos los signos de referencia que figuren en los dibujos deben ser claros. No se utilizarán paréntesis, círculos o comillas, en combinación con cifras y letras.

- Todas las líneas deben ser, en principio, trazadas con instrumentos de dibujo técnico.

- La altura de las cifras y letras no debe ser inferior a 0,32 cm.

- Una misma hoja de dibujo puede contener varias figuras.

- Cuando las figuras dibujadas sobre varias hojas estén destinadas a constituir una sola figura del conjunto de ellas, deben disponerse de tal forma que la figura del conjunto pueda componerse sin que quede oculta ninguna parte de las figuras que lo componen.

- Las distintas figuras deben estar dispuestas preferentemente en sentido vertical, claramente separadas unas de otras. Si se disponen horizontalmente debe situarse la parte superior de las figuras en el lado izquierdo de la hoja.

- Las figuras deben estar numeradas correlativamente en cifras árabes, independientemente de la numeración de las hojas.

- Los signos de referencia pueden ser utilizados en los dibujos sólo si figuran en la descripción y viceversa.

- Los dibujos no deben contener texto alguno, a excepción de breves indicaciones indispensables, tales como "agua", "vapor", ¨ corte "según AB", y las palabras claves para su comprensión.

- Los diagramas se considerarán como dibujos.

TRAMITACION DE LA SOLICITUD DE PATENTE

1.- ADMISION A TRÁMITE Y OTORGAMIENTO DE FECHA DE PRESENTACIION

La solicitud que cumpla con los requisitos básicos será admitida a trámite, y la Dirección Nacional de Propiedad Industrial procederá a certificar la fecha y hora de presentación asignándole un número de orden.

2. EXAMEN DE LA SOLICITUD

Dentro de los **treinta días hábiles** siguientes a la fecha de presentación, la Dirección Nacional de Propiedad Industrial procederá a examinar si la solicitud reúne todos los requisitos necesarios para seguir con el trámite. Si del examen resulta que no cumple con tales requisitos, se lo hará saber al solicitante para que la complete dentro del **plazo de dos meses** contados desde la fecha de notificación. Dicho plazo será **prorrogable** por una sola vez y por un periodo igual, sin que pierda su prioridad. Si transcurrido dicho plazo el solicitante no completó los requisitos, la Dirección Nacional de Propiedad Industrial declarará **abandonada** la solicitud y

perderá su prelación, sin embargo de lo cual la oficina guardará la confidencialidad de la solicitud.(Art.39 de la Decisión 486 del Acuerdo de Cartagena)

3. PUBLICACION DE LA SOLICITUD

Transcurridos 18 meses desde la fecha de presentación de la solicitud o desde la fecha de prioridad, la Dirección Nacional de Propiedad Industrial ordenará la publicación de la solicitud.

El solicitante podrá también en cualquier momento luego de superado el examen de forma, solicitar que se publique un extracto de su solicitud en la Gaceta de la Propiedad Intelectual. (Art. 40 de la Decisión 486 del Acuerdo de Cartagena)

Mientras la publicación no se realice, el expediente será reservado, para terceros, y sólo podrá ser examinado por terceros con el consentimiento escrito del solicitante o cuando el solicitante hubiere iniciado acciones judiciales o administrativas contra terceros fundamentando en la solicitud.(Art. 141 de la Ley de Propiedad Intelectual)

4. OPOSICIONES

Dentro del **plazo de sesenta días** siguientes a la fecha de la publicación, quien tenga legítimo interés (terceros) podrán presentar por una sola vez, oposiciones fundamentadas que puedan desvirtuar la patentabilidad, a petición de la parte interesada se otorgará por una sola vez un plazo adicional, por igual tiempo (Art. 42 de la Decisión 486 del Acuerdo de Cartagena)

Quien presente una oposición sin fundamento responderá por los daños y perjuicios.

Presentada una oposición, se **notificará al solicitante concediéndole un plazo de sesenta días** contados a partir de la notificación, plazo que podrá ser **prorrogable** por una sola vez y por el mismo lapso, para que haga valer si lo estima conveniente, sus argumentos, presente documentos o redacte nuevamente las reivindicaciones o la descripción de la invención.(Art. 43 de la Decisión 486 del Acuerdo de Cartagena)

5. EXAMENES DE FONDO

Dentro del **plazo de seis meses** desde la fecha de publicación de la solicitud, e independientemente de que se hayan presentado oposiciones a la misma, el solicitante deberá pedir que se realice el examen que determinará si la solicitud es o no patentable. Vencido dicho plazo, la solicitud se considerará **abandonada**. (Art. 44 de la Decisión 486 del Acuerdo de Cartagena)

Para dicho examen se podrá, requerir el informe de expertos o de organismos científicos o tecnológicos que se consideren idóneos, para que emitan opinión sobre la novedad, nivel inventivo y aplicación industrial de la invención. Así mismo, cuando lo estime conveniente, podrá requerir informes de oficinas nacionales competentes de otros países. Incluso la Dirección Nacional de Propiedad Industrial de considerar necesario podrá requerir al solicitante información sobre solicitudes extranjeras dentro de un **plazo que no excederá de 3 meses,** si no se presenta dicha información dentro del tiempo concedido, **la patente se denegará**.

Si durante el examen se encontrare que la solicitud no cumple con alguno de los requisitos necesarios, se le requerirá por escrito al solicitante para que dentro de **plazo de sesenta días** contados a partir de la notificación de respuesta; dicho plazo será prorrogable por una sola vez por un periodo **de treinta días adicionales**. Si no se diere respuesta en el tiempo concedido o si dichas respuestas no dilucidaran los problemas, la patente será **denegada**. (Art. 45 de la Decisión 486 del Acuerdo de Cartagena)

6. CONCESION DE LA PATENTE.

Si el resultado del examen fuere favorable, se otorgará el título de concesión de la patente. Si fuere parcialmente desfavorable, se otorgará la patente solamente para las reivindicaciones aceptadas. Si fuere desfavorable se denegará.

La concesión implica el pago de los derechos de concesión: abonados éstos, se expide el correspondiente Título de Patente, y para mantener vigente la patente se deberán pagar las anualidades correspondientes.

DERECHO DE PRIORIDAD

Para proteger una invención en otros países miembros del Convenio de la Unión de París y de la Organización Mundial de Comercio, el solicitante puede ejercer el **derecho de prioridad;** es decir, cuando una persona (natural o jurídica) presenta una solicitud en un país de la Unión, tiene un plazo de 12 meses para presentarla en otro u otros países miembros del Convenio de París que por determinadas razones resulten de su interés y reclamar la fecha correspondiente a la primera solicitud presentada. El Ecuador es país miembro.

MANTENIMIENTO DE LA PATENTE

Para mantener vigente los derechos que confiere la patente, o en su caso para mantener vigente la solicitud de patente en trámite, deberán pagarse las tasas periódicas establecidas por la Ley de Propiedad Intelectual y por la Resolución CD-IEPI-99-008, publicada en el R.O. No.336 del 10 de Diciembre de 1999, y por la resolución CD-IEPI 01- 082, publicada en el R.O. 389 del 14 de agosto del 2001, dichas anualidades deberán pagarse por años adelantados, teniendo como fecha de vencimiento de cada anualidad el último día del mes en que fue presentada la solicitud.

La Dirección Nacional de Propiedad Industrial concederá un plazo de gracia de seis meses contados desde la fecha de inicio del periodo anual correspondiente, a fin de que el interesado cumpla con el pago de las tasas mencionadas mas el recargo, antes de declararla Caducada. (Art.80 de la Decisión 486 del Acuerdo de Cartagena)

Durante el plazo de gracia, la patente o la solicitud de patente mantendrán su vigencia plena.