Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Instituto Costarricense de Electricidad

Centro Nacional de Control de Energía

ICE – CENCE

Prototipo de interfaz de indicaciones de alarmas remotas para la red de comunicaciones del ICE Sector Electricidad.

Informe de Proyecto de Graduación para optar por el Grado de Bachiller en Ingeniería Electrónica

Carlos Iván Castillo Santamaría

Cartago, Junio del 2002

RESUMEN

Con el fin de monitorear y controlar todas las subestaciones y plantas de generación de energía del país, el Centro Nacional de Control de Energía (CENCE) debe mantener comunicación continua las 24 horas del día a través de la red de comunicaciones del ICE sector electricidad. Debido a la importancia de la información que se maneja en cada uno de estos sitios, y a la necesidad de conocer la misma en todo momento, se debe garantizar la confiabilidad y continuidad de dicha red.

De gran importancia para esta red son los sitios de repetición, ya que estos son los que se encargan de retransmitir las comunicaciones entre el centro de control y las subestaciones, así como las plantas generadoras. Estas retransmisiones se deben ya sea a la ubicación geográfica de algunos de estos sitios, o bien a las largas distancias. Si los sitios de repetición dejan de funcionar, detienen el tráfico de comunicación entre el centro de control y las subestaciones o plantas generadoras.

Es por ello que se debe conocer el estado de todos los sitios de repetición de la red de comunicaciones, para poder realizar un mantenimiento preventivo, o bien, inmediato al momento en que se presenta alguna emergencia en el mismo. Todas las condiciones que pueden provocar a corto o mediano plazo una alarma en los sitios de repetición deben ser monitoreadas, estas fuentes de alarmas se obtienen de señales analógicas y digitales del equipo presente en el sitio.

Para cumplir con esta necesidad, se desarrolló el prototipo del hardware que se requiere en cada sitio para reportar remotamente las alarmas que se preseten. Este hardware utiliza protocolo Modbus sobre una interfaz RS-232, para comunicarse remotamente con el CENCE; además, monitorea un dispositivo Modbus en el sitio que reporta el estado de la planta de emergencia del mismo. La solución realizada se basa en la utilización de un microcontrolador PIC18C252.

Palabras claves: Protocolo Modbus, Protocolo I^2C, PIC18C252, CENCE, MPLAB C18, Monitoreo remoto de alarmas.

ABSTRACT

With the purpose of monitor and to control all the substations and plants of generation of energy of the country, the National Center of Control of Energy (CENCE) it should maintain continuous communication the 24 hours of the day through the network of communications of the ICE sector electricity. Due to the importance of the information that is managed in each one of these places, and to the necessity of knowing this information in all moment, it should be guaranteed the dependability and continuity of this network.

Very importance for this network are the repetition places, they take charge to re-transmit the communications between the control center and the substations, as well as the generating plants. These retransfers are either owed to the geographical location of some of these places, or at the long distances. If the repetition places stop to work, they stop the communication traffic between the control center and the substations or generating plants.

For these reasons it is that the state of all the places of repetition of the network of communications should be known, to be able to implement a preventive maintenance, or, immediate to the moment in that some emergency is presented. All the conditions that can cause to short or medium term an alarm in the repetition places should be reported, these sources of alarms are obtained of analogical and digital signs of the present equipment in the places.

To fulfill this necessity, it was developed the prototype of the hardware that is required in each place to report the remote alarms. This hardware uses protocol Modbus on an RS-232 interfaz, to communicate remote alarms to the CENCE; also, it checks a Modbus device in the place that reports the state of the emergency plant. The solution is based on the use of a PIC18C252 microcontroler.

Key words: Modbus Protocol, I^2C Protocol, PIC18C252, CENCE, MPLAB C18, alarms remote report.

A mis padres Carlos y Elida, mi hermana Keilyn, los seres más importantes de mi vida, así como el más reciente miembro de la familia, Jose Carlo, ustedes son mi inspiración y ejemplo, apoyo y consuelo constante en mi vida.

Agradezco a la familia Rodríguez Santamaría, a Juan Rafael, Esperanza, Juan Diego y Kimberly, sin su apoyo, compresión y cariño durante estos 5 años, no hubiera podido alcanzar mi meta.

Agradezco al Ingeniero Rodrigo Chacón por la oportunidad que me brindó de realizar el proyecto de graduación en el Centro de Control de energía; así como a todo el personal del CENCE, que durante estos 4 meses me consideró como un compañero más.

ÍNDICE GENERAL

Indice

ÍNDICE DE FIGURAS

Indice

Indice

ÍNDICE DE TABLAS

CAPITULO 1
INTRODUCCIÓN

1.1 Descripción de la empresa

El Instituto Costarricense de Electricidad (ICE) fue creado por el Decreto - Ley No.449 del 8 de abril de 1949 como una institución autónoma, con personalidad jurídica y patrimonio propio. Está dotado de plena autonomía e independencia administrativa, técnica y financiera. Al ICE le corresponde, por medio de sus empresas, desarrollar, ejecutar, producir y comercializar todo tipo de servicios públicos de electricidad y telecomunicaciones, así como actividades o servicios complementarios a estos, con el fin de promover el mayor bienestar de los habitantes del país y fortalecer la economía nacional.

En la actualidad cuenta con 8609 empleados, siendo su presidente ejecutivo el Ing. Pablo Cob Saborío. El ICE está constituido por los sectores: ICE-Electricidad e ICE-Telecomunicaciones y sus empresas, la Compañía Nacional de Fuerza y Luz (CNFL) y Radiográfica Costarricense S.A. (RACSA), es un grupo que se ha caracterizado por una gestión de desarrollo de clase internacional tendiente a satisfacer las necesidades evolutivas que plantean los clientes y un entorno altamente competitivo.

Es así como, con más de medio siglo de existencia, el ICE ha logrado la construcción de numerosas obras hidroeléctricas, térmicas y geotérmicas; la instalación de paneles solares en comunidades alejadas y la producción de energía eólica. La administración superior del ICE está organizada de la forma que se muestra en la figura 1.1:

Indice

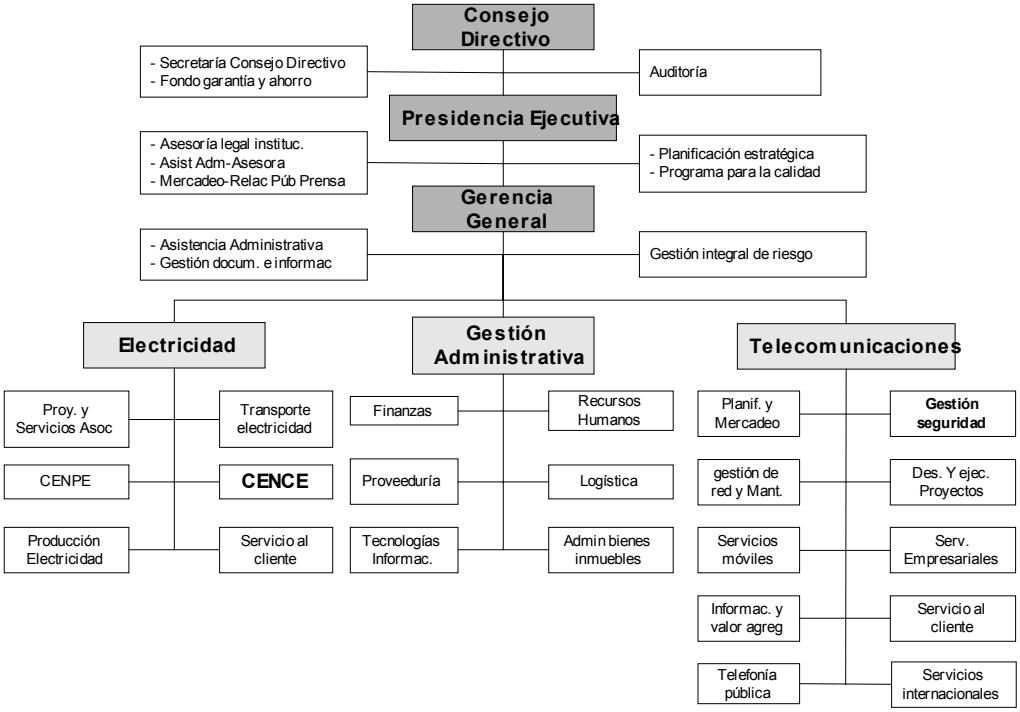


Figura 1.1 Estructura organizacional del ICE.

El proyecto de graduación se realizó en el sector ICE-Electricidad, y como se muestra en la figura 1.2, el sector electricidad está organizado en unidades estratégicas de negocios.

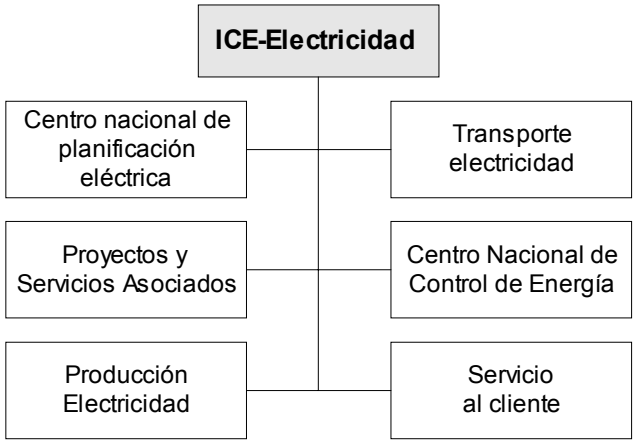


Figura 1.2 Organización de unidades estratégicas de negocio del sector ICE-electricidad.

El lugar donde se realizó el proyecto de graduación fue en la UEN: Centro Nacional de Control de Energía (CENCE).

Indice

El CENCE se ubica 400 metros norte de la esquina sureste del edificio principal del ICE en La Sabana. El director del departamento es el: Ing. Jorge Zamora Soto. Cuenta con un total de 97 empleados, 25 de ellos son ingenieros en las siguientes especialidades: electrónica de potencia, control automático, informática, ingeniería eléctrica y telemática.

El Centro Nacional Control Energía es la unidad estratégica de negocio de ICELEC responsable del Planeamiento y Pre-despacho del Sistema Eléctrico Nacional, además de la coordinación de la operación integrada con el propósito de garantizar el suministro continuo y de calidad de la energía eléctrica, por medio de un marco de optimización y eficiencia económica de la Generación y Transmisión.

Por otra parte, realiza las evaluaciones pertinentes de los paros programados de las unidades generadoras, líneas de transmisión y subestaciones cumpliendo con el plan de maniobras sobre la operación del SEN (Sistema eléctrico nacional): plan de mantenimiento, conexión y desconexión. Además, monitorea el SEN y reporta inmediatamente las fallas al grupo de mantenimiento y despacha en condiciones de emergencia la reserva disponible.

Para llevar a cabo la supervisión y coordinación en forma remota y de la operación del sistema nacional de potencia cuenta con un sistema de comunicación y procesamiento de datos que opera con alta tecnología.

1.2 Definición del problema y su importancia

El CENCE realiza el control de energía del país controlando y monitoreando las 48 subestaciones de control con las que cuenta, así como las plantas de generación de energía. Para ello mantiene comunicación las 24 horas del día con cada uno de estos sitios, por medio de la red de comunicaciones del ICE sector Electricidad. Por ello es de suma importancia garantizar la confiabilidad y continuidad de las comunicaciones en la red del ICELEC. De gran importancia para dicha red son los sitios de repetición, ya que estos son los que se encargan de retransmitir las comunicaciones entre el centro de control y las subestaciones, así como plantas generadoras. Si algunos de estos sitios de repetición dejan de funcionar, detiene el tráfico de comunicación entre el centro de control y una o más subestaciones.

En estos sitios de repetición se pueden presentar distintas situaciones, ya sea en el equipo allí presente o en las condiciones ambientales del inmueble, que se pueden definir como alarmas, ya que cuando se presentan pueden implicar fallas a corto o mediano plazo, y hasta en forma inmediata, en el funcionamiento de la estación repetidora.

Tales alarmas se pueden presentar de distintas fuentes: equipos de microondas, unidad de planta generadora de emergencia, el grupo electrógeno (planta generadora, tanque de combustible), iluminación de la torre, sistema de seguridad local, variación en los niveles de temperatura y/o humedad, puertas abiertas, equipo principal o de respaldo dañado. Además, desde el centro de control en La Sabana es importante conocer la potencia de recepción y transmisión de los sitios de repetición, así como el voltaje en las baterías, para determinar si el rango de operación es el adecuado.

Es de suma importancia para el centro nacional de control de energía, conocer en la forma más efectiva y rápida posible el estado de los sitios de repetición, y por supuesto sus alarmas, así como su prioridad. Algunas alarmas tienen mayor

importancia que otras, ya que pueden implicar el mal funcionamiento o la paralización de la retransmisión y por lo tanto de las comunicaciones en forma inmediata. Dependiendo del tipo y naturaleza de la alarma se corrige el problema desde el centro de control, o bien, se envía un equipo de mantenimiento al sitio de repetición para que repare o sustituya el equipo, o realice la tarea necesaria para eliminar tal alarma.

En la actualidad el CENCE afronta varios problemas debido a la poca información con la que cuenta de los sitios de repetición, ya que muchas veces descubre equipo dañado cuando algún grupo de técnicos realiza una visita de supervisión y mantenimiento, y en algunos casos, cuando la retransmisión no se realiza debido a que equipo principal y de respaldo ya se han dañado. Por ello es común descubrir equipo en mal estado hasta mucho tiempo después del momento en que este dejó de funcionar; si se conociera de antemano algunas condiciones del mismo, se podría evitar que los repetidores dejaran de funcionar por estos motivos, o al menos disminuir al máximo el tiempo en que estos dejan de funcionar.

Durante el tiempo que se mantenga fuera de funcionamiento, no existe comunicación entre el centro de control y las subestaciones o las plantas generadoras afectadas, lo que impide el trabajo de control y monitoreo de las mismas. En tales condiciones las subestaciones o las plantas podrían funcionar en condiciones no adecuadas o dejar de funcionar completamente, y en estos períodos no se produce consumo de energía por parte de los usuarios afectados, y por lo tanto se presenta una reducción en el ingreso del ICE en este campo; la repercusión económica que se presenta en tales casos varía dependiendo del sector afectado así como de la hora a la que se produce. También se puede incurrir en un gasto adicional al pagar salarios extraordinarios y viáticos a técnicos que brindan mantenimiento a los sitios de repetición, ya que el mal funcionamiento de los mismos se puede presentar en cualquier momento. Además, si las interrupciones se presentan continuamente, disminuye la calidad del servicio que brinda el ICE a todos los consumidores.

Indice

Por estas razones es que el Instituto Costarricense de Electricidad tiene un gran interés en desarrollar el proyecto que presente la solución a este problema, brindando todo el apoyo a la persona que se encargue del mismo.

El CENCE desea un dispositivo que se pueda colocar en cada sitio de repetición, conectado a señales digitales y analógicas, de las cuales se puede obtener la información requerida en el centro de control; cuando se presenten las alarmas el dispositivo deberá realizar los reportes correspondientes al centro nacional de control de energía.

1.3 Objetivos Alcanzados

1.3.1 Conocer todas las fuentes de alarmas de los sitios de repetición de la red de comunicaciones del ICELEC.

1.3.2 Especificar las señales y los tipos de señales que generan las alarmas que se deben monitorear en los sitios de repetición.

1.3.3 Especificar el microcontrolador que gobernará el sistema de monitoreo de alarmas y los demás componentes de hardware.

1.3.4 Diseñar una rutina para el microcontrolador maestro que se comunique por medio del protocolo Modbus.

1.3.5 Realizar pruebas a la rutina de comunicación Modbus del microcontrolador maestro.

1.3.6 Evaluar las pruebas de la rutina de comunicación Modbus del microcontrolador maestro.

1.3.7 Diseñar una rutina para el microcontrolador capaz de monitorear 16 fuentes de alarma digitales.

1.3.8 Realizar pruebas de la rutina de monitoreo de señales digitales.

1.3.9 Evaluar las pruebas a la rutina de monitoreo de señales digitales.

1.3.10 Diseñar una rutina para el microcontrolador esclavo para que se comunique por medio del protocolo Modbus.

1.3.11 Realizar pruebas de la comunicación Modbus del microcontrolador esclavo.

1.3.12 Evaluar las pruebas de comunicación Modbus del microcontrolador esclavo.

1.3.13 Diseñar un programa para el microcontrolador esclavo capaz de monitorear 3 señales analógicas.

Indice

1.3.14 Realizar pruebas de la rutina de monitoreo de señales analógicas.

1.3.15 Evaluar pruebas de la rutina de monitoreo de señales analógicas.

1.3.16 Diseñar rutinas para ambos microcontroladores para que se comuniquen entre sí por medio del protocolo I^2C.

1.3.17 Realizar pruebas a la rutina de comunicación entre microcontroladores por medio del protocolo I^2C.

1.3.18 Evaluar pruebas de la rutina de comunicación entre microcontroladores por medio del protocolo I^2C.

1.3.19 Realizar pruebas de todas las rutinas de los microcontroladores con todos los componentes de hardware del prototipo.

1.3.20 Demostrar que el dispositivo funciona en forma correcta de acuerdo a los requerimientos establecidos.

1.3.21 Preparar la documentación para la transferencia del conocimiento tecnológico obtenido durante la realización del proyecto.

CAPÍTULO 2
ANTECEDENTES

2.1 Estudio del problema a resolver

Desde la apertura de más cerros de repetición cuatro años atrás, se han presentado serias dificultades en la atención de fallas o averías en los sitios de repetición de la red de comunicaciones del ICELEC, ya que dichos sitios son desatendidos y completamente autónomos.

En los sitios de repetición se presentan fallas en el equipo de comunicaciones, fallas en la planta generadora de emergencia, condiciones adversas en el inmueble (alta temperatura, humedad, fuego, etc), y hasta intrusiones no permitidas que terminan en vandalismo. Todas estas situaciones resultan de alta importancia para el conocimiento del personal del Centro de Control de Energía, y hoy día se puede tener conocimiento a nivel general, de un número muy limitado de alarmas.

La especificación detallada y más amplia de las alarmas facilita el trabajo para técnicos e ingenieros en la atención de averías, ya que cuando se realiza la visita al sitio se tiene conocimiento del origen de la falla; además, como el número de sitios de repetición ha aumentado, se hace necesario reducir al máximo el número de visitas de los técnicos a tales sitios.

Indice

2.2 Requerimientos de la empresa

La principal necesidad del CENCE, es conocer en forma rápida y precisa el estado de los sitios de repetición y del equipo presente en los mismos, es decir, tener a mano las advertencias, fallas y valores de los equipos en tales sitios.

Ya que no todas las fuentes de alarmas son comunes en los sitios de repetición, y aunque lo fueran, no siempre están activas en todos los sitios, es conveniente poder activar y desactivar alarmas en forma remota (desde el CENCE). La tarjeta electrónica en los sitios, monitoreará únicamente las alarmas que estén activas.

Cuando se genera una alarma, es importante tener la opción de reconocerla desde el Centro de Control, este reconocimiento le dice a la tarjeta en el sitio de repetición, que el sistema en el CENCE ya tiene conocimiento de dicha alarma, por lo que no es necesario reportarla nuevamente.

Para cada señal analógica existirán dos límites superiores y dos límites inferiores; cuando la señal se encuentre entre los dos primeros límites (superiores e inferiores) el valor de la señal es encuentra en un rango aceptable. Cuando sea menor al primer límite inferior se presenta una alarma no urgente, cuando es menor que el segundo límite inferior se presenta una alarma urgente. Así mismo, cuando sobrepasa el primer límite superior se presenta una alarma no urgente, y cuando sobrepasa el segundo límite superior se presenta una alarma urgente.

Además es necesario monitorear el estado y valor de los parámetros de la planta de emergencia del sitio. Dicha planta presenta la opción comunicar todos los datos correspondientes a la misma por medio de protocolo Modbus.

En cuanto al protocolo lógico de transmisión de los datos en forma remota, se requiere que sea confiable, y muy conocido para lograr una alta compatibilidad con otros sistemas.

2.3 Solución propuesta

La solución para dicho problema se basó en la construcción de un dispositivo que utilice como núcleo dos microcontroladores. Se construyó un prototipo en una protoboard, el cual se conecta a 16 terminales discretas (contactos secos), 3 señales analógicas, y un dispositivo local que utilice comunicación por protocolo Modbus (planta generadora de emergencia).

Monitoreando y procesando estas señales se determina si existe alguna alarma, la cual debe ser reportada. La información dirigida al Centro de Control sale del dispositivo a través de un puerto serie (protocolo RS-232) el cual se conecta localmente a una interface V24, que transmitirá la información por radio, microondas o fibra óptica, además permite conectar el dispositivo con cualquier computadora personal en caso de ser necesario. El protocolo lógico utilizado es Modbus.

A continuación se presenta un diagrama de bloques general (figura 2.1), del sistema que se desea.

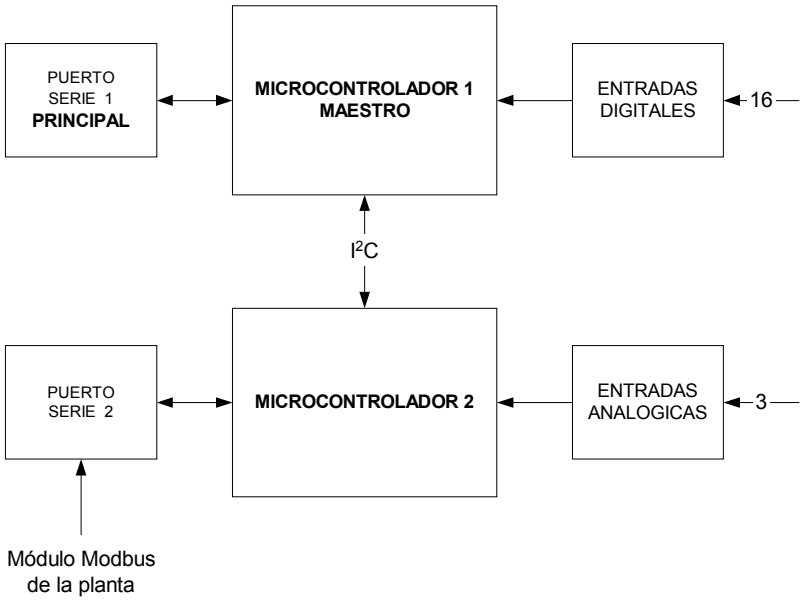


Figura 2.1 Diagrama de bloques del dispositivo deseado.

Los bloques de señales digitales y analógicas que se muestran en la figura 2.1, deben tomarse en cuenta ya que existe la opción de expandir el número de señales digitales y analógicas que se monitorean para futuras aplicaciones, ya que es posible que la cantidad de fuentes de alarmas aumenten. En el caso de las señales analógicas, se establece el tipo de señal como de tensión, los valores que determinan los umbrales son variables, y controlados por el sistema principal en el Centro de Control.

El microcontrolador 2 (Esclavo en figura 2.1), monitorea continuamente las señales analógicas conectadas al mismo, y determina si existe alguna alarma. Por medio del bloque RS-232 (Puerto Serie 2, en figura 2.1), y sí existe algún dispositivo conectado, chequea advertencias, fallas y valores en dicho dispositivo. Por medio del protocolo I^2C se comunica con el microcontrolador maestro, brindándole la información de las señales analógicas y de la planta generadora de emergencia.

El microcontrolador 1 (Maestro en figura 2.1), monitorea las 16 señales digitales de la tarjeta, pregunta al esclavo por las correspondientes alarmas analógicas y de la planta generadora. A través del Puerto Serie 1, responde a los paquetes Modbus que le envía el sistema principal en el CENCE, y si dichos paquetes son para la planta generadora, los redirecciona a la misma.

Se maneja el reconocimiento de las alarmas desde el Centro de Control, para evitar la transmisión de información redundante. Así como la habilitación y e inhibición de alarmas desde el Centro de Control, para que la tarjeta en el sitio de repetición no monitoree alarmas que no se encuentran activas.

Indice

CAPÍTULO 3
PROCEDIMIENTO METODOLÓGICO REALIZADO

3.1 Conocimiento de las fuentes de alarmas de los sitios de repetición de la red de comunicaciones.

3.1.1 Se realizó una visita a una estación de repetición con el fin de observar y conocer las características del equipo presente en la misma. (2 día)

3.1.2 Entrevistas al personal del CENCE relacionada con la operación de los sitios de repetición. (3 día)

3.2 Especificación de las señales, y los tipos de señales que generan alarmas.

3.2.1 Consultar manuales de usuario del equipo que generará alarmas. (3 días)

3.2.2 Crear lista de las señales, tipos de señales, características de las mismas y fuente de la señal. (1 día)

3.3 Especificación del microcontrolador y de los demás componentes de hardware del sistema.

3.3.1 Determinación de las principales características del microcontrolador que se adecuan al proyecto. (1 día)

3.3.2 Consulta en hojas de datos, de software y hardware, de los principales fabricantes de microcontroladores. (2 días)

3.3.3 Escogencia del microcontrolador adecuado al proyecto, de bajo costo y con las características requeridas. (1día)

3.3.4 Se realizó lista definitiva de componentes a utilizar, considerando las características del equipo analizado y los requerimientos del proyecto. (2 día)

Indice

3.3.5 Búsqueda de los componentes en los principales distribuidores nacionales e internacionales para escoger los lugares donde se realizará la compra. (2 días)

3.3.6 Se realizó compra definitiva de componentes. (1 día)

3.3.7 Tiempo de entrega de componentes. (40 días)

3.4 Diseño de una rutina para el microcontrolador maestro que se comunique por protocolo Modbus.

3.4.1 Consulta de manuales de hardware y software del microcontrolador. (12 días)

3.4.2 Consulta de documentación del protocolo Modbus. (5 días)

3.4.3 Programación de rutina para comunicación Modbus del microcontrolador maestro. (6 días)

3.5 Realización de pruebas a la rutina de comunicación Modbus del microcontrolador maestro.

3.5.1 Descarga del programa diseñado en el microcontrolador. (3 días)

3.5.2 Simulación de entradas al microcontrolador y verificación que el microcontrolador transmite y recibe a través del puerto serie los paquetes requeridos. (3 días)

3.6 Evaluación de las pruebas de la rutina de comunicación Modbus del microcontrolador maestro.

Se verificó que se diera una adecuada respuesta del microcontrolador con respecto a los comandos o paquetes Modbus que se le ingresan. (3 días)

3.7 Diseño de una rutina para el microcontrolador que monitoree 16 fuentes de alarma digitales.

3.7.1 Programación de rutina que controle y monitoree los puertos de entrada salida. (1 día)

Indice

3.7.2 Programación de rutina que capture y compare los valores de los puertos de entrada salida. (2 día)

3.7.3 Se agruparon las rutinas diseñadas. (1 días)

3.8 Realizar pruebas de la rutina de monitoreo de señales digitales.

3.8.1 Descarga del programa diseñado para monitorear señales digitales. (2 días)

3.8.2 Simulación de entradas al microcontrolador y verificación de que el microcontrolador capture la información correspondiente. (3 días)

3.9 Evaluación de las pruebas a la rutina de monitoreo de señales digitales.

Se verificó la información que el microcontrolador almacena, con respecto a las señales de entrada simuladas. (2 días)

3.10 Diseño de una rutina para el microcontrolador esclavo para que se comunique por medio del protocolo Modbus.

3.10.1 Estudio de rutina realizada en el punto 4.3 de esta metodología. (2 día)

3.10.2 Programación de rutina que comunique por medio de protocolo Modbus al microcontrolador esclavo. (3 día)

3.11 Realizar pruebas de la comunicación Modbus del microcontrolador esclavo.

3.11.1 Descarga del programa diseñado para comunicación Modbus del esclavo. (2 días)

3.11.2 Simulación de una entrada Modbus al microcontrolador y observar los paquetes que este genera. (1 días)

3.12 Evaluación de las pruebas de comunicación Modbus del microcontrolador esclavo.

Se verificó que se de una adecuada respuesta del microcontrolador con respecto a los comandos o paquetes Modbus que se le ingresan. (1 día)

3.13 Diseño de un programa para el microcontrolador esclavo que monitoree las 3 señales analógicas.

3.13.1 Programación de rutina que controle y almacene la conversión analógica-digital de 3 señales analógicas en el microcontrolador. (1 día)

3.13.2 Programación de rutina que maneje y verifique las magnitudes de la conversión analógica digital. (1 día)

3.13.3 Se agruparon las rutinas diseñadas. (2 días)

3.14 Realizar pruebas de la rutina de monitoreo de señales analógicas.

3.14.1 Descarga del programa diseñado para monitorear las señales analógicas. (2 días)

3.14.2 Simulación de entradas analógicas al microcontrolador y verificación de que el microcontrolador capture y procese adecuadamente la información correspondiente. (1 día)

3.15 Evaluación de pruebas de la rutina de monitoreo de señales analógicas.

Se verificó que la información que el microcontrolador almacena se correcta, con respecto a las señales analógicas de entrada simuladas. (2 días)

3.16 Diseño de rutinas para ambos microcontroladores para que se comuniquen entre si por medio del protocolo I^2C.

3.16.1 Consulta de documentación técnica del protocolo de comunicación serial I^2C. (3 días)

3.16.2 Programación de rutina para comunicación por I^2C para el microcontrolador maestro. (6 días)

3.16.3 Programación de rutina para comunicación por I^2C para el microcontrolador esclavo. (5 días)

3.17 Realización de pruebas a la rutina de comunicación entre microcontroladores por medio del protocolo I^2C.

Indice

3.17.1 Descarga de los programas en los microcontroladores para la comunicación por I^2C. (2 días)

3.17.2 Simulación de entradas a un microcontrolador, y verificar que el microcontrolador transmite y recibe por I^2C información correspondiente. (3 días)

3.18 Evaluación de pruebas de la rutina de comunicación entre microcontroladores por medio del protocolo I^2C.

Se verificó que la información transmitida de un microcontrolador a otro por medio de I^2C sea la correcta, de acuerdo a las entradas simuladas. (3 días)

3.19 Realizar pruebas de todas las rutinas de los microcontroladores con todos los componentes de hardware del prototipo.

3.19.1 Se agruparon en un solo programa las diferentes rutinas desarrolladas para el microcontrolador maestro. (2 días)

3.19.2 Se agruparon en un solo programa las diferentes rutinas desarrolladas para el microcontrolador esclavo. (2 días)

3.19.3 Se montó en la protoboard todos los componentes del prototipo. (1 días)

3.19.4 Descarga en los microcontroladores de los programas diseñados con todas las rutinas agrupadas. (1 días)

3.20 Evaluación de las pruebas realizadas al prototipo.

Se simularon alarmas en las entradas y se verificó que se transmita a través del puerto serie y por medio del protocolo Modbus la información correspondiente. (2 días)

3.21 Demostración de que el dispositivo funciona en forma correcta de acuerdo a los requerimientos establecidos.

3.21.1 Verificar que los paquetes Modbus enviados correspondientes a las señales de entrada simuladas sean los correctos. (1 días)

Indice

3.21.2 Verificar que el prototipo se comporta correctamente con respecto a la información que se envía por protocolo Modbus. (1 días)

3.21.3 Determinar si el funcionamiento del sistema responde a los requerimientos establecidos. (1 día)

3.21.4 Realizar las correcciones necesarias al hardware y software del prototipo en el microcontrolador, después del análisis del funcionamiento del sistema. (1 días)

3.22 Preparación de documento para la empresa, donde se transferirá el conocimiento tecnológico obtenido y generado en la realización del proyecto. Así como la preparación de la presentación final del proyecto en la empresa. (4 días)

Indice

CAPÍTULO 4
DESCRIPCIÓN DEL HARDWARE UTILIZADO

En la figura 2.1 se presentó el diagrama de bloques general del hardware que se utiliza, los principales elementos son los microcontroladores de la familia de Microchip®, PIC18C252. Estos microcontroladores cuentan con 16Kbytes de memoria de programa (EPROM), para un total de 8192 instrucciones, y 1536 bytes de memoria de datos (RAM). Su arquitectura es optimizada para la utilización de un compilador del lenguaje C. Tiene 22 pines de entrada-salida, algunos de los cuales son utilizados por algunos módulos periféricos. Cuenta adicionalmente con 4 timers/counters de 16 bits, un módulo USART, un módulo convertidor analógico digital (ADC) de 10 bits, y un módulo de puerto serie síncrono (MSSP), que se puede implementar en dos modos: *3-wire SPI* ó I^2C™. Además, permite el manejo de interrupciones por medio de prioridades. El las figuras B1.1 y B1.2 se presentan las características más importantes de dicho microcontrolador, así como la distribución de pines del mismo.

Se utilizó un MAX233 para realizar la conversión entre la señal serial digital del microcontrolador a una señal RS-232; las características más importantes, así como la distribución de pines del mismo se presentan en las figuras B1.4 y B1.5. Este circuito integrado es conectado a ambos microcontroladores, ya que el mismo puede manejar dos puertos series independientes.

Para mejorar el acople de impedancias a las señales analógicas con los pines del PIC, se utilizan 3 amplificadores operacionales, conectados como seguidores de tensión. De esta forma la señal analógica correspondiente a una alarma o transductor en el sitio, se conecta a una resistencia de entrada muy alta, pero las entradas analógicas en el microcontrolador tienen una resistencia muy baja conectada a ellas.

Indice

Otro elemento muy importante es el programador de los microcontroladores, ya que por medio del mismo se descargan las rutinas diseñadas. El programador utilizado es exclusivo en el uso de microcontroladores de la familia Microchip®, se utiliza el PICSTART PLUS® v2.30. La descarga de las rutinas en los PIC's se realiza en forma serie por medio de una PC.

En el microcontrolador maestro los 8 bits del puerto B son utilizados como entradas digitales, estas ocho líneas funcionan como un bus interno en la tarjeta, ya que se conectan a la salida de los buffers de tercer estado. Los buffers que se utilizan son un par de 74LS244, que se sirven de acople entre las señales digitales correspondientes a alarmas, y los pines de entrada del microcontrolador. El diagrama de distribución de pines, así como las principales características de este circuito integrado se presentan en la figura B1.3. Y para controlar la habilitación de los buffers se establecen los pines 0 y 1 del puerto A como salidas digitales. Los pines 6 y 7 del puerto C se deben configurar como entradas digitales, ya que en ellos se encuentran las señales Tx y Rx de la USART del PIC. En este mismo puerto pero en los pines 3 y 4 se encuentran las líneas SCL y SDA, que son las señales de reloj y datos en el protocolo I^2C, y por ello deben ser configuradas como entradas digitales.

En el microcontrolador esclavo se configura el puerto A con 5 entradas analógicas y una entrada de referencia. Los pines 0, 1, 2, 4 y 5 son entradas analógicas, y el pin 3 funciona como la entrada de tensión de referencia para el convertidor analógico-digital. Al igual que en el microcontrolador maestro los pines 3, 4, 6 y 7 corresponde a las señales SCL, SDA, Tx y Rx del protocolo I^2C y de la USART respectivamente, por ello deben ser configurados con entradas digitales.

Ambos microcontroladores utilizan un oscilador de 10 MHz que le brinda la señal de reloj a cada uno. Por medio del módulo PLL con que cuenta cada PIC, se multiplica la frecuencia internamente por 4, y es por ello que la frecuencia interna de cada microcontrolador es de 40 MHz. En este caso se utilizaron dos osciladores TTL de 10 MHz, pero es conveniente cambiar el uso del mismo a un oscilador paralelo de

10 MHz. En la figura 4.1 se muestra la forma de conectar ambos tipos de osciladores a las terminales OSC1 y OSC2.

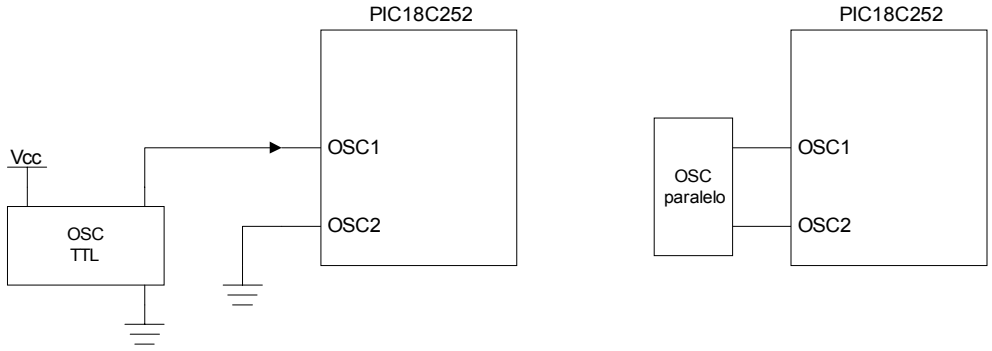


Figura 4.1 Diagrama de conexión de osciladores paralelos y osciladores TTL al PIC18C252.

Para la interconexión entre ambos microcontroladores a través del bus I^2C, es necesario conectar una resistencia de pull-up entre la correspondiente línea y la alimentación, ya que los pines de entrada en modo I^2C son de colector abierto.

Indice

CAPÍTULO 5
DESCRIPCIÓN DEL SOFTWARE DEL SISTEMA

El software que se utilizó para la programación de los microcontroladores es el MPLAB® IDE, el cual en conjunto con el PICSTAR PLUS permiten descargar las rutinas diseñadas, en los PIC's.

MPLAB® IDE es un sistema de desarrollo completo, ya que en el mismo se puede editar código y realizar simulaciones. Las rutinas que se implementen para los PIC's pueden desarrollarse en un lenguaje básico con las 76 instrucciones de los microcontroladores de Microchip®, este lenguaje básico es un seudo ensamblador; o bien, se pueden desarrollar rutinas a un nivel más alto por medio de lenguaje C.

Para compilar rutinas en seudo-ensamblador se utiliza el compilador MPASM, incluido en el MPLAB® IDE. Sin embargo, en el desarrollo del proyecto se utilizó un demo del compilador del lenguaje C, el MPLAB® C18; esto para sacar provecho de las facilidades en el desarrollo de estructuras y algoritmos más complejos que brinda dicho lenguaje.

Una vez compilado un archivo correspondiente a un programa para el PIC, se genera un archivo con extensión .hex, el cual se carga en la memoria de programa del microcontrolador.

Como el modelo de hardware utiliza dos microcontroladores, en la parte lógica o de software se realiza una división de tareas entre los mismos. Se hace una denominación de microcontrolador esclavo y microcontrolador maestro de acuerdo al protocolo I^2C, ya durante la comunicación entre ambos PIC's uno de ellos es maestro y el otro esclavo.

El microcontrolador esclavo debe realizar el monitoreo de las señales analógicas, para ello el programa del PIC debe realizar las conversiones de

analógico a digital y comparar los valores obtenidos con los umbrales que representan alarmas en cada señal. Además, debe monitorear continuamente la planta de emergencia del sitio, en caso de existir una, y determinar la existencia de una alarma en ella. Cuando el sistema principal en el CENCE envía un paquete a la planta, el PIC esclavo debe reenviar el paquete a la planta, y esperar la respuesta de la misma para enviarla al Centro de Control. Por último, en el PIC esclavo deben estar las rutinas que le permitan al mismo comportarse como un esclavo en el protocolo I^2C, siempre que el maestro inicie transacciones.

Por otro parte, el programa del microcontrolador maestro debe monitorear las señales digitales, y determinar bit por bit cuando se presenta una alarma. Debe preguntar al PIC esclavo por las alarmas analógicas que este monitorea, y de acuerdo a los valores que recibe del esclavo, actualiza la información en sus registros internos. También tiene que enviar al PIC esclavo la información de cuales alarmas analógicas debe monitorear, y cuales son los rangos de operación de las mismas. Este microcontrolador es el que tiene acceso al puerto serie principal, por ello debe comunicarse por medio del protocolo Modbus con el Centre de Control. Cuando reciba paquetes Modbus dirigidos a la planta de emergencia, debe reenviar el paquete al PIC esclavo por medio del protocolo I^2C, para que el esclavo lo envíe a la planta, luego debe monitorear el esclavo para determinar cuando se recibió la respuesta de la planta, y de esta forma enviarla al CENCE.

Todas las rutinas desarrolladas en el los microcontroladores fueron probadas físicamente en el prototipo, y también se utilizaron pequeños programas en una PC que facilitaron dichas pruebas. Todas las rutinas que se realizaron en la PC fueron desarrolladas por medio del lenguaje Delphi 5.

Indice

CAPÍTULO 6
ANÁLISIS Y RESULTADOS

6.1 Explicación del diseño

En la figura 6.1 se presenta un diagrama de entradas y salidas de la tarjeta electrónica. Como entradas se tienen las 3 señales analógicas y las 16 señales digitales que se deben monitorear, además, por medio de un puerto serie se debe chequear el estado de la planta de emergencia del sitio, la cual brinda información por medio del protocolo Modbus.

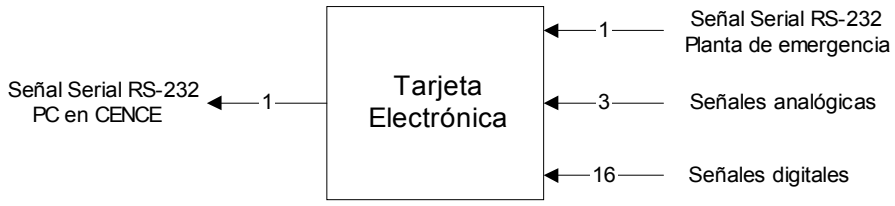


Figura 6.1 Diagrama de entradas/salidas de la tarjeta electrónica.

La única salida del sistema es el puerto serie con la señal RS-232, por la cual se transmite información al Centro de Control de Energía por medio del protocolo Modbus.

En la figura 2.1 se presentó un diagrama de bloques del hardware básico de la tarjeta electrónica. En dicha tarjeta los módulos RS-232 (Puerto serie 1 y 2) se desarrollan con un manejador de señal serial TTL a RS-232, MAX233; el cual puede manejar dos puertos series independientes. Por medio de conectores DB9 se conecta la tarjeta a la interfaz V24 y a la planta de emergencia (en caso de existir una).

Los microcontroladores utilizados para el maestro y el esclavo, son PIC18C252 de la familia Microchip®.

Indice

El bloque de entradas analógicas cuenta con 3 amplificadores operacionales conectados como seguidores de voltaje, para realizar un adecuado acople de impedancias entre la fuente de la señal y la entrada en el PIC. Por medio del convertidor analógico-digital interno del microcontrolador y sus 3 canales, se realiza la conversión respectiva.

El bloque de entradas digitales cuenta con 2 registros 74LS244, los cuales son buffers con salida en tercer estado (3State). Resistencias de pull-up de 10KΩ conectadas entre V_{CC} y la entrada de tipo contacto seco, se utilizaron para generar las señales de 0 y 5 voltios en las entradas de los buffers 74LS244, correspondientes a las alarmas digitales. El microcontrolador habilitará un buffer a la vez para realizar el correspondiente monitoreo de las señales digitales. De tal forma que las entradas digitales se manejan a través de un bus de 8 bits, esto para brindar la posibilidad de aumentar en un futuro el número de señales a monitorear.

La comunicación entre Maestro y Esclavo, se realizará por medio del protocolo I^2C, el cual físicamente utiliza 3 líneas: datos, reloj y tierra.

A continuación, en la figura 6.2 se presenta un diagrama de bloques más detallado del hardware de la tarjeta electrónica, según lo explicado anteriormente.

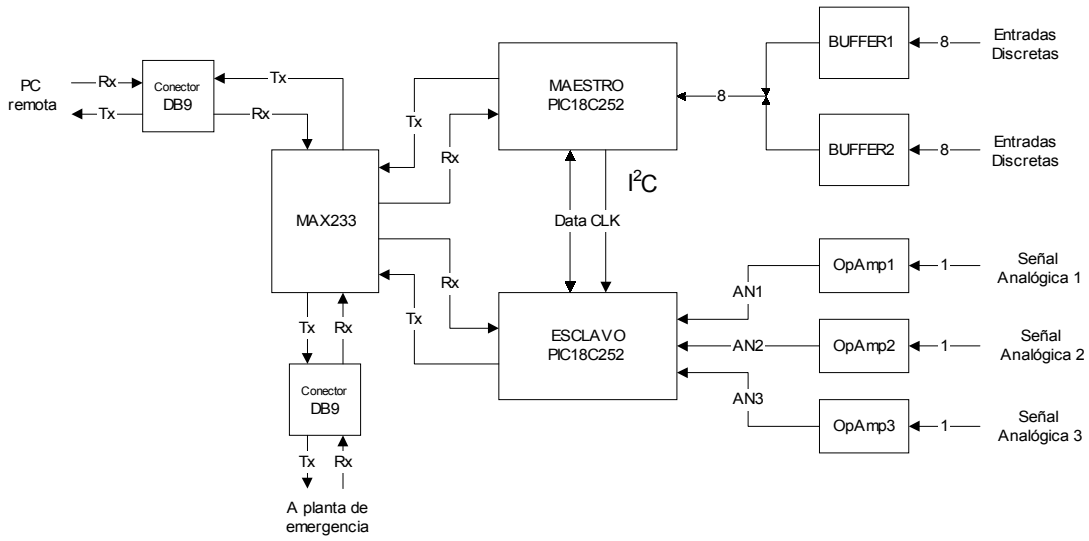


Figura 6.2 Diagrama de bloques del hardware de la tarjeta electrónica.

Indice

La tarjeta electrónica es controlada por los microcontroladores, cada uno de ellos cuenta con un programa que se encarga de realizar las tareas que le corresponden, tanto al maestro como al esclavo. Ambos PIC's utilizan el protocolo Modbus, el maestro para comunicarse con el Centro de Control, y el esclavo para comunicarse con la planta, además, la comunicación entre PIC's se da por medio del protocolo I^2C. En los siguientes apartados se presenta una descripción general del protocolo Modbus y protocolo I^2C, los cuales ayudarán a entender el desarrollo de las rutinas en los microcontroladores. Además, el programa desarrollado en lenguaje C para cada microcontrolador se analiza en apartados distintos, para facilitar la comprensión de cada rutina.

6.1.1 Protocolo Modbus

Este protocolo define una estructura de mensaje que puede trabajar sobre diversos tipos de redes. Los puertos Modbus estándar son interfaces seriales compatibles con RS-232. Los dispositivos en una red Modbus se comunican usando una técnica de maestro-esclavo, en donde solo el maestro puede iniciar las transacciones (llamadas ''queries''). Por cada ''query'' que el maestro envíe, el esclavo, después de realizar la acción pedida, fabrica un paquete Modbus de respuesta y se lo envía al maestro para confirmar la acción.

Cada ''query'' tiene asociado una dirección y un código de función, que se utilizan para direccionar el esclavo, y le indican al mismo que acción debe tomar; además, se incluirán bytes de datos conteniendo información adicional para que el esclavo realice la función deseada. Existirá un campo con el chequeo de error, que servirá para validar la integridad del contenido del mensaje. El paquete de respuesta, si es normal, contiene los datos pedidos por el maestro, o bien, una replica del ''query'', con su respectivo campo de chequeo de error. Si ocurre un error, el código de función es modificado, para indicar que la respuesta es de error.

Existen dos modos de transmisión: ASCII y RTU. Para el desarrollo de este proyecto se utiliza el segundo. En modo RTU (remote terminal unit) cada mensaje debe ser transmitido continuamente, y se codifican dos caracteres hexadecimales en cada campo de 8 bits del mensaje. Cada campo del mensaje contiene: 1 bit de inicio, 8 bits de datos, 1 bit de paridad (en caso de ser utilizada), 1 bit de parada (o 2 bits de parada si no se utiliza paridad).

Cada mensaje en Modbus RTU inicia con un intervalo de silencio correspondiente al tiempo de transmisión de 3.5 caracteres, luego se transmite la dirección del esclavo que se direcciona, y el resto del paquete. Cada dispositivo esclavo monitorea la red continuamente, cuando el primer byte es recibido se decodifica el valor para determinar si corresponde a la dirección del esclavo, si no es así, se espera hasta el inicio del siguiente paquete. Después del último carácter transmitido, un intervalo similar de 3.5 caracteres debe seguir al mensaje. Cada paquete debe ser transmitido continuamente, ya que si un intervalo de silencio de más de 1.5 caracteres ocurre antes de que se halla completado el mensaje, el dispositivo receptor elimina el mensaje incompleto y asume que el próximo byte corresponde a un nuevo mensaje.

En Modbus RTU el chequeo del error se realiza con el método Cyclical Redundancy Check (CRC), el cual es aplicado a todo el mensaje. El campo del CRC es de 2 bytes, conteniendo un valor binario de 16 bits, la parte baja del CRC es enviada primero y la parte alta después. El valor CRC es calculado por el dispositivo transmisor y colocado al final del mensaje; el dispositivo receptor recalcula el CRC y lo compara con el valor recibido en el campo de CRC para determinar posibles errores. El algoritmo para calcular el CRC es el siguiente:

a. Un registro de 16 bits (Reg_CRC) es precargado con 1’s.

b. Se aplica la función XOR al primer carácter de 8 bits del mensaje y el byte bajo del registro de contenido (Reg_CRC), dejando el resulta en el Reg_CRC.

c. El registro de contenido es desplazado en la dirección del bit menos significativo (LSB), rellenando con cero el bit más significativo (MSB).

d. Si el LSB extraído es 1, al registro de contenido se le aplica una XOR con la constante polinomial A001 hex;

   Si el LSB extraído es cero no se aplica la XOR.

e. Se repiten los pasos c y d hasta que se hallan realizado 8 desplazamientos.

f. Se repite del paso b al e para el próximo byte de 8 bits del mensaje; se continua haciendo esto hasta que todos los bytes del paquete hallan sido procesados.

g. El valor del chequeo de error CRC se encuentra en el registro Reg_CRC.

La figura 6.3 presenta el diagrama de flujo del algoritmo de cálculo del CRC.

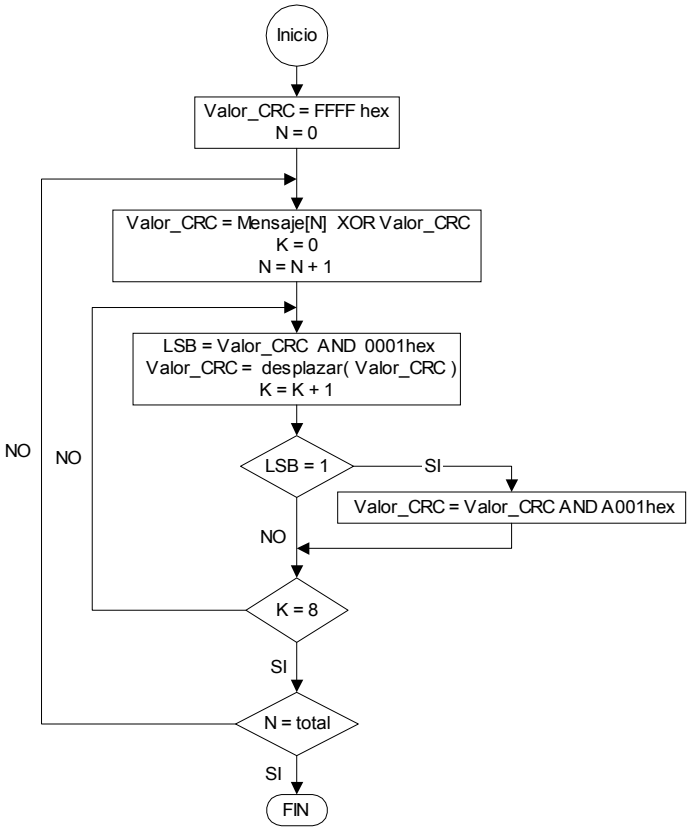


Figura 6.3 Diagrama de flujo del algoritmo de cálculo del CRC.

El contenido del paquete Modbus depende de la tarea que se desea realizar en el esclavo, la cual está determinada por la función Modbus incluida en el mismo. Para el desarrollo de este proyecto se implementaron dos funciones: la función 03 y la función 16.

La función 03 lee el contenido binario de registros en el esclavo, los cuales contienen la información que le interesa al maestro. El ‘’query’’ especifica el registro inicial y la cantidad de registros a leer. Los registros 1-16 son direccionados como 0-15. En la figura 6.4 se presenta la forma del paquete.

Dirección	Función	Dir Inicial Hi	Dir Inicial Lo	# Regs Hi	# Regs Lo	CRC Lo	CRC Hi

Figura 6.4 Paquete de Petición Modbus para la función 03.

En la figura anterior, cada bloque corresponde a un byte (2 caracteres Hexadecimales), *Dirección* se refiere a la dirección del esclavo (a quien se dirige la petición). *Función* corresponde a la función Modbus del paquete, en este caso la 03. La dirección inicial de los registros que se desean leer ocupa 2 bytes, y se divide enviando la parte alta primero y la parte baja después. El número de registros (*# Regs*) que se desean leer (a partir de la dirección inicial) ocupa 2 bytes, y se divide enviando la parte alta primero y la parte baja después. Por último se encuentra el valor del chequeo del error del paquete.

La respuesta típica para la función 03 es como la que se muestra en la figura 6.5.

Dirección	Función	Byte Count	n Data	CRC Lo	CRC Hi

Figura 6.5 Paquete de respuesta Modbus para la función 03.

La respuesta a la petición 03, devuelve la dirección del esclavo para que el maestro conozca el origen de la respuesta, y además, devuelve el mismo código de función. En el campo *Byte Count* se transmite el número de bytes que contiene el paquete, correspondientes a datos de los registros que se leyeron en el esclavo. *n*

Data representa los bytes con el contenido de los registros que se leyeron en el esclavo, cada registro es de 2 bytes, y se envía la parte alta del registro primero y la parte baja después.

La otra función que se implementa, la función 16, escribe en los registros Modbus del esclavo, sobrescribiendo el valor correspondiente. Los valores en estos registros permanecen invariables hasta que la lógica del programa los altere. El ‘’query’’ de la función 16 es como se muestra en al figura 6.6.

Dirección	Función	Dir Inicial Hi	Dir Inicial Lo	# Regs Hi	# Regs Lo	Byte Count	n Data	CRC Lo	CRC Hi

Figura 6.6 Paquete de Petición Modbus para la función 16.

En el ‘’query’’ de la función 16 se incluye la dirección del esclavo, y la función Modbus, en este caso 16 (10 hex). La dirección inicial de los registros que se desean modificar en el esclavo es de 2 bytes, se envía la parte alta primero y la parte baja después. El número de registros a modificar (*# Regs*) también es de 2 bytes, se envía la parte alta primero y la parte baja después). *Byte Count* contiene el número de bytes incluidos en el paquete, que corresponden a los datos que se sobrescribirán en los registros del esclavo. Y por último se incluye el campo del CRC. La respuesta a la función 16 tiene la forma que se presenta en la figura 6.7.

Dirección	Función	Dir Inicial Hi	Dir Inicial Lo	# Regs Hi	# Regs Lo	CRC Lo	CRC Hi

Figura 6.7 Paquete de respuesta Modbus para la función 16.

En la respuesta se devuelven todos los datos enviados en el ‘’query’’, menos el *Byte Coun*t y *n Data*, con el propósito de que el maestro sepa que la acción se llevó a cabo en forma exitosa.

Cuando un dispositivo maestro envía un ‘’query’’ a un dispositivo esclavo, este espera una respuesta normal; pero realmente se pueden presentar una de cuatro situaciones, como se describe a continuación:

1) Si el dispositivo esclavo recibe la petición sin error de comunicación y puede manejarla, envía una respuesta normal.

2) Si el dispositivo esclavo no recibe la petición debido a un error de comunicación, ninguna respuesta es retornada. El programa maestro eventualmente procesará una condición de time out para el ‘’query’’.

3) Si el esclavo recibe el ‘’query’’, pero detecta un error de comunicación (paridad o CRC), ninguna respuesta es retornada. El programa maestro eventualmente procesará una condición de time out para el ‘’query’’.

4) Si el esclavo recibe petición sin error de comunicación, pero no puede manejarla, el esclavo retorna una respuesta de excepción, informando al maestro la naturaleza del error.

El paquete Modbus de una respuesta de excepción, cuenta con 4 campos tal como se muestra en la figura 6.8.

Dirección	Función	Exception Code	CRC Lo	CRC Hi

Figura 6.8 Paquete de respuesta de excepción Modbus.

En la figura anterior, la *dirección* corresponde a la dirección del esclavo que generó la excepción; la *función* contiene el valor de la función que el maestro pidió al esclavo, es decir, la función en el paquete de petición, pero con un 1 en el bit más significativo, indicando que se trata de una excepción. En el campo *Exception Code* se envía un valor que le dice al maestro que condición en el esclavo causó la excepción. Y finalmente se agrega el valor del CRC.

Los códigos de excepción del protocolo Modbus son los que se muestran en la tabla 6.1.

Indice

Tabla 6.1 Códigos de excepción Modbus.

Código	Nombre	Descripción
01	Función ilegal	El código de función recibido no es una acción posible en el dispositivo esclavo.
02	Dirección de datos ilegal	La dirección de datos recibida en el ‘’query’’ no es una dirección permitida en el esclavo.
03	Valor de dato ilegal	El valor de datos contenido en la petición no es permitido para el esclavo.
04	Falla en el dispositivo esclavo	Un error en el dispositivo esclavo ocurrió mientras intentaba ejecutar la acción pedida.
05	ACKNOWLEDGE	El dispositivo esclavo recibió la petición y la está procesando, pero un período largo de tiempo se requiere para ejecutarse.
06	Dispositivo esclavo ocupado	El dispositivo esclavo esta procesando un comando de programa de larga duración.

Las funciones 03 y 16 que se implementaron realizan la lectura y escritura de los registros Modbus de la tarjeta electrónica, el contenido de dichos registros tiene la información correspondiente a las alarmas monitoreadas por la tarjeta electrónica, a este nivel no interesa como se procesa dicha información internamente, únicamente importan los datos contenidos en tales registros.

Por ello se implementó un bloque de memoria donde la tarjeta electrónica almacena la información que genera durante el monitoreo y procesamiento de las entradas del sistema. Ya que es el microcontrolador maestro el que tiene comunicación Modbus con el sistema principal, es en el mismo donde se implementan los registros con dicha información.

En el apéndice A.2 se presentan todos registros Modbus incluidos en dicho bloque de memoria, así como una descripción de la información que contiene cada registro. Los registros del 1 al 16 (en las tablas A2.1, A.2.2 y A2.3) contienen

información que utiliza el microcontrolador para el procesamiento de las alarmas: alarmas habilitadas, alarmas reconocidas, alarmas normalmente en bajo o alto, valores de las señales analógicas que determinan rangos de operación, etc. Los registros del 17 al 22 (tablas A.2.3 y A.2.4) contienen la información generada después del procesamiento de las alarmas: alarmas activas, estado de la tarjeta electrónica, y valores de las señales analógicas. En los registros del 23 a 50 (tabla A.2.4) se encuentran el stack o pila de alarmas y el stack o pila de contadores de décimas de segundo de cada alarma; cada vez que se presente una alarma se ingresa el código de la misma al stack de alarmas y un contador de décimas de segundo de dicha alarma también se ingresa, pero en el stack de contadores (realizando una rutina de push). Esta pila se utiliza para llevar un registro cronológico en forma temporal, tan sólo por el tiempo necesario para reportar la alarma al Centro de Control; una vez reconocida la alarma, se elimina la misma de la pila. El contador de cada alarma se incrementa cada 100 ms, y le sirve al sistema principal para conocer el tiempo que lleva dicha alarma activa.

6.1.2 Protocolo I2C

El protocolo I2C fue diseñado para la transferencia de datos entre circuitos integrados, controladores y memorias EEPROM seriales, ADC's seriales, registros de desplazamiento, etc. En este proyecto se utiliza para comunicar dos controladores. La interfase física del bus consiste de dos líneas de colector abierto, una para el reloj (SCL) y otra para los datos (SDA); cada bus necesita de una resistencia de pull-up, la cual no debe permitir una corriente superior a 3 mA, por lo que la resistencia no puede ser menor a 1600Ω. En el proyecto se utiliza una resistencia de 4.4 KΩ en cada línea. El bus puede tener un maestro y muchos esclavos, y es el maestro el encargado de generar la señal de reloj. El protocolo soporta direccionamiento de 7 o 10 bits.

Todas las transferencias en el bus son iniciadas por el maestro, y son hechas 8 bits a la vez, transmitiéndose primero el bit más significativo. El I2C incluye un

mecanismo de reconocimiento, después de cada transferencia de 8 bits un noveno pulso de reloj es enviado por el maestro, en el cual se transmite un bit de reconocimiento por el dispositivo receptor. El reconocimiento del receptor (ACK) se da cuando el bit es 0, e indica que el dato fue recibido exitosamente, y el no reconocimiento (NACK) se da cuando el bit es 1, e indica que el dato no se pudo recibir en forma correcta. El no reconocimiento (NACK) también se utiliza para terminar una transferencia después de que el último byte es recibido.

Existen 3 condiciones en el bus I2C que deben ser identificadas por los dispositivos esclavos, estas son: Start, Stop y Restart. La condición de Start le dice al dispositivo esclavo que una transacción está iniciando en el bus I2C. Después de una condición de Stop el esclavo debe reiniciar la lógica de recepción del mismo, ya que la transacción ha terminado. La condición de Restart es una condición de Start repetida, es como una combinación entre stop-start, y es utilizada por el maestro para no liberar el bus.

Una operación de escritura en I2C inicia cuando el maestro ejecuta una condición de Start en el bus. Luego el maestro envía un byte de dirección, conteniendo la dirección del esclavo; el bit menos significativo (LSB) indica si la operación I2C es de escritura (LSB = 0) o lectura (LSB = 1). Después de transmitida la dirección, y durante el noveno pulso de reloj, y si la dirección recibida corresponde a la del esclavo, este puede reconocer la recepción del byte enviando el bit ACK. Después de recibir el ACK, el maestro continúa enviando los bytes de datos, y durante el noveno pulso de reloj después de cada dato el esclavo responde con el ACK. Después del último byte de datos un NACK es enviado por el esclavo, para indicar que no se recibirán más datos, después de esto el maestro inicia una secuencia de Stop para liberar el bus.

Una operación de lectura es desarrollada en forma similar a la de escritura, en este caso el bit menos significativo (LSB) de la dirección se pone en 1 para indicar que se trata de una operación de lectura. Cuando el esclavo recibe el byte de dirección, envía el bit de reconocimiento (ACK), y mantiene la línea del reloj (SCL) en

bajo. Manteniendo en cero el reloj, el esclavo puede tomar tanto tiempo como necesite para preparar los datos que se tienen que enviar al maestro. Cuando el dispositivo esclavo está listo, libera el reloj y permite que el maestro extraiga los datos del esclavo, durante el noveno pulso de reloj, después de que el esclavo transmite un byte, el maestro envía el bit de reconocimiento (ACK). Si el esclavo recibe el ACK sigue transmitiendo datos, pero si se recibe NACK la transacción ha terminado, y el esclavo reinicia su lógica I2C y espera por la próxima condición de Start.

El maestro puede iniciar dos tipos de transacciones, un mensaje de lectura de datos o un mensaje de escritura de datos. Para cada uno de ellos se sigue los formatos que se presentan en las figuras 6.9 y 6.10.

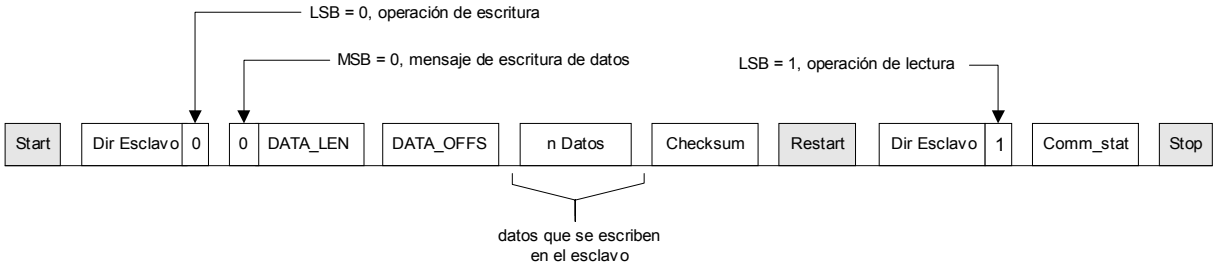


Figura 6.9 Forma del mensaje de escritura de datos en I^2C.

El mensaje de escritura de datos (figura 6.9) inicia con una condición de Start provocada por el maestro. Luego el maestro envía la dirección I2C del esclavo con el LSB en 0, indicando que se va ha realizar una operación de escritura para que la lógica del módulo MSSP se configure de la forma adecuada. El siguiente byte es el *DATA_LEN*, el cual tiene dos propósitos: primero, los 7 bits menos significativos indican el número de bytes de datos que se escribirán en el esclavo, y segundo, el MSB indica si los datos serán leídos del esclavo (MSB = 1) o escritos en el mismo (MSB = 0). El siguiente byte que se envía es *DATA_OFFS* e indica la dirección inicial del buffer de datos a partir del cual se deben escribir. Seguidamente se transmiten el número de bytes indicados en *DATA_LEN* de datos que se escribirán en los registros internos del esclavo; cuando se transmite el último byte de datos, el maestro envía

una sumatoria en complemento a dos de chequeo de error de todos los elementos del paquete, incluyendo la dirección I2C del nodo esclavo. Luego el maestro inicia una condición de Restart para continuar con una operación de lectura, para la cual el maestro debe transmitir la dirección I2C del esclavo con el LSB en 1. En este momento el esclavo está listo para transmitir datos, y le envía al maestro el byte *COMM_STAT*. Para finalizar la transacción el maestro debe enviar un NACK y generar la condición de Stop.

Con el byte *COMM_STAT* el maestro puede informarse del estado del esclavo y si existe un error en el mismo, en la tabla 6.2 se presenta una descripción bit por bit del contenido de dicho registro.

Tabla 6.2 Definición de los bits del registro *COMM_STAT.*

Bit	Nombre de Bit	Descripción
Bit 0	Comm_stat.chkfail	Indica que ocurrió un error el checksum del último mensaje recibido.
Bit 1	Comm_stat.rxerror	Indica que el esclavo no interpretó el último mensaje del maestro correctamente.
Bit 2	Comm_stat.ovflw	Indica que el maestro ha solicitado leer o escribir bytes de datos en el esclavo, fuera del rango válido de direcciones para el esclavo en particular.
Bit 3	Comm_stat.sspov	Indica que un overflow ha ocurrido en el módulo SSP, porque el esclavo no es lo suficientemente veloz para procesar los datos entrantes por I2C.
Bit 4		No usado
Bit 5		No usado
Bit 6		No usado
Bit 7	Comm_stat.r_w	Indica si el último mensaje del maestro corresponde a una operación de lectura (R/W = 1) o de escritura (R/W = 0).

El mensaje de lectura de datos (figura 6.10) inicia con una condición de Start generada por el dispositivo maestro, luego envía la dirección I2C del esclavo con el LSB en 0, indicando que se va ha realizar una operación de escritura. El siguiente byte es el *DATA_LEN*, el cual indica en sus 7 bits menos significativos el número de

bytes de datos que se leerán del esclavo, y en el MSB indica que los datos serán leídos del esclavo (MSB = 1). El siguiente byte que se envía es *DATA_OFFS* e indica la dirección inicial del buffer de datos a partir del cual se leen los datos. Luego el maestro envía el checksum en complemento a dos, para verificar que no existan errores de transmisión. El maestro genera una condición de Restart para continuar con una operación de lectura, la cual inicia cuando transmite la dirección I2C del esclavo con el bit menos significativo en 1. Seguidamente el dispositivo esclavo está habilitado para transmitir, primero envía el byte *COMM_STAT* que le sirve al maestro para conocer si existe algún problema en el esclavo, si *COMM_STAT* no presenta errores, el maestro reconoce el byte y sigue recibiendo los bytes de datos que se indicaron con *DATA_OFFS* y *DATA_LEN*. Cuando el maestro ha terminado de recibir los datos correspondientes, envía un NACK al esclavo y genera una condición de Stop en bus.

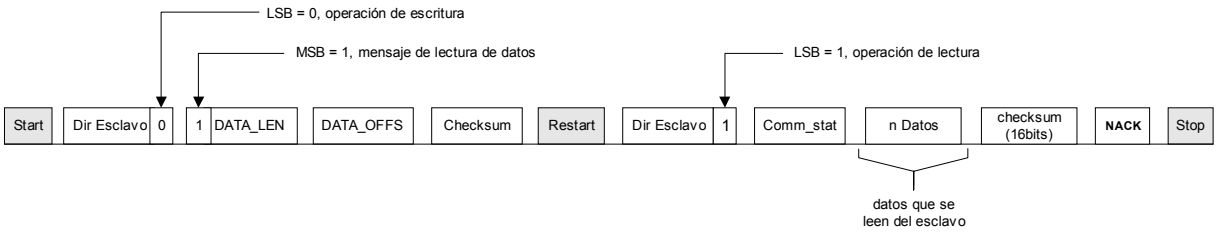


Figura 6.10 Forma del mensaje de lectura de datos en I^2C.

La comunicación I2C se implementa por medio del formato de los mensajes de escritura y lectura presentados en las figuras 6.10 y 6.11, tal como se explicó anteriormente; las rutinas desarrolladas para el maestro y para el esclavo que realizan las tareas de comunicación entre ambos microcontroladores, se presentan en las secciones respectivas.

6.1.3 Programa del microcontrolador maestro

Los programas de los microcontroladores se desarrollaron por medio de lenguaje C, por lo que todas las rutinas que se explican en este documento por

medio de diagramas de flujo fueron realizadas en dicho lenguaje. Microchip® brinda gratuitamente un demo del compilador MPLAB C18, por medio del cual se convierte el código fuente (archivo .c) en un archivo hexadecimal (.hex) que se descarga en el microcontrolador. Junto con el demo proporcionado por MIcrochip®, se incluyen librerías precompiladas que se utilizan en el desarrollo de los programas, para manejar y configurar módulos periféricos. En la tabla 6.3 se presenta una breve descripción de las librerías utilizadas en el programa principal del microcontrolador maestro.

Tabla 6.3 Librerías precompiladas en lenguaje C utilizadas en el programa del PIC maestro.

Librería	Rutinas Utilizadas	Descripción
Usart.h	OpenUsart	Realiza la configuración del módulo USART en el modo y la forma deseada.
Timers.h	OpenTimer0	Configuración del timer 0 en el modo y la forma deseada.
	OpenTimer1	Configuración del timer 1 en el modo y la forma deseada.
Delays.h	Delay10KTCYx	Aplica un delay de x veces 10000 ciclos internos del microcontrolador.

El microcontrolador maestro es el principal elemento de la tarjeta electrónica, ya que es él quien tiene comunicación por medio de protocolo Modbus con el sistema principal en el Centro de Control de Energía, y es él quien tiene en sus registros internos toda la información del estado de las alarmas que monitorea la tarjeta electrónica.

Como se utilizan dos microcontroladores se dividen las tareas de la tarjeta electrónica, por lo que el PIC maestro se encarga de monitorear las señales digitales, y el PIC esclavo monitorea las señales analógicas y el estado de la planta de emergencia. Al realizar funciones distintas, el microcontrolador maestro debe preguntar periódicamente al microcontrolador esclavo sobre el estado de las señales analógicas y de la planta, así mismo, debe brindar al esclavo la información sobre

cuales señales debe monitorear, y cuales son los rangos de operación de la respectiva señal. Otra tarea que debe realizar el PIC maestro es el redireccionamiento de paquetes Modbus dirigidos a la planta, así como esperar la respuesta de la planta y transmitirla al sistema principal en el CENCE.

Se desarrolla un arreglo llamado REGS de 100 elementos de 8 bits cada uno, correspondientes a 50 registros Modbus de 16 bits cada uno, este bloque de memoria está destinado a contener la información de las alarmas que monitorea la tarjeta electrónica. Como puede existir un dispositivo externo (planta de emergencia), local al sitio de repetición, se reservan los primeros 500 registros modbus para dicho equipo. A partir del registro 501 y hasta el 550 se refieren a registros internos de la tarjeta, en los cuales se encuentra información de la misma.

Antes de actualizar estos registros, internamente se modifican registros temporales del estado de las alarmas, con los valores parciales del monitoreo de las señales digitales y analógicas, de tal forma que el arreglo REGS se modifique únicamente con valores definitivos. En el apéndice A.2 se presenta una descripción del contenido de cada uno de los registros Modbus.

El programa principal del microcontrolador maestro realiza la inicialización de las variables y los módulos periféricos, y llama rutinas que realizan tareas específicas; en la figura 6.11 se presenta un diagrama de flujo de dicho programa. En el programa del PIC maestro la primera tarea que se realiza es la inicialización de los timers 0 y 1, así como de los módulos USART y MSSP; también inicializa los puertos de entrada salida y los registros internos del microcontrolador. Cada una de estas rutinas de inicialización se detalla más adelante. Luego el programa principal pone el bit 1 del puerto A (RA1) en 1 y el bit 0 del puerto A (RA0) en 0, los pines correspondientes a RA0 y RA1 se conectan físicamente con las líneas de habilitación de los buffers 74LS244, cada buffer se habilita con un 0, de esta forma se permite el paso de las alarmas digitales del bloque 1 y se deshabilita el bloque 2.

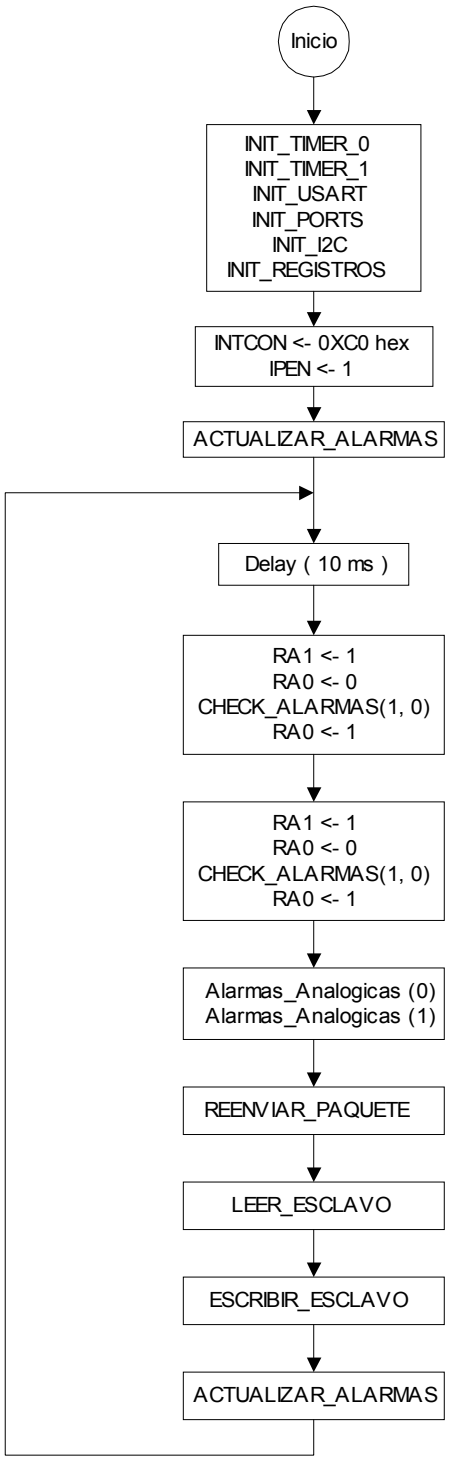


Figura 6.11 Diagrama de flujo del programa principal del microcontrolador maestro.

Inmediatamente se llama a la rutina CHECK_ALARMAS, que se encarga de revisar bit por bit las ocho señales del bloque 1; así mismo, guarda la información

obtenida en los registros temporales de alarmas. Seguidamente deshabilita el bloque 1 de alarmas y se habilita el bloque 2 (RA0 = 1 y RA1 = 0), luego llama a la rutina CHECK_ALARMAS para que se revise bit por bit las señales del bloque 2.

Después de ser monitoreadas las 16 señales digitales, el programa principal llama a la rutina Alarmas_Analogicas, la cual chequea los bits de alarmas reportados por el PIC esclavo, y realiza las operaciones de reconocimiento de alarmas, establecimiento del tipo de alarma, y almacenamiento de la misma en el stack de alarmas, en caso de ser necesario.

Luego por medio de la rutina REENVIAR_PAQUETE, se determina si existe algún paquete Modbus proveniente del sistema central en el CENCE que debe ser reenviado a la planta, de ser así, se transmite el paquete por el bus I2C al PIC esclavo para que este lo transmita a la planta. Cuando un paquete ha sido reenviado, es necesario esperar la respuesta de la misma, y esta rutina se encarga también de dicha tarea.

Para conocer el estado de las alarmas del microcontrolador esclavo, y el valor de las señales analógicas, se llama a la rutina LEER_ESCLAVO, la cual lee los registros internos del PIC esclavo y obtiene esta información, y de ser necesario actualiza los valores correspondientes en los registros temporales de alarmas, para luego cargar estos datos en los registros Modbus del microcontrolador maestro (el contenido de estos registros se presentan en el apéndice A.2).

Si los registros Modbus se modifican, uno o varios bits de configuración han cambiado, y por lo tanto es necesario informarle al PIC esclavo de estos cambios, para ello se llama a la rutina ESCRIBIR_ESCLAVO, la cual envía los nuevos valores de los registros al microcontrolador esclavo.

Por último, después de haber revisado todas las fuentes de alarmas y haber realizado las tareas que le corresponden, se actualizan los valores de los registros Modbus en el arreglo REGS con la información obtenida. Se repite indefinidamente el

monitoreo de las señales, y el intercambio de información con el esclavo, para mantener control total del estado de las alarmas monitoreadas por la tarjeta.

En las figuras 6.12 y 6.13 se presentan los diagramas de flujo de las rutinas encargadas de realizar la inicialización de los timer 0 y timer 1.

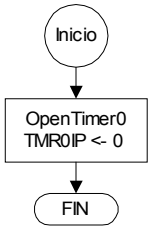


Figura 6.12 Diagrama de flujo de la rutina de inicialización del timer 0.

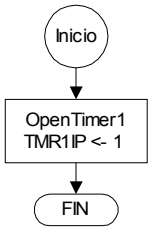


Figura 6.13 Diagrama de flujo de la rutina de inicialización del timer 1.

En ambas rutinas, para la inicialización de los timers se llaman a las rutinas respectivas proporcionadas en la librería precompilada timers.h. Timer 0 se configura en modo de 16 bits, con el reloj interno del microcontrolador como fuente de conteo, por lo que este timer realiza conteos de ciclos internos del PIC, el ciclo interno del PIC es de $4/f_{OSC}$, y el oscilador utilizado para cada microcontrolador es de 10 MHz por un factor de 4 que brinda el PLL interno del PIC, es decir, funciona a 40 MHz, por lo tanto se realizan conteos de 100 ns. El timer 0 se utiliza para manejar la temporización Modbus de los paquetes que se reciben, tal como se explica más adelante. El timer 1 se configura en modo de 16 bits, con el reloj interno del microcontrolador como fuente de conteo, pero con una prescala de 8, lo que quiere decir que el timer 1 se incrementa después de cada 8 ciclos internos del microcontrolador, realizando conteos de 800 ns. Además, se habilita la interrupción de overflow del timer, es decir, cuando el timer 1 experimente un overflow en su conteo se genera una interrupción por dicho evento, y se le da una prioridad alta a la

misma (TMR1IP = 1). Este timer se utiliza para incrementar los contadores de décimas de segundo de las alarmas activas.

En la figura 6.14 se presenta el diagrama de flujo de la rutina de inicialización de la USART, la cual se configura en modo asincrónico, con 8 bits de datos, recepción continua y 9600 bps; además, se habilita la interrupción de recepción en la USART, a la cual se le da baja prioridad (RCIP = 0). Los pines 6 y 7 del puerto C (RC6 y RC7) se configuran con entradas.

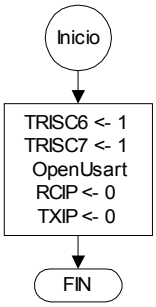


Figura 6.14 Diagrama de flujo de la rutina de inicialización de la USART.

El diagrama de flujo de la figura 6.15 corresponde a la rutina de inicialización de los puertos de entrada salida, donde se escoge el puerto A como salidas digitales (ADCON1 = 0x07 hex, TRISA = 0), y todas se inicializan en 1 (PORTA = 0xFF hex), mientras que el puerto B se configura con todos los pines como entradas digitales (TRISB = 0xFF hex).

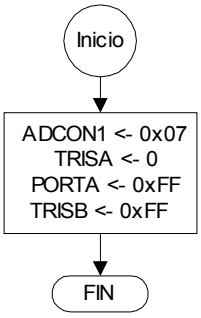


Figura 6.15 Diagrama de flujo de la rutina de inicialización de los puertos de entrada/salida.

El módulo MSSP se configura en modo I^2C, tal como se muestra en el diagrama de flujo de la figura 6.16, en donde los pines 3 y 4 del puerto C (RC3 y RC4) se

habilitan como entradas, el registro SSPADD se carga con la constante CONS_400Hz, la cual contiene el valor correspondiente para que el módulo genere una señal de reloj de 400 Hz. Se le da alta prioridad a las interrupciones de colisión de bus (BCLIP) y de actividad en el módulo I2C (SSPIP), luego se configura el módulo MSSP como maestro (SSPCON1 = 0b00101000).

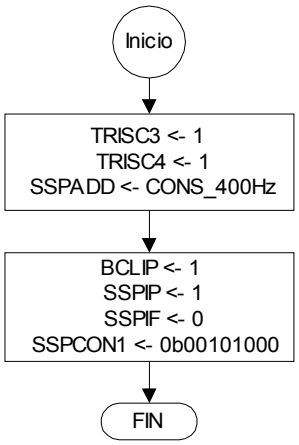


Figura 6.16 Diagrama de flujo de la rutina de inicialización del módulo MSSP.

Para el correcto funcionamiento del programa del PIC maestro es necesario darle algún valor inicial a algunas variables y registros implementados. En la figura 6.17 se presenta el diagrama de flujo de la rutina que inicializa estos registros.

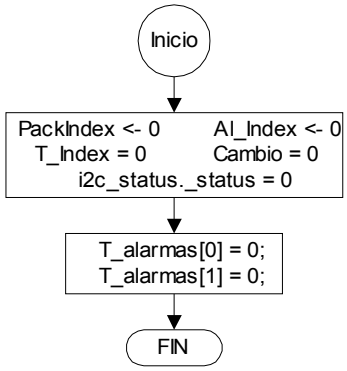


Figura 6.17 Diagrama de flujo de la rutina de inicialización de los registros internos.

Es necesario limpiar los apuntadores o índices de los arreglos que reciben los paquetes Modbus, y que contienen las alarmas activas; así como la variable cambio y al byte de estado i2c_status. Los registros de alarmas temporales también deben

limpiarse. En la explicación de futuras rutinas se detallará el uso de cada una de estas variables y registros.

En la figura 6.11 después de la inicialización, vienen las rutinas que realizan tareas específicas, como monitorear señales digitales, preguntar al esclavo, actualizar registros en el esclavo, etc. Antes de explicar cada una de estas rutinas, se presentan las rutinas de atención de interrupción.

Cuando se presenta una interrupción de baja prioridad, el hardware del PIC realiza un direccionamiento del flujo del programa la rutina Interrupcion_L, cuando esta rutina termina de ejecutarse, se vuelve a la misma rutina y misma instrucción donde se estaba durante la activación de la interrupción. En la figura 6.18 se presenta el diagrama de flujo de la rutina Interrupcion_L.

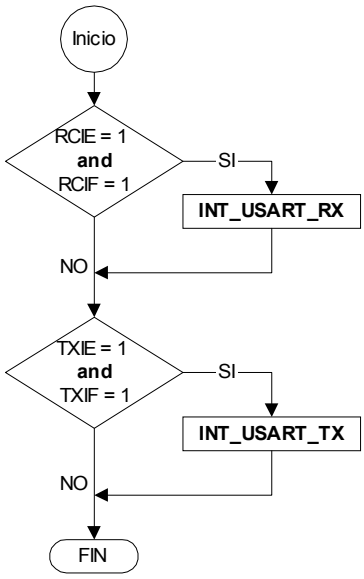


Figura 6.18 Diagrama de flujo de la rutina de interrupción de baja prioridad.

En dicha rutina se chequea que el llamado a interrupción se halla dado por la recepción de un carácter en la USART, para ello las banderas de habilitación de la interrupción de recepción de la USART (RCIE), y la bandera de interrupción de recepción en la USART (RCIF) deben estar en uno. Para procesar el carácter recibido se llama a la rutina INT_USART_RX. Pero si las banderas de habilitación de la interrupción de transmisión de la USART (TXIE), y la bandera de interrupción de

transmisión en la USART (TXIF) están en uno, entonces el llamado ha interrupción se dio porque la USART está lista para transmitir otro carácter, y entonces se llama a la rutina INT_USART_TX

Cuando la interrupción que se presenta es de alta prioridad, el flujo del programa se direcciona a la rutina Interrupcion_H. En la figura 6.19 se muestra el diagrama de flujo de la rutina de atención de interrupción de alta prioridad.

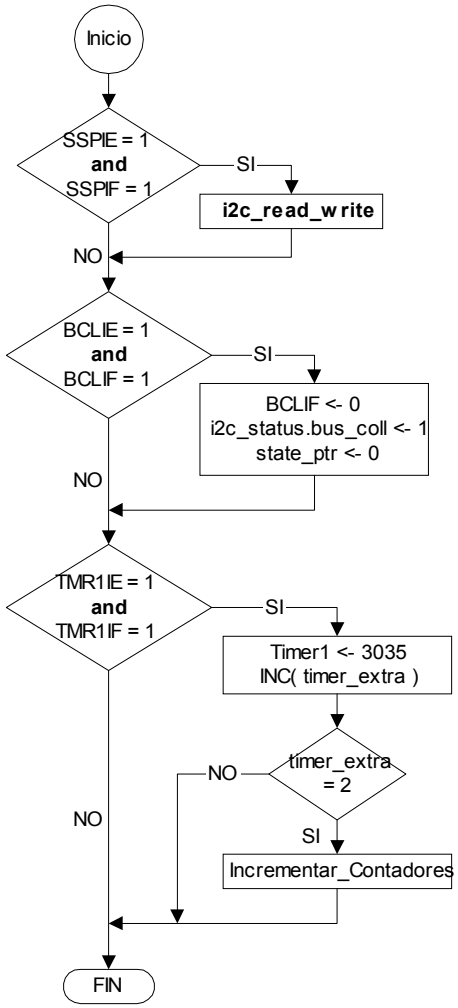


Figura 6.19 Diagrama de flujo de la rutina de interrupción de alta prioridad.

En la cual se prueba si la bandera de habilitación de interrupción del módulo MSSP y la bandera de interrupción del mismo están activas, de ser así, se llama a la rutina i2c_read_write.

Cuando la interrupción se presenta por una colisión de bus en el módulo I2C, las banderas BCLIE y BCLIF están activas, se hace necesario limpiar la bandera de interrupción y activar el bit de error por colisión de bus en el byte i2c_status.bus_coll.

Sí la bandera de habilitación de interrupción del timer 1 y la bandera de interrupción del timer 1 están activas, se carga el timer 1 con 3035, de tal forma que el mismo realice un conteo de 62500 hasta que se presente la interrupción de overflow, luego se incrementa la variable timer_extra, y si la misma es 2, se incrementa el conteo de los contadores de décimas de segundo de las alarmas activas. De esta forma los contadores se incrementan cada 2*62500 conteos del timer 1, lo que equivale a 100 ms.

Como se mostró en la figura 6.18, cuando llega un carácter por el puerto serie se llama a la rutina INT_USART_RX. Cuando esta situación ocurre, es porque el sistema principal en el CENCE está transmitiendo un paquete Modbus a la tarjeta electrónica, y el microcontrolador maestro debe responder a la petición del mismo. El diagrama de flujo de la rutina INT_USART_RX se presenta en la figura 6.20, dicha rutina es la que se encarga de recibir los caracteres por el puerto serie, y organizarlo para reconstruir adecuadamente el paquete Modbus que se envió remotamente. El timer 0 se utiliza para controlar la temporización Modbus de cada paquete que recibe el PIC maestro.

En el diagrama de flujo mostrado en la figura 6.20, los datos recibidos se almacenan en un arreglo llamado *Pack*, y la variable *PackIndex* determina el número de elementos en *Pack*.

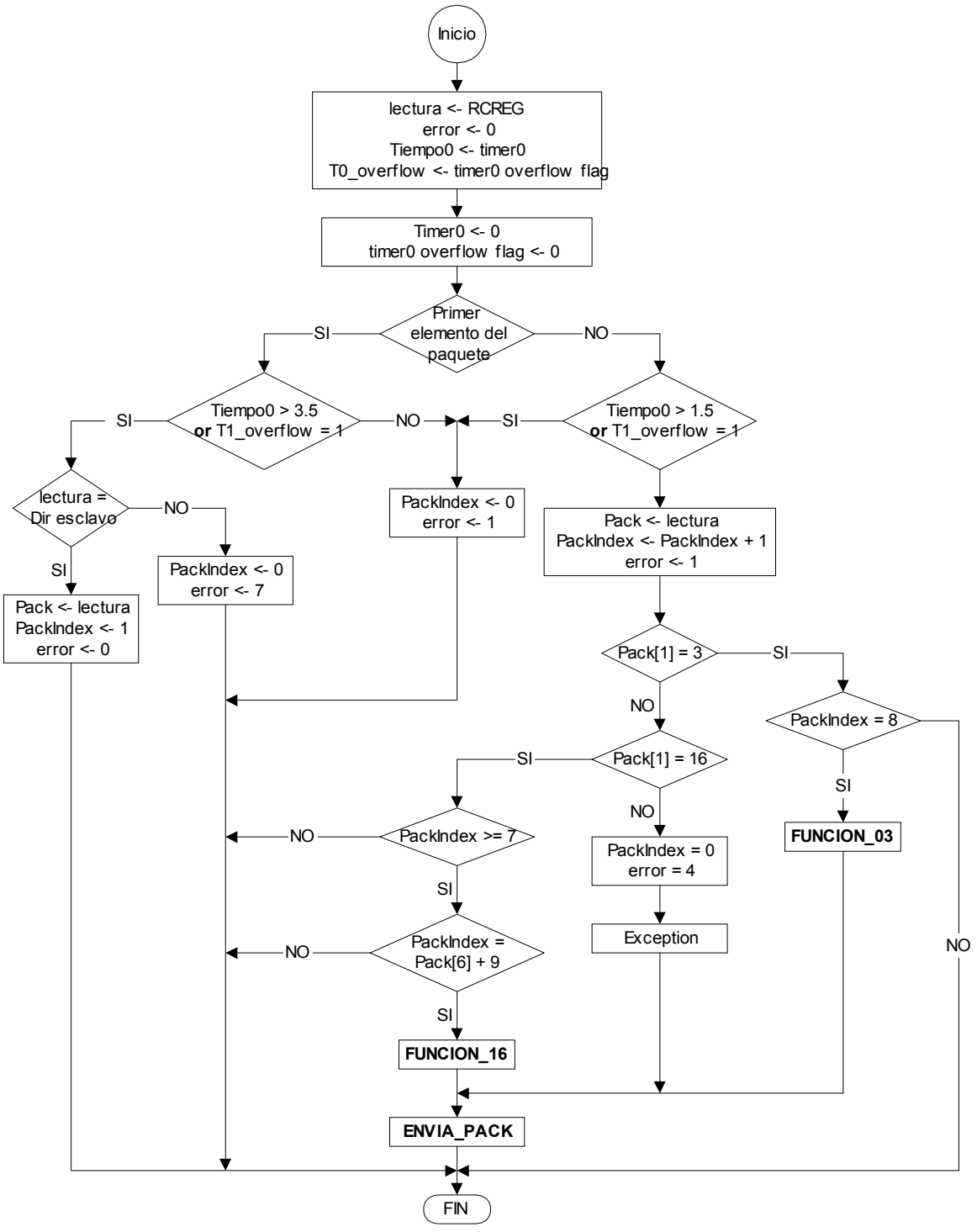


Figura 6.20 Diagrama de flujo de la rutina INT_USART_RX.

Cada carácter recibido se almacena en la variable temporal *lectura*, si este corresponde al primer carácter del paquete (cuando PackIndex = 0), se compara el

valor del timer 0 (almacenado previamente en la variable tiempo0) con la constante equivalente al tiempo de transmisión de 3.5 caracteres (T35), si el conteo del timer es menor, o bien, se ha presentado un overflow en el timer, se genera el error correspondiente ya que no se cumple con las especificaciones del protocolo, y se limpia el contenido de *Pack* asignando 0 elementos en *PackIndex*; si el valor de tiempo0 es mayor, se compara lectura con la dirección del esclavo, si es la misma se agrega *lectura* a Pack y se incrementa PackIndex, de lo contrario se mantiene en cero elementos, y se toma el próximo carácter recibido como el primero de otro paquete. Después de recibir cada carácter el timer 0 es reiniciado, y la bandera de overflow del timer 0 se limpia, para chequear la temporización Modbus del próximo carácter.

Si el carácter recibido no es el primero del paquete (*PackIndex* $\neq$ 0), se verifica que se cumpla con la temporización Modbus (el tiempo entre un byte recibido y otro no puede ser superior al tiempo de transmisión de 1.5 caracteres, T15). Si no se cumple con la temporización, se genera el error correspondiente y se limpia *Pack*. Si el carácter recibido es correcto, se verifica que la función indicada en el mismo sea la 03 o la 16 (para ello el elemento 2 de *Pack* debe ser 03 ó 16, y como ya se recibió un carácter anteriormente, se cuenta con al menos 2 caracteres en *Pack*), de lo contrario se genera un mensaje de excepción, ya que no se puede manejar la función indicada.

Si la función del paquete es la 03, se siguen recibiendo caracteres hasta que el paquete este completo, es decir, cuando *PackIndex* sea 8, y se llama a la rutina FUNCION_03. Para la función 16 el número de caracteres en el paquete es variable, ya que depende del número de datos que contenga, la longitud de dicho paquete se obtiene del valor almacenado en el elemento 7 de *Pack* más el número de bytes fijos en el paquete (en este caso 9), para entender mejor esto se puede observar la forma del paquete Modbus para función la 16 en la figura 6.6. Luego se hace un llamado a la rutina FUNCION_16.

Una vez realizada la acción correspondiente a cada función, se transmite el paquete de respuesta que se encuentra en *Pack*, por medio de la rutina Envia_Pack. Después de la transmisión se limpia *Pack* asignando cero a *PackIndex*, y se toma el siguiente byte recibido por el puerto serie como el primero del próximo paquete. Cuando se termine de ejecutar la rutina de interrupción, el control del programa vuelve a donde se había quedado cuando se recibió el último carácter en el puerto serie.

La rutina que realiza la función 03 se presenta en la figura 6.21, por medio de diagramas de flujo. En esta rutina se reciben los paquetes que cumplan con todas las condiciones indicadas por el protocolo Modbus.

En dicho diagrama de flujo, lo que se realiza inicialmente es verificar el error en el paquete, con el recálculo del CRC y la comparación con el valor enviado en el paquete (en la figura 6.3 se presenta la rutina de cálculo del CRC). Si existe algún error, no se realiza la acción indicada y no se genera respuesta al maestro. Si el CRC es correcto, se determina si el paquete es para la planta (para ello se revisa el campo correspondiente a dirección inicial, la cual debe ser menor a 500), de ser así, se verifica la existencia de la misma, si hay una, se redirecciona el paquete a la planta por medio de la rutina REENVIAR_PACK; si no hay planta se genera y envía una respuesta de excepción, indicando que el registro direccionado es inválido. Si la dirección inicial en el paquete es mayor al número total de registros que se puede direccionar en la tarjeta, se genera y envía una excepción, indicando que la dirección recibida es inválida. Cuando se recibe un valor correcto de dirección inicial, se verifica que el número de registros a leer no supere el número máximo de registros en la tarjeta, cuando se tenga un valor válido de dirección inicial y un valor válido del número de registros a leer, se direcciona el registro deseado en la tarjeta, y a partir del mismo se copia el número de registros a leer indicado por el paquete de petición. Todos estos datos se cargan en *Pack*, se realiza el cálculo del CRC y se agrega al final del paquete, para ser transmitido por la rutina ENVIA_PACK como respuesta al maestro.

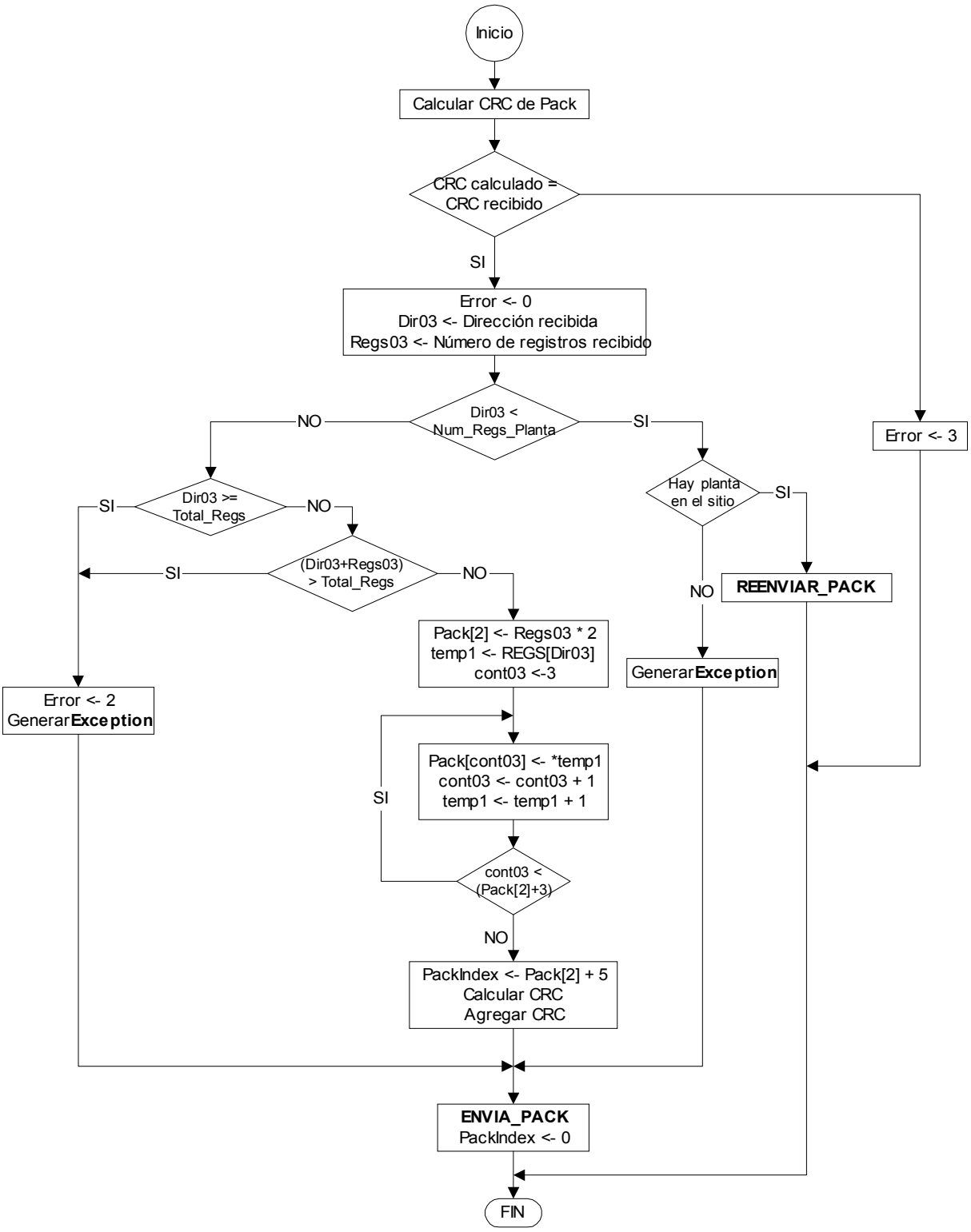


Figura 6.21 Diagrama de flujo de la rutina que ejecuta la función 03 Modbus.

Indice

En la figura 6.22 se presenta el diagrama de flujo de la rutina de la función 16.

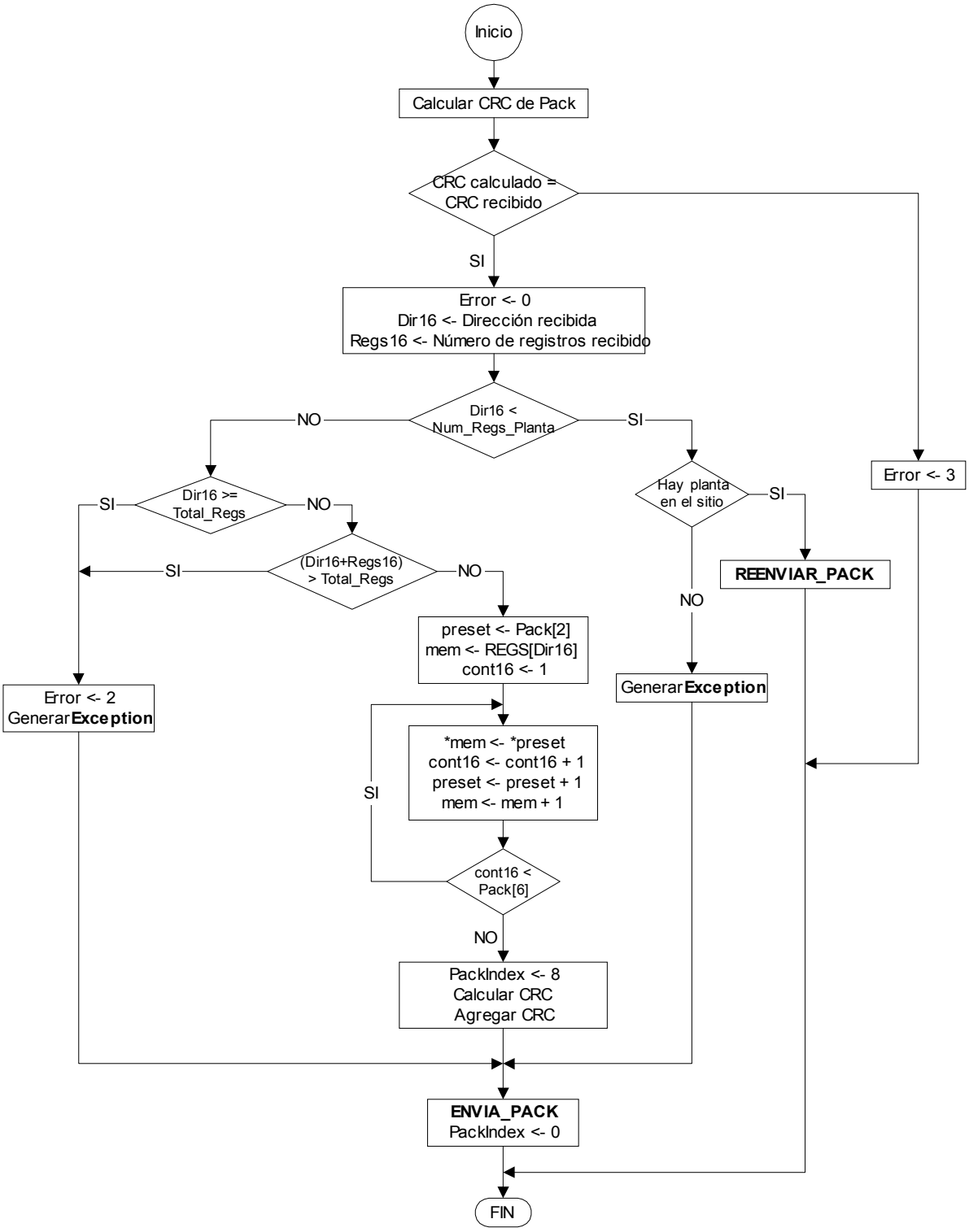


Figura 6.22 Diagrama de flujo de la rutina que ejecuta la función 16 Modbus.

Indice

A la rutina que ejecuta la función 16 también llegan únicamente paquetes que cumplen con las condiciones del protocolo Modbus, inicialmente se realiza el cálculo del CRC y se compara con el valor recibido en el paquete, si existe algún error no se envía respuesta. Si el paquete llega sin errores se determina si la dirección inicial corresponde a un registro de la planta, si es para la planta y existe una, se redirecciona el paquete a la misma, de lo contrario, se genera una respuesta de excepción indicando que el registro direccionado es inválido. Si la dirección de registro inicial no corresponde a la planta, se determina si es mayor al número máximo de registros de la tarjeta, de ser así, se genera una excepción indicando que el registro direccionado es inválido.

Cuando se recibe un valor correcto de dirección inicial, se verifica que el número de registros a escribir no supere el número máximo de registros en la tarjeta, cuando se tenga un valor válido de dirección inicial y un valor válido del número de registros a escribir, se direcciona el registro deseado en la tarjeta, y a partir del mismo se copian los datos incluidos en el paquete. Finalmente se crea un paquete de respuesta en *Pack*, se realiza el cálculo del CRC y se agrega al final del paquete, para ser transmitido por medio de la rutina ENVIA_PACK.

La rutina que crea los paquetes de excepción Modbus se presenta en diagramas de flujo en la figura 6.23.

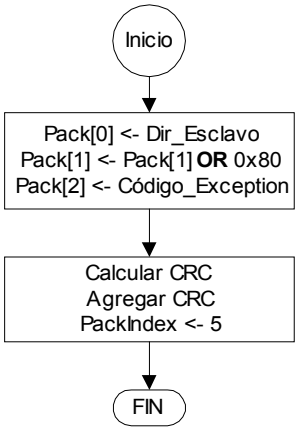


Figura 6.23 Diagrama de flujo de la rutina que crea paquetes de excepción Modbus.

Dicho paquete tiene una longitud de 5 elementos, e incluye: la dirección del esclavo que genera la respuesta, la función recibida en la petición que generó la excepción, pero con el bit más significativo en 1, el código de excepción que indica la causa de la excepción, y por último el campo del CRC.

El diagrama de flujo mostrado en la figura 6.24, corresponde a la rutina que realiza la transmisión de un arreglo determinado a través del puerto serie. El arreglo a enviarse debe tener una longitud conocida, la cual se carga en la variable *tx_cont*, esta longitud se utiliza para saber cuando se termina de enviar el arreglo, la dirección del primer elemento del arreglo se carga en apuntador *tx_ptr*. Luego se habilita la interrupción de transmisión del puerto serie.

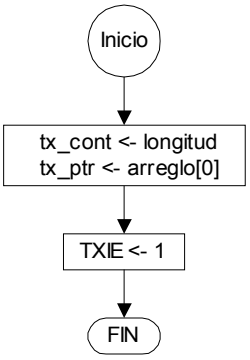


Figura 6.24 Diagrama de flujo de la rutina que transmite un arreglo por la USART.

Como se habilitó la interrupción de transmisión de la USART, cada vez que el registro de transmisión este vacío, se va generar la interrupción, y se llamará a la rutina INT_USART_TX. El diagrama de flujo de dicha rutina se presenta en la figura 6.25, en donde cada elemento del arreglo se carga en el registro de transmisión de la USART (TXREG), se incrementa el apuntador al siguiente elemento del arreglo y se decrementa *tx_cont*. Cuando *tx_cont* es 0 se han transmitido todos los elementos del arreglo, por lo que se deshabilita la interrupción de transmisión del puerto serie.

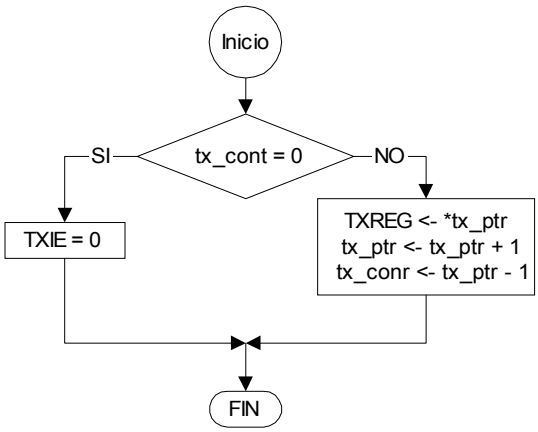


Figura 6.25 Diagrama de flujo de la rutina INT_USART_TX.

Una de las tareas importantes que debe realizar el microcontrolador maestro es el monitoreo de las 16 señales digitales, y tal como se mostró en la figura 6.11 esto lo realiza habilitando un registro 74LS244 a la vez, por medio de los pines RA0 y RA1, donde cada uno corresponde a un bloque de 8 alarmas digitales. Cuando hay un bloque de alarmas listas para monitorearse, se llama a la rutina CHECK_ALARMAS, la cual se muestra por medio de diagramas de flujo en la figura 6.26.

En este diagrama de flujo, primero se cargan los registros de mascara, reconocimiento y alarmas activas, de las 8 señales que se monitorean en el momento. Luego bit por bit se evalúa cada señal, primero se determina si la alarma correspondiente está habilitada, si está inhibida, se limpian los bits de alarma y reconocimiento de la alarma respectiva, si anteriormente había una alarma, esta se extrae del Stack de alarmas, por medio de la rutina Del_alarma. Si la alarma está habilitada, se determina si la alarma está activa, si no hay alarma, y anteriormente si existía, se extrae la alarma del Stack, y se limpian los bits de alarma y reconocimiento, luego se activa Cambio. Si la alarma existe, y anteriormente no existía, se activa el bit de alarma y Cambio, además se agrega al Stack la alarma correspondiente (con Push_Alarma), si la alarma existía, y se reconoció la alarma, la misma se extrae del Stack de alarmas. El análisis anterior se realiza para cada bit del puerto B, que corresponde a una señal digital, y se realiza para bloques de 8 señales

digitales. Luego se actualizan los registros temporales con los nuevos datos de alarmas, reconocimiento, y máscara.

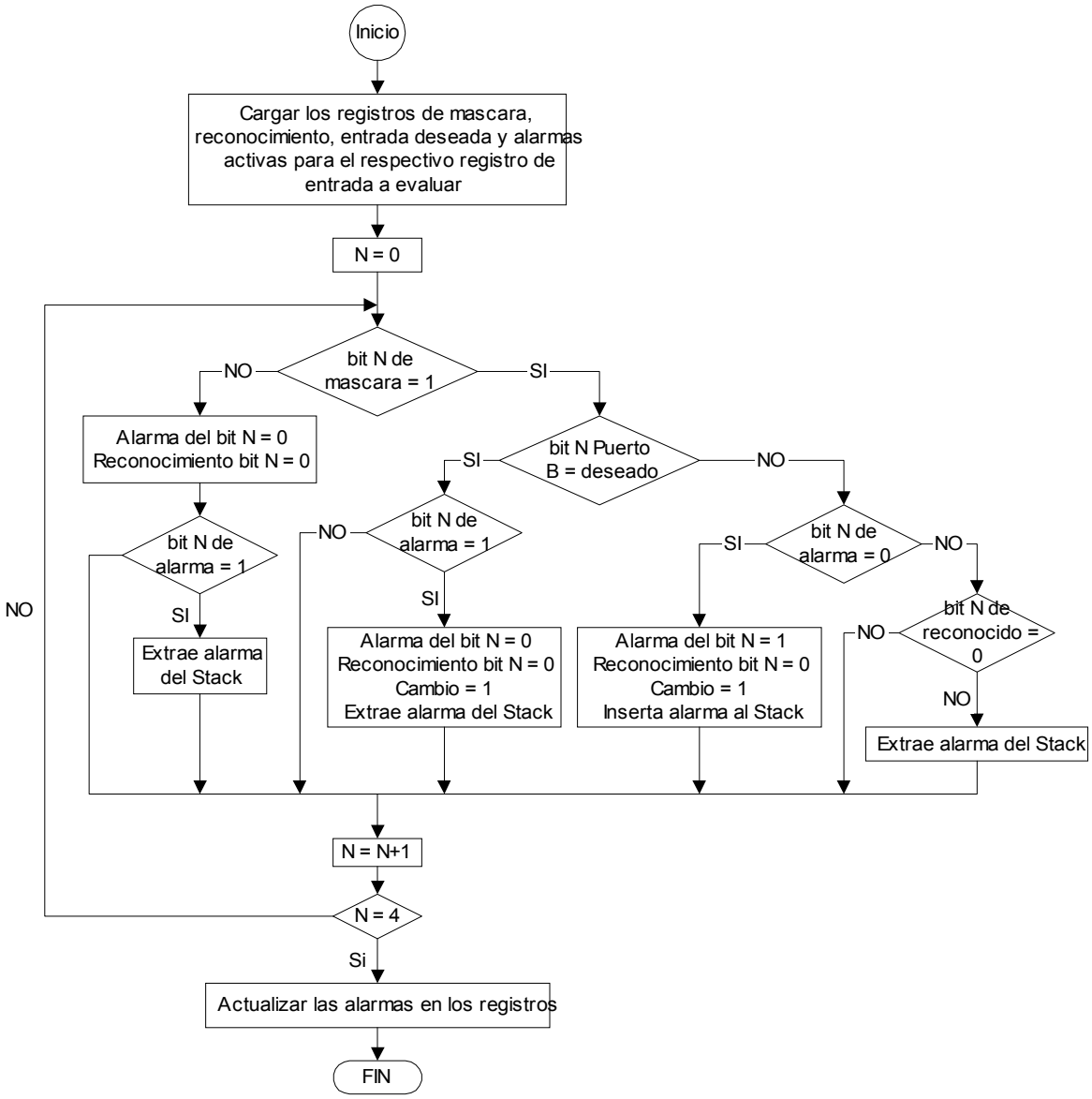


Figura 6.26 Diagrama de flujo de la rutina que chequea 8 alarmas digitales.

Cada vez que se agrega una alarma al Stack se llama a la rutina Push, y para eliminar una alarma del mismo a la rutina Del. En la figura 6.27 se presenta el diagrama de flujo de la rutina Push.

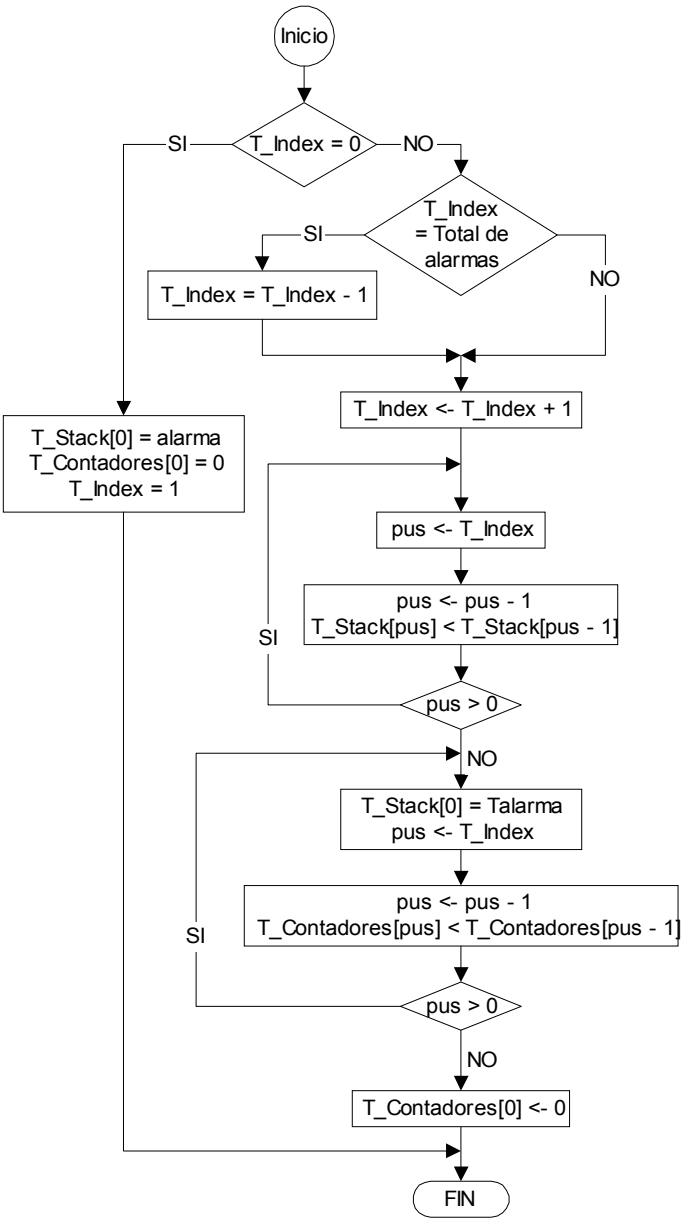


Figura 6.27 Diagrama de flujo de la rutina Push.

La rutina mostrada en el diagrama de flujo anterior, crea un registro de desplazamiento, cada vez que se agrega una alarma, se desplazan las alarmas existentes y la nueva alarma se agrega al inicio. Adicionalmente, se agrega un contador iniciado en cero en el stack de contadores para la alarma que recién se ingresa. Durante la actualización del stack de alarmas se inhibe la interrupción del timer1, para evitar que se realice un incorrecto incremento en el stack de contadores.

Indice

Por otra parte la rutina que saca una alarma del stack, primero busca la alarma en el stack, luego realiza un desplazamiento desde la última alarma hasta la alarma que se desea eliminar, sobrescribiendo en ella.

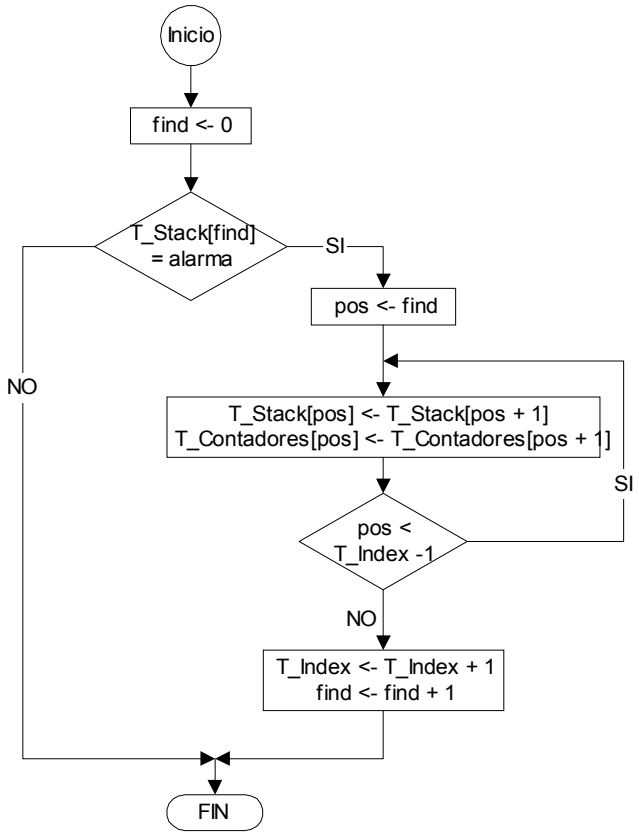


Figura 6.28 Diagrama de flujo de la rutina Del.

Durante la actualización del stack de alarmas la interrupción del timer1 se deshabilita.

Volviendo al diagrama de flujo de la figura 6.11, después de chequear las señales digitales, el programa principal llama a la rutina Alarmas_Analogicas, el diagrama de flujo de esta rutina se presenta en la figura 6.29.

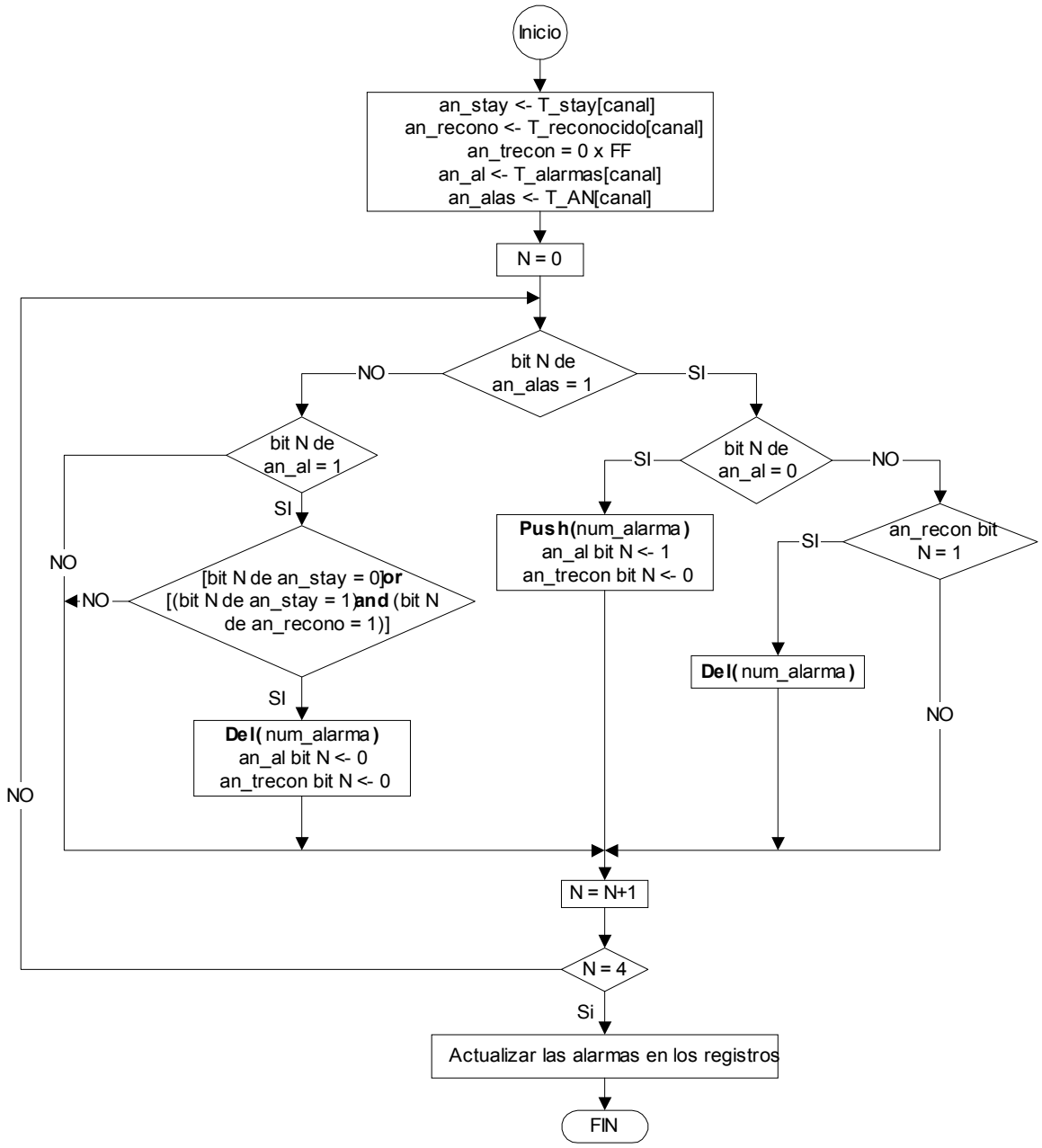


Figura 6.29 Diagrama de flujo de la rutina Alarmas_Analogicas.

En la rutina Alarmas_Analogicas, se cargan en variables temporales los valores de los registros stay, reconocido, alarmas y las alarmas analógicas reportadas por el microcontrolador esclavo. Luego se chequea bit por bit las alarmas reportadas por PIC esclavo (alas), cuando hay una alarma, y la alarma correspondiente no ha sido activada en los registros internos del microcontrolador maestro, la alarma

correspondiente se activa y se agrega en el stack de alarmas, además, se limpia el bit de reconocimiento correspondiente. Si la alarma analógica esta activa, y ya se ha activado el bit de alarma correspondiente en los registros internos del PIC maestro, se verifica si la alarma se ha reconocido y se extrae la alarma del stack. Cuando no existe alarma analógica en el PIC esclavo, y la alarma en los registros del microcontrolador maestro está activa, se extrae la alarma del stack y se desactiva el bit correspondiente, siempre que se halla reconocido la alarma, o bien, la alarma no sea tipo ‘’mantenida’’. El chequeo de las señales analógicas se realiza todas las alarmas activas.

Después de realizar estas tareas en el programa principal, el mismo procede a leer los registros internos del PIC esclavo. Para ello llama a la rutina que se muestra en el diagrama de flujo de la figura 6.30.

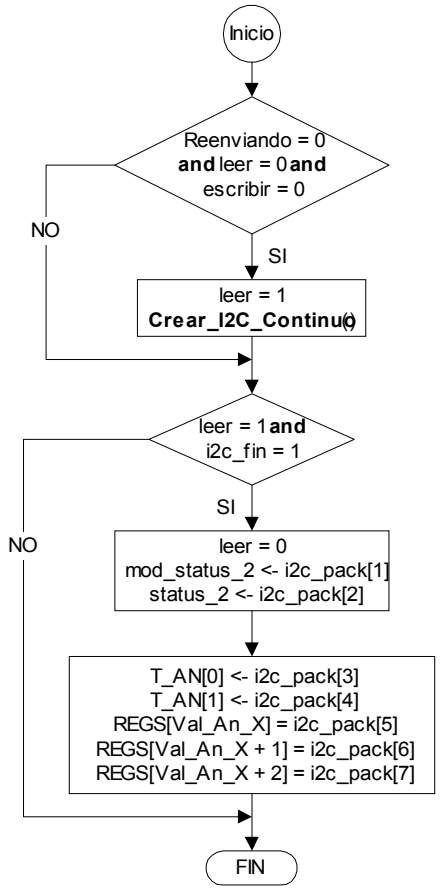


Figura 6.30 Diagrama de flujo de la rutina LEER_ESCLAVO.

Indice

En esta rutina se chequea el estado de las banderas: reenviando, leer y escribir; si estas banderas están en cero el bus I^2C está libre y preparado para transmitir, por ello se llama a la rutina Crear_I2C_Continuo, que crea el paquete de lectura de datos en el esclavo y lo transmite. Además, chequea las banderas leer y fin, ya que si ambas están activas quiere decir que se ha transmitido un paquete de lectura de datos y la respuesta ya se ha recibido, por lo que ya se pueden leer los datos recibidos del esclavo, con los cuales se actualizan los valores de las variables temporales de alarmas, el byte de status y los valores de las señales analógicas.

Siguiendo con el diagrama de flujo de la rutina del programa principal (figura 6.11) se hace un llamado a la rutina ESCRIBIR_ESCLAVO, la cual se presenta en la figura 6.31.

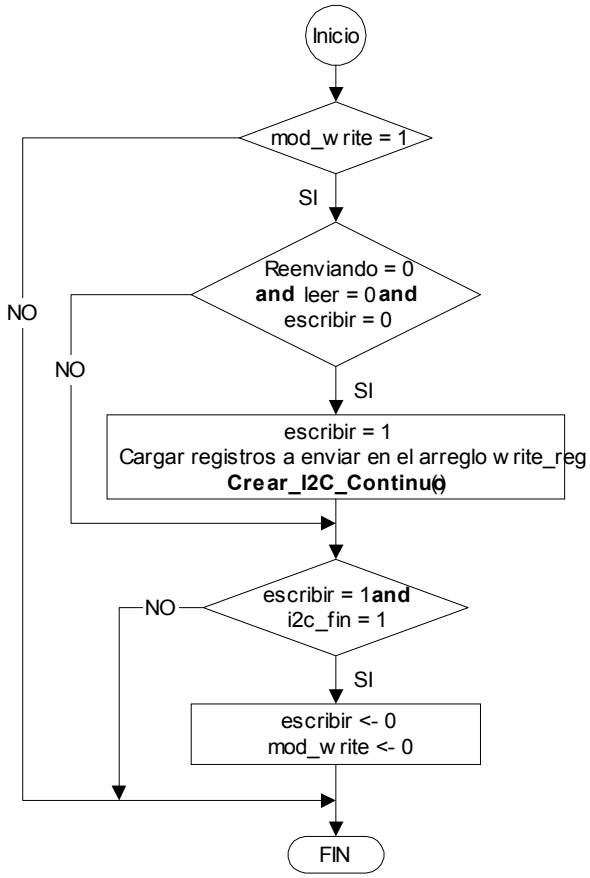


Figura 6.31 Diagrama de flujo de la rutina ESCRIBIR_ESCLAVO.

Esta rutina se ejecuta cuando la bandera mod_write esta activa, la misma se pone en uno cuando el sistema realiza la función 16 modbus, ya que esto indica que los registros internos del microcontrolador maestro ha sido modificado, y por lo tanto es necesario modificar los registros internos del PIC esclavo. Si es necesario escribir al microcontrolador esclavo se verifica que el bus I^2C este libre, luego se activa la bandera escribir (esto para tomar posesión de dicho bus) y se cargan los datos que se desean transmitir al esclavo, los cuales son los registros Modbus del maestro que dicen como debe trabajar el PIC esclavo; luego se llama a la rutina Crear_I2C_Continuo para crear el paquete y enviarlo por el bus I^2C. Por último se chequean las banderas escribir y fin, si ambas están en uno entonces ya se terminó de transmitir el paquete, y se deben limpiar las banderas correspondientes.

Una rutina muy importante y que se ha mencionado anteriormente es la rutina Crear_I2C_Continuo, la cual crea el paquete I^2C que se va enviar e inicia la transmisión del mismo, el diagrama de flujo del mismo se presenta en la figura 6.32.

Esta rutina inicia cargando con 1 el puntero state_ptr, y read_write con 1. Luego se cargan el registro inicial y la cantidad de registros que se leerán o escribirán, y se carga la dirección del esclavo I^2C. Si el paquete es de lectura, el MSB del byte de data_len se pone en uno, y la longitud del paquete se actualiza (i2c_lon <- 3). Luego si el paquete es de escritura, se limpia read_write y se cargan los datos que se escribirán en el esclavo y se actualiza el valor de i2c_lon, que contiene el número de elementos en i2c_pack. Cuando el paquete está listo se realiza el cálculo del checksum y el valor del mismo se agrega al final del i2c_pack. Por último al puntero i2c_ptr se le asigna la dirección del primer elemento de i2c_pack, y a temp_cont el valor en i2c_lon, además, se limpia la bandera de interrupción por colisión de bus (BCLIF) y se activan las banderas de habilitación de interrupción por colisión de bus (BCLIE), habilitación de interrupción del módulo MSSP (SSPIE), y la bandera de interrupción del módulo MSSP (SSPIF).

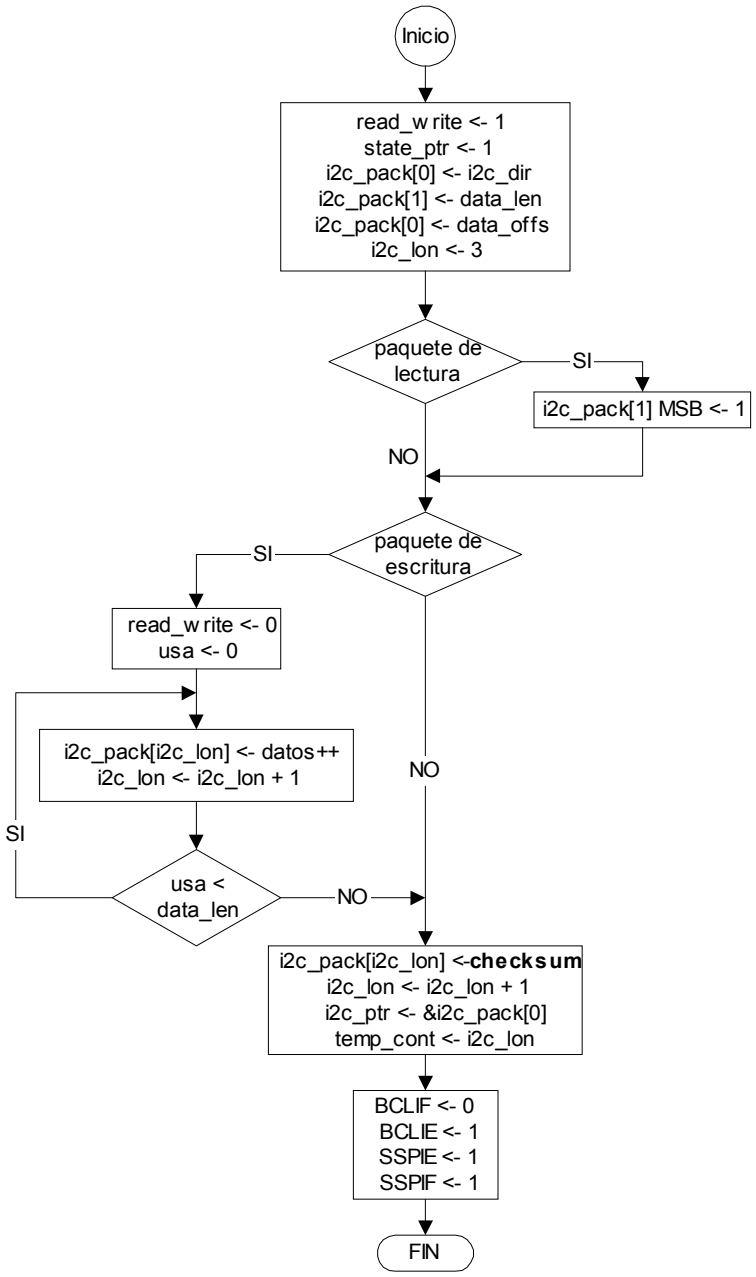


Figura 6.32 Diagrama de flujo de la rutina Crear_I2C_Continuo.

Cuando se activa la bandera SSPIF se hace un llamado a la interrupción de alta prioridad y se llama a la rutina I2C_Read_Write, tal como se muestra en la figura 6.19. Esta es la que se encarga de transmitir el paquete de datos de lectura o escritura incluidos en i2c_pack, y de longitud i2c_lon. El diagrama de flujo de la rutina se presenta en la figura 6.33.

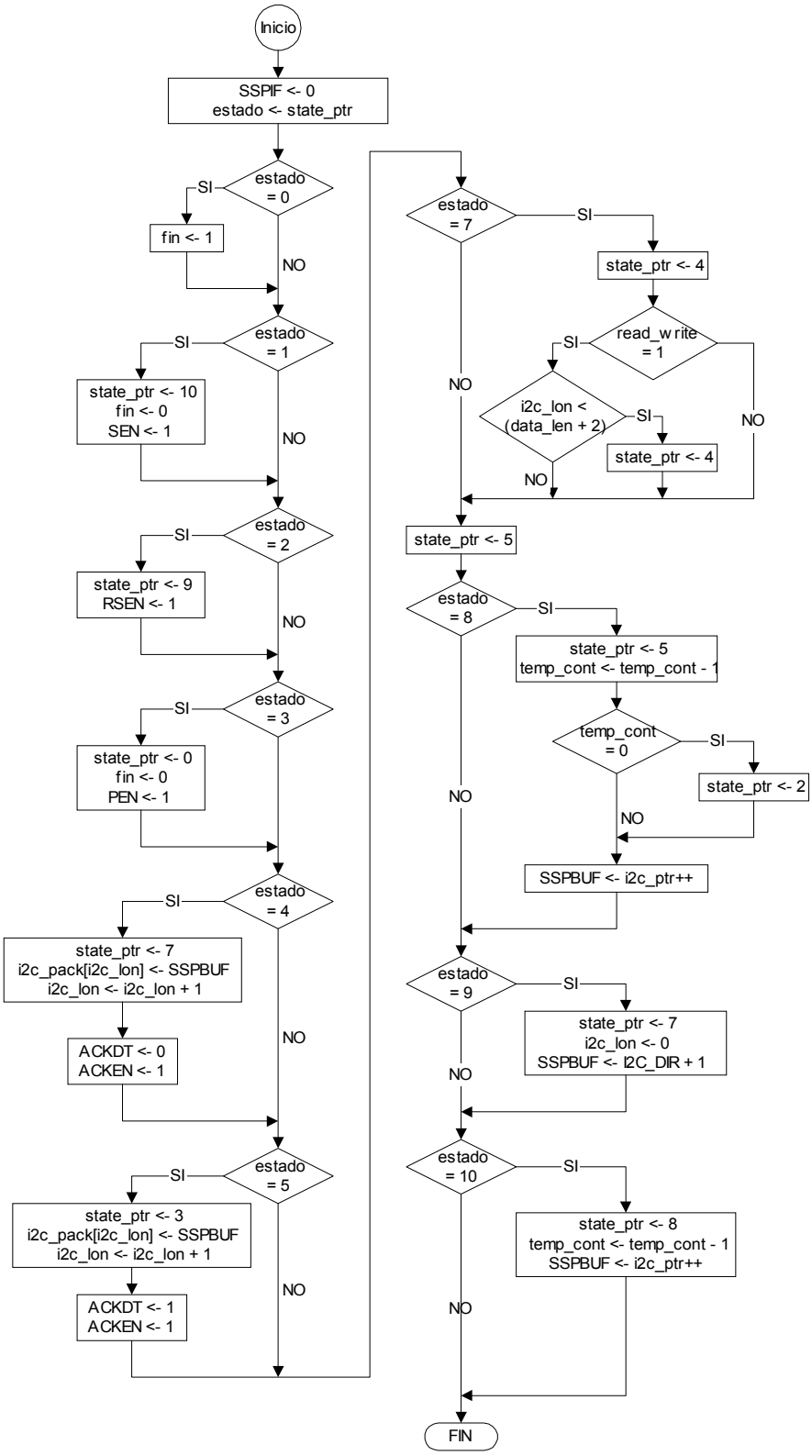


Figura 6.33 Diagrama de flujo de la rutina I2C_Read_Write.

Indice

Para realizar dicha tarea, esta rutina utiliza 10 diferentes estados, en los cuales se realiza una tarea específica de acuerdo al protocolo I^2C descrito anteriormente. El estado 0 es el último estado en el protocolo I^2C, y en este se activa la bandera fin. El estado 1 activa una condición de START en el bus, y apunta al siguiente estado del protocolo (estado 10). En el estado 2 se activa la condición de RESTART y se apunta al siguiente estado en el protocolo I^2C (estado 9). La condición de STOP se genera en el estado 3, además, se apunta al último estado del protocolo I^2C (estado 0). En el estado 4 se recibe un byte del esclavo y se carga en el arreglo i2c_pack, y se incrementa el valor en i2c_lon, se chequea si es el byte correspondiente a COMM_STAT, de ser así, se determina la existencia de errores; por último, genera el bit de reconocimiento al esclavo. En el estado 5 también se recibe un byte del esclavo, pero en este caso no se envía reconocimiento al esclavo, por lo que se recibe el último byte del paquete. En el estado 7 se habilita al maestro en modo recepción, y se apunta al siguiente estado: estado 5 si faltan datos de recibir, o estado 4 si solo falta un dato por recibir. El maestro escribe datos en el estado 8, aquí el valor apuntado por i2c_ptr se carga en SSPBUF y se incrementa el apuntador; si faltan datos por enviar el próximo dato seguirá siendo el 8, pero si ya se han enviado todos los datos el siguiente estado es el 2. En el estado 9 se carga la dirección I^2C del esclavo en el registro SSPBUF con el bit más significativo en 1. Y por último, en el estado 10 se carga la dirección del esclavo con el MSB en 0.

Cada vez que se transmiten datos por el bus I^2C se debe activar la bandera de interrupción SSPIF, para que la rutina I2C_Read_Write estado por estado transmita la información y espere la respuesta.

La última rutina que ejecuta el programa principal de la figura 6.11 es la de actualización de los valores de los registros Modbus, el diagrama de flujo de dicha rutina se presenta en la figura 6.34. Cuando la rutina Actualizar_Alarmas se ejecuta se deshabilitan las interrupciones del timer1 y de recepción del puerto serie, para evitar incrementos incorrectos en el stack de contadores, y que el sistema principal en el CENCE lea información parcial en los registros Modbus del PIC maestro.

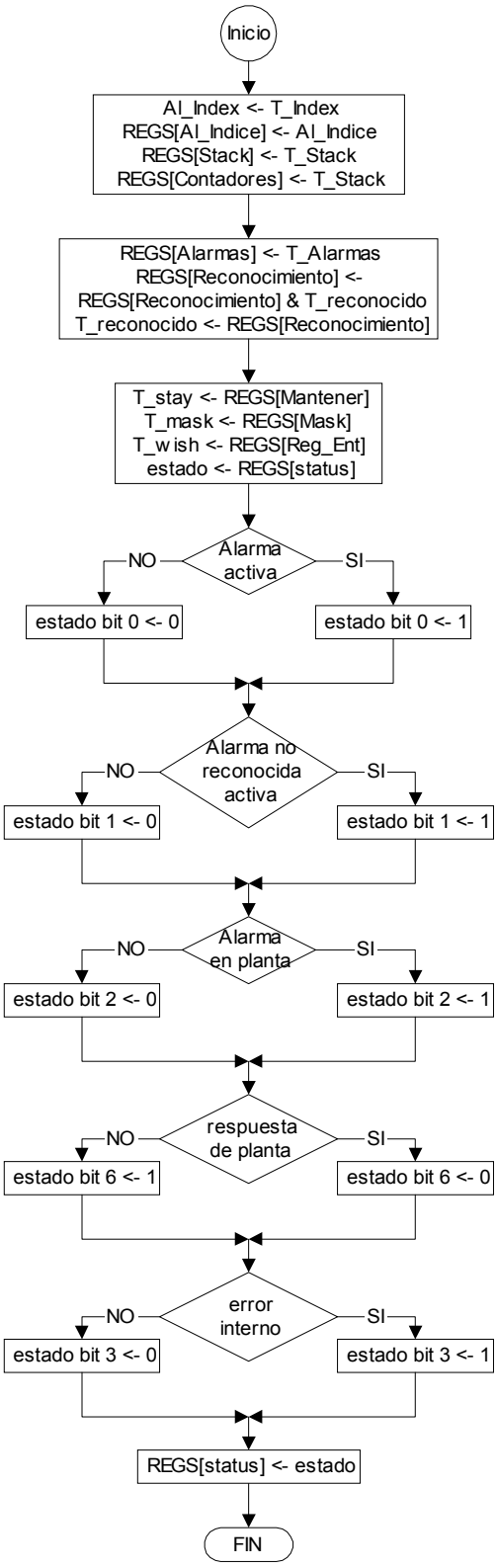


Figura 6.34 Diagrama de flujo de la rutina Actualizar_Alarmas.

Indice

La primera acción que se realiza es cargar el valor del índice temporal del stack de alarmas en la variable global para este propósito (Al_Index) y en el registro Modbus correspondiente (REGS[Al_Indice]). Seguidamente pasa el contenido de los stacks temporales de alarmas y de contadores a los stacks de alarmas y contadores en los registros Modbus. Luego las alarmas que se han registrado en los registros temporales se cargan en los registros Modbus (REGS[Alarmas]). El valor de los registros temporales de reconocimiento se almacena en los registros Modbus aplicando una operación and con el valor almacenado previamente, esto para limpiar los bits de reconocimiento de las alarmas que ya se han desactivado. Y después de aplicar la máscara a estos bits, se vuelve a cargar el valor de los registros Modbus en los registros temporales de reconocimiento.

Seguidamente se cargan los valores de las alarmas de tipo intermitente, de tipo activas en bajo o en alto y las alarmas habilitadas, en los registros temporales para estas tareas (T_stay, T_mask y T_wish). Además, se carga el valor del status del PIC maestro en la variable temporal estado. Luego se chequean los registros de alarmas, y en caso de existir alguna, se activa el bit correspondiente en la variable estado; además, se chequea si las alarmas que están activas no están reconocidas, de ser así, se activa el bit correspondiente a una alarma activa sin reconocer en el byte estado. Leyendo el byte de status del PIC esclavo se puede conocer si existe alguna alarma en la planta, y activar el bit correspondiente en el byte de estado del maestro. También se puede conocer si la planta en el sitio responde o no a las peticiones del esclavo y activar o limpiar el bit correspondiente en estado. Y en caso de existir algún error en la tarjeta electrónica se activa la bandera correspondiente en estado. Por último, se carga el valor de la variable estado en el registro Modbus correspondiente (REGS[status]).

Las rutinas de chequeo de alarmas digitales y analógicas, y la lectura y escritura en el PIC esclavo se repiten indefinidamente, con un delay de 10 ms entre una vuelta y otra.

6.1.4 Programa del microcontrolador esclavo

Al igual que el programa del microcontrolador maestro, en el esclavo todas las rutinas se desarrollan en C. En la tabla 6.4 se presentan las librerías precompiladas y las rutinas de las mismas utilizadas en el programa del PIC maestro.

Tabla 6.4 Librerías precompiladas en lenguaje C utilizadas en el programa del PIC esclavo.

Librería	Rutinas Utilizadas	Descripción
Usart.h	OpenUsart	Realiza la configuración del módulo USART en el modo y la forma deseada.
Timers.h	OpenTimer0	Configuración del timer 0 en el modo y la forma deseada.
	OpenTimer1	Configuración del timer 1 en el modo y la forma deseada.
Delays.h	Delay10KTCYx	Aplica un delay de x veces 10000 ciclos internos del microcontrolador.
Adc.h	OpenADC	Configura el módulo convertidor analógico-digital en la forma deseada.

El microcontrolador esclavo monitorea las señales analógicas, pregunta a la planta si existe alguna alarma en la misma y reenvía paquetes Modbus del sistema principal en el CENCE a la planta de emergencia; además, debe comunicar al PIC maestro el estado de las alarmas que monitorea, así como preguntar al mismo sobre las alarmas que se deben monitorear. En forma similar al microcontrolador maestro, en el PIC esclavo se implementa un arreglo llamado REGS2 de 50 elementos de 8 bits cada uno, en donde se almacena la información de las alarmas monitoreadas en este microcontrolador. En el apéndice A.3 se presenta una descripción detallada de la información que contienen estos registros.

En el programa principal del PIC esclavo primero se realiza la inicialización de las variables y los módulos periféricos, y luego se llaman a las rutinas que monitorean las alarmas, y la rutina que actualiza las mismas en el arreglo REGS2. La figura 6.35 presenta el diagrama de flujo del programa principal del PIC esclavo.

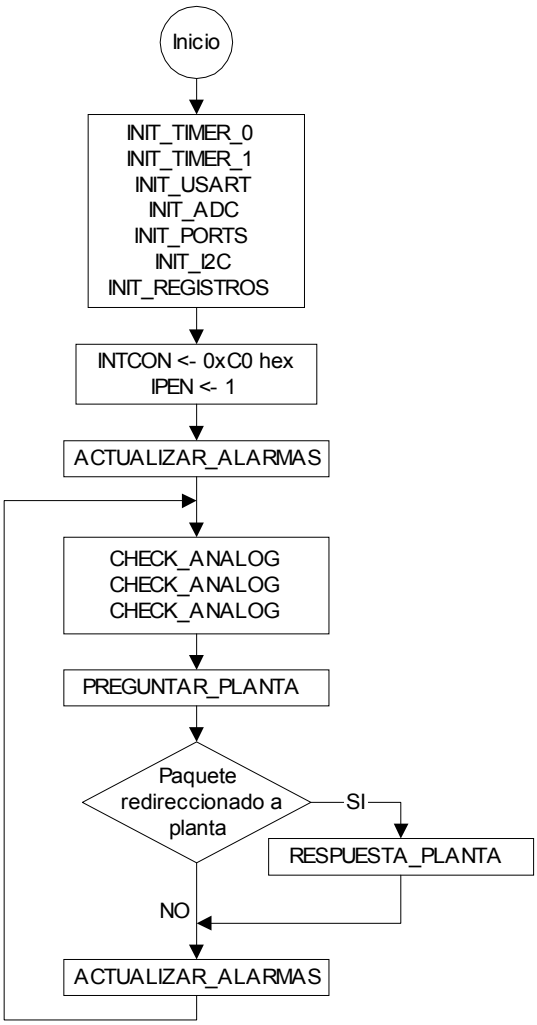


Figura 6.35 Diagrama de flujo del programa principal del microcontrolador esclavo.

Los primeros módulos que se inicializan son los timers 0 y 1, los cuales se configuran de la misma forma que en el PIC maestro, tal como se muestra en las figuras 6.12 y 6.13. Así mismo, la USART del PIC esclavo se configura igual que microcontrolador maestro, como se muestra en la figura 6.14, en modo asincrónico, con 8 bits de datos, recepción continua y a 9600 bps, y se habilita la interrupción de recepción del puerto serie y se le da baja prioridad.

El ADC se configura con 5 canales analógicos y una entrada de referencia (5 voltios), los 10 bits del resultado se justifican a la izquierda, para tomar únicamente los 8 bits más significativos. El puerto A se configura como entradas analógicas

(TRISA = 0xFF hex). En la figura 6.36 se presenta el diagrama de flujo de la rutina que inicializa el ADC.

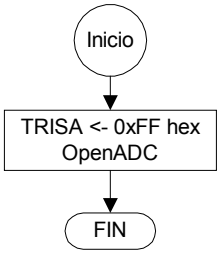


Figura 6.36 Diagrama de flujo de la rutina de inicialización del ADC.

El módulo MSSP se configura en modo I^2C, de la forma que se muestra en la figura 6.37, los pines 3 y 4 del puerto C (RC3 y RC4) se habilitan como entradas, el registro SSPADD se carga con la constante I2C_DIR, la cual contiene la dirección I2C del esclavo. Se habilitan y se le da alta prioridad a las interrupciones de colisión de bus (BCLIP y BCLIP) y de actividad en el módulo I2C (SSPIE y SSPIP), luego se configura el módulo MSSP como esclavo y con direccionamiento de 7 bits (SSPCON1 = 0b00110110).

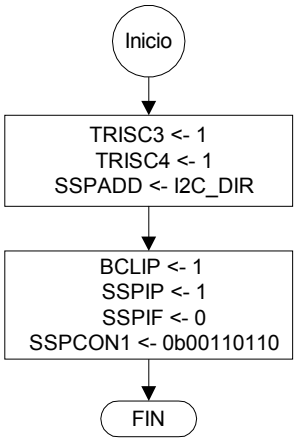


Figura 6.37 Diagrama de flujo de la rutina de inicialización del módulo I^2C.

Después de la inicialización el programa principal se cicla en la revisión del las señales analógicas (rutina CHECK_ANALOG), y preguntando a la planta (en caso de existir una) si tiene alguna alarma (rutina PREGUNTAR_PLANTA). Y cuando se halla reenviado un paquete del sistema principal a la planta, se hace necesario esperar la

respuesta de la misma, con la rutina RESPUESTA_PLANTA, para enviarla al PIC maestro.

Cuando se presenta una interrupción de baja prioridad, el hardware del PIC realiza un direccionamiento del flujo del programa a la rutina Interrupcion_L. En la figura 6.38 se presenta el diagrama de flujo de la rutina Interrupcion_L. En dicha rutina se chequea que el llamado a interrupción se halla dado por la recepción de un carácter en la USART, para ello las banderas de habilitación de la interrupción de recepción de la USART (RCIE), y la bandera de interrupción de recepción en la USART (RCIF) deben estar en uno. Para procesar el carácter recibido se llama a la rutina INT_USART_RX. Pero si las banderas de habilitación de la interrupción de transmisión de la USART (TXIE), y la bandera de interrupción de recepción en la USART (TXIF) están en uno, entonces el llamado ha interrupción se dio porque la USART está lista para transmitir otro carácter, y entonces se llama a la rutina INT_USART_TX.

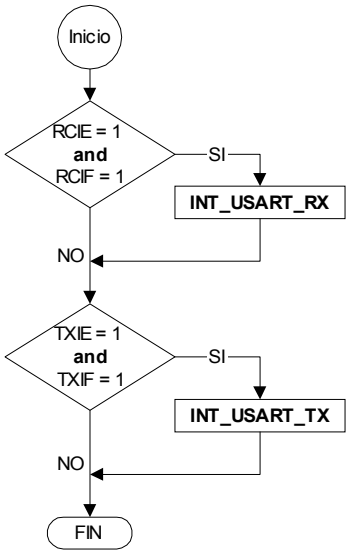


Figura 6.38 Diagrama de flujo de la rutina de interrupción de baja prioridad.

Cuando la interrupción que se presenta es de alta prioridad, el flujo del programa se direcciona a la rutina Interrupcion_H. En la figura 6.39 se muestra el diagrama de flujo de la rutina de atención de interrupción de alta prioridad, en la cual se prueba si la bandera de habilitación de interrupción del módulo MSSP y la

bandera de interrupción del mismo están activas, de ser así, se llama a la rutina i2c_read_write.

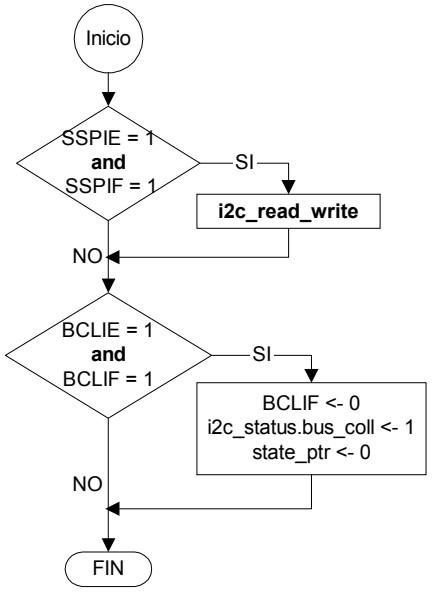


Figura 6.39 Diagrama de flujo de la rutina de interrupción de alta prioridad.

Cuando la interrupción se presenta por una colisión de bus en el módulo I2C, las banderas BCLIE y BCLIF están activas, se hace necesario limpiar la bandera de interrupción y activar el bit de error por colisión de bus en el byte i2c_status.bus_coll.

Cuando el microcontrolador esclavo se comunica con la planta de emergencia, el PIC esclavo se debe implementar como Modbus maestro, ya que es él quien hace las peticiones a la planta. En este microcontrolador se desarrolla una rutina que implemente paquetes Modbus, o ‘’queries’’; en la figura 6.40 se presenta un diagrama de flujo de dicha rutina. La cual, utiliza un registro llamado ModPack para crear el paquete Modbus respectivo, donde la variable ModIndex contiene el número de elementos en ModPack. Primero se limpia el contenido de ModPack (ModIndex = 0), y se le agrega la dirección del esclavo al que se dirige el paquete, y luego se agrega el código de función que se va a implementar. Las funciones que se implementaron son las 03 y 16, las cuales tienen en común los campos de dirección inicial y el número de registros. Si la función a implementar es la 03, se realiza el cálculo del CRC para terminar la creación del paquete.

Indice

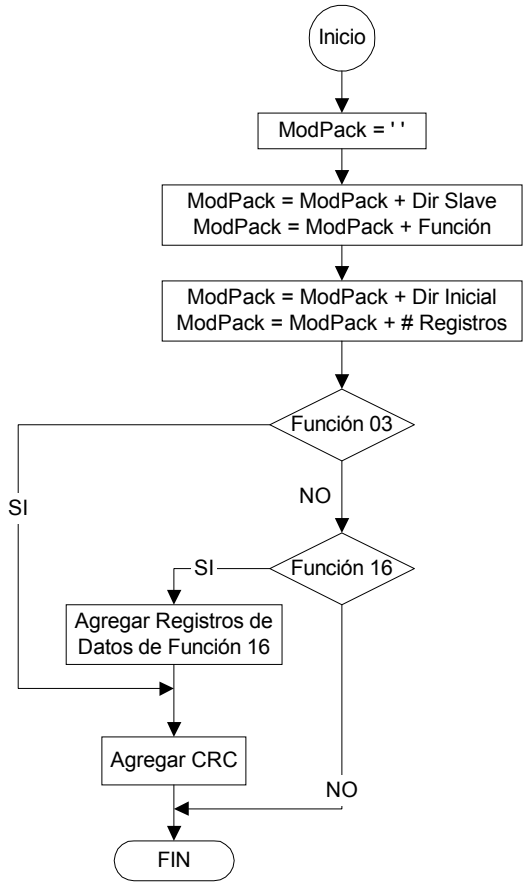


Figura 6.40 Diagrama de flujo de la rutina que implementa paquetes Modbus de petición.

Por otra parte, si la función es la 16, se tiene que agregar un campo de Byte Count, que indica el número de bytes de datos que se desean escribir en el esclavo en los registros indicados por la dirección inicial y el número de registros.

La rutina mostrada en el diagrama de flujo de la figura 6.33, se utiliza para crear paquetes con la función 03 principalmente, esto para leer registros de la planta y determinar si existe alguna alarma en la misma. La función 16 se utiliza principalmente para configurar la planta, escribiendo a los registros de la misma, pero los paquetes son generados por el sistema principal maestro.

Una vez generado un ‘’query’’, es necesario enviarlo por el puerto serie, y después de la transmisión se debe esperar la respuesta del esclavo, y de esta forma determinar y existió algún error de comunicación, o si la petición fue realizada. Para realizar transmisiones por el puerto serie se hace uso de las rutinas Envia_Pack y

INT_USART_TX, las cuales cargan el arreglo a enviar en puntero que se transmite e incrementa cada vez que se presente la interrupción del puerto serie de transmisión; estas rutinas funcionan de la misma forma que en el PIC maestro, como se muestra en las figuras 6.24 y 6.25.

Para manejar las respuestas de la planta a los paquetes de petición, se llama a la rutina INT_USART_RX cada vez que se recibe un carácter por el puerto serie. En la figura 6.41 se presenta el diagrama de dicha rutina.

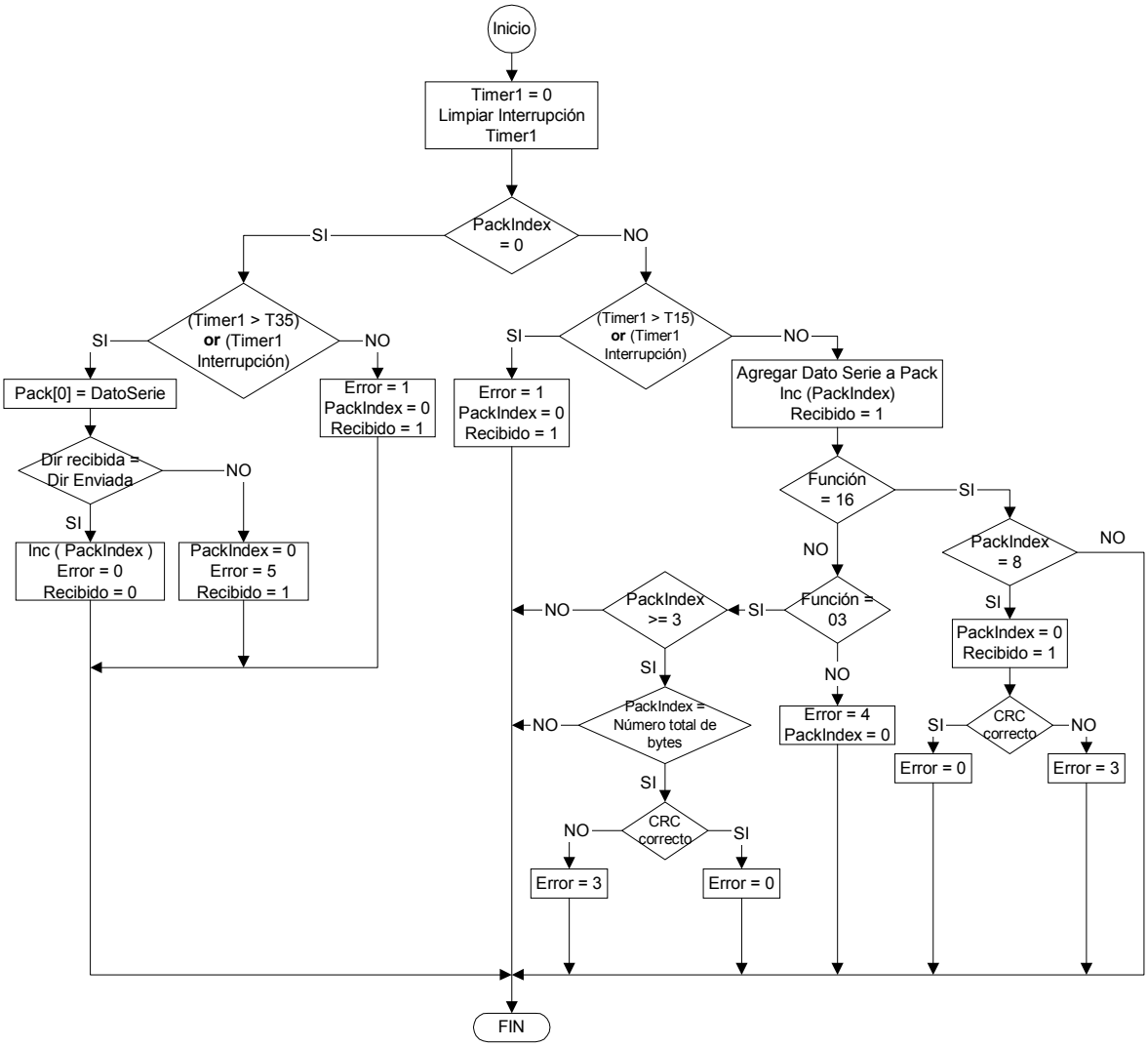


Figura 6.41 Diagrama de flujo de la rutina INT_USART_TX en el PIC esclavo.

El timer1 se utiliza, para controlar la temporización modbus en los paquetes de respuesta que se reciban. Existe la variable global Recibido, que le indica al programa principal cuando se recibió la respuesta al ‘’query’’ que se envió previamente.

Cuando se recibe un carácter en el puerto serie, se llama a la rutina de interrupción, se reinicia el conteo del timer1, y se limpia la bandera de interrupción del puerto serie, luego se verifica si se trata del primer byte de la respuesta, de ser así, se determina si el byte cumplió con la temporización indicada por el protocolo modbus, si no lo cumplió, se le indica al programa principal por medio de las variables Recibido y Error, que la respuesta llegó pero con error en la temporización. Si el primer byte llega correctamente, se compara el valor del byte con la dirección del esclavo del cual se espera la respuesta, si son diferentes se genera el error respectivo, si son iguales, se agrega el byte al arreglo Pack.

Si el carácter recibido no es el primero del paquete de respuesta, se verifica que el tiempo de llegada entre bytes, no sea superior al tiempo de transmisión de 1.5 caracteres (T15), si no se cumple esta condición se genera el error respectivo y se le comunica al programa principal que se recibió el paquete con este error. Cuando la temporización es correcta, se determina si el código de función recibido es el 03, 06, o alguno de los anteriores pero con un 1 en el bit más significativo.

Si el código de función recibido es el 03, se espera hasta que el número de elementos del paquete recibido sea igual al número de bytes indicados en el campo 3 de la respuesta (Byte Count) más los bytes de dirección, función y CRC. Cuando se halla completado el paquete se verifica el CRC, si existe un error, se le comunica al programa principal la existencia del mismo, y que el paquete se recibió. Si no hay error en el CRC, en el paquete recibido se encuentran los valores de los registros que se leyeron en el esclavo.

Cuando la función recibida es la 16, se espera hasta que el número de elementos recibidos sea igual a 8, ya que la respuesta a un ‘’query’’ con la función 16 cuenta con este número de elementos. Cuando el paquete está completo, se

chequea el valor del CRC, si existe algún error, se le indica al programa principal el mismo y que el paquete ya se recibió. Cuando se recibe el paquete sin errores, se comparan los valores de dirección inicial y número de registros recibidos, con los valores que se transmitieron en el ‘’query’’, si son distintos, se debe generar el error respectivo al programa principal, si el paquete es correcto, no se generan errores y se le indica que ya se recibió la respuesta.

Además de los valores de códigos de función ya descritos, se pueden recibir respuestas de excepción, donde el campo de la función tiene el valor de la función incluida en el ‘’query’’, pero con el bit más significativo en uno. Sin importar cual función estaba en el ‘’query’’, solamente se espera hasta que la longitud del paquete sea de 5, y únicamente se le chequea el CRC, si esta bien, se le indica al programa principal que se recibió la respuesta, y que es una respuesta de excepción. Cuando se recibe algún otro valor en el campo del código de función, se genera un error que se comunica al programa principal.

El programa principal debe chequear las variables Recibido y Error, para determinar cuando se recibe el paquete de respuesta, y si el mismo se recibió con o sin errores, ya que dependiendo de esta información se deben tomar las medidas necesarias.

Como se presentó en la figura 6.35 el monitoreo de las señales analógicas se realiza por medio de la rutina CHECK_ANALOG. Al igual que en el monitoreo de las señales digitales, se deben considerar las alarmas habilitadas y activas. El diagrama de flujo de la rutina Check_analog se presenta en la figura 6.42. En esta rutina primero se cargan los registros de mascara y alarmas, para la señal analógica o canal que se evalúe en el momento. Se verifica que al menos una de las 4 alarmas por señal analógica este habilitada, de ser así, se realiza la conversión analógica digital respectiva, llamando a la rutina Promediar, y se almacena el valor en la variable conversión y en el registro respectivo (REGS2[Val_An_X]). Luego se chequea bit por bit la alarma correspondiente; dichas alarmas se activa cuando la señal analógica supera alguno de los límites superiores, o bien, está por debajo de

alguno de los límites inferiores. Estos límites se encuentran almacenados en los registros internos del PIC esclavo. Cada vez que se halla presentado una alarma analógica se incrementa un contador de eventos, y el bit correspondiente a la alarma se activa cuando el contador de eventos sea igual a la constante Num_Event. Para desactivar la alarma, el contador de eventos también debe tener un conteo de Num_Event. De tal forma que para activar o desactivar la alarma correspondiente se deben dar un número de eventos de NUM_Event en forma continua, ya que si no hay continuidad entre un a muestra y otra el contador de eventos se reinicia.

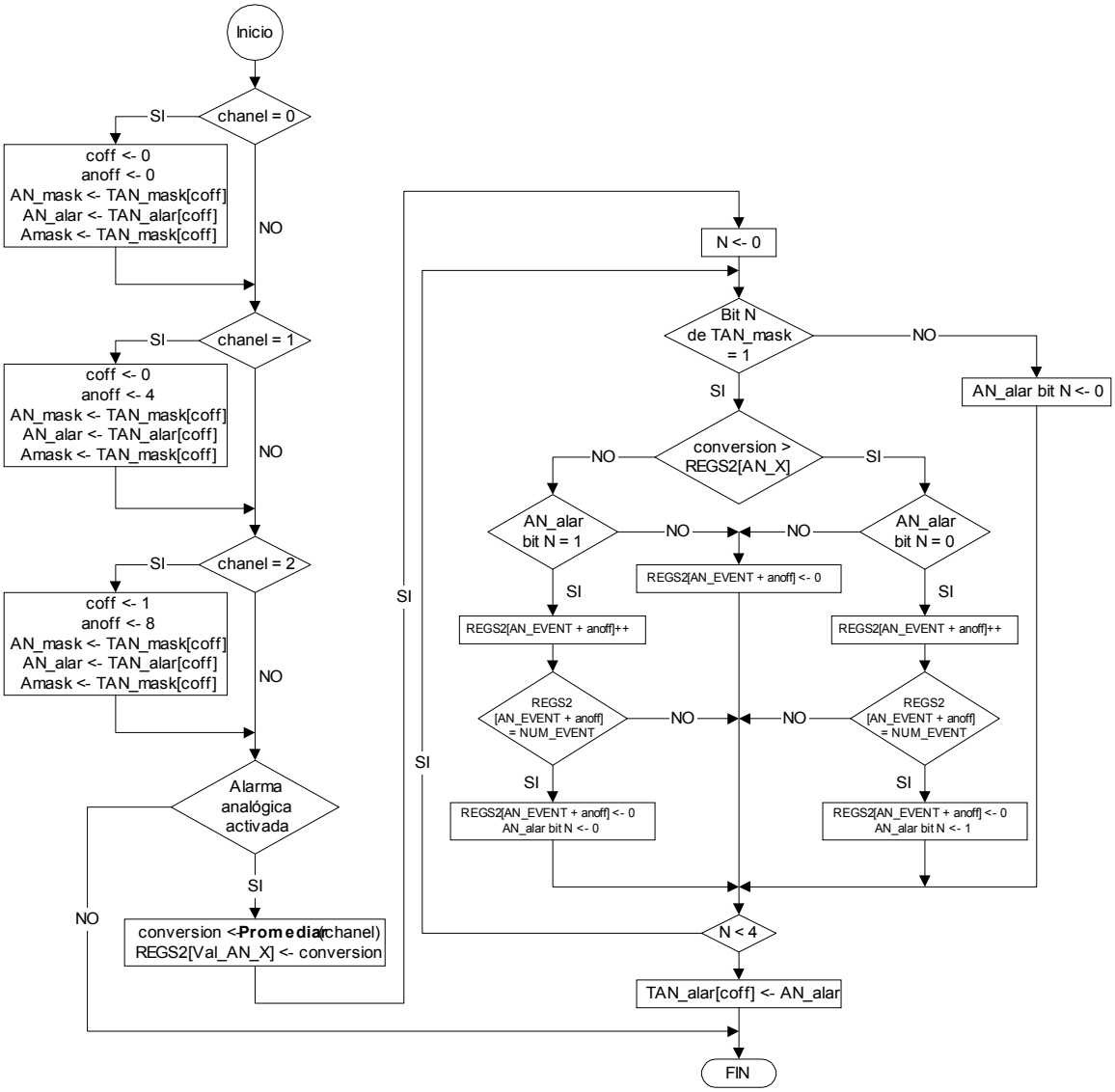


Figura 6.42 Diagrama de flujo de la rutina CHECK_ANALOG.

Indice

La rutina que se encarga de realizar la conversión analógica digital es la rutina Promediar, el diagrama de flujo de la misma se presenta en la figura 6.43.

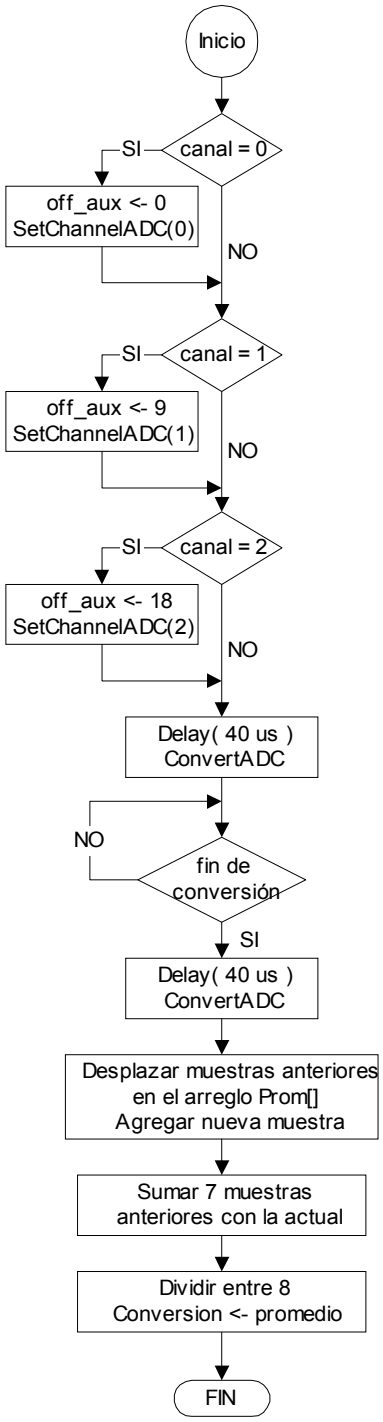


Figura 6.43 Diagrama de flujo de la rutina Promediar.

Indice

En esta rutina además se realiza un promedio del valor obtenido en la conversión, con los 7 valores anteriores muestreados. Primero se selecciona el canal de entrada a convertir, por medio de la rutina precompilada SetChannelADC(). Luego se da un delay de 40 μs para el respectivo tiempo de adquisición del ADC del microcontrolador; y se llama a la rutina precompilada ConvertADC(), que activa la conversión de analógica a digital en el ADC para el canal seleccionado. Seguidamente se chequea cuando se presenta el final de la conversión, para cargar el valor de los 8 bits más significativos en la variable conversion. La nueva muestra de la señal analógica es ingresada en el arreglo Prom, el cual se maneja como un stack, y se realiza un desplazamiento de las 7 muestras anteriores de la señal, que se almacenaron previamente en dicho arreglo. Luego se realiza una sumatoria de los 8 elementos en Prom, y finalmente se realiza una división entre 8 para promediar las muestras de la señal analógica. Este promedio funciona como un pequeño filtro digital, que elimina muestras incorrectas y mantiene la señal discreta más estable.

Esta rutina es llamada dentro de Check_Analog en donde el valor de la conversión y de las alarmas respectivas se almacena en los registros internos del microcontrolador esclavo. Y es obligación del PIC maestro leer estos registros para conocer estos valores, así como escribir en los mismos la información de cuales alarmas están habilitadas y cuales son los umbrales de las alarmas. Para que el PIC esclavo permita tales operaciones, se deben implementar en el mismo las rutinas necesarias para que funcione como un esclavo en el protocolo I^2C. Por ello se habilita la interrupción por eventos en el módulo MSSP, el cual fue inicializado en modo I^2C. De tal forma que cuando se presenta algún evento en el bus, se genera una interrupción, y la bandera de habilitación del módulo MSSP y la bandera de interrupción del módulo MSSP están activas; cuando esta situación ocurre se llama a la rutina I2C_Read_Write. El diagrama de flujo de esta rutina se presenta en la figura 6.44. En ella se chequea el estado de las banderas: start (S), read/write (R_W), data/address (D_A) y buffer full (BF); que corresponden a los bits 0, 2, 3 y 5 en el registro SSPSTAT. Enmascarando estos bits y guardando el resultado en la variable stat, se puede conocer en que estado I^2C se encuentra el esclavo.

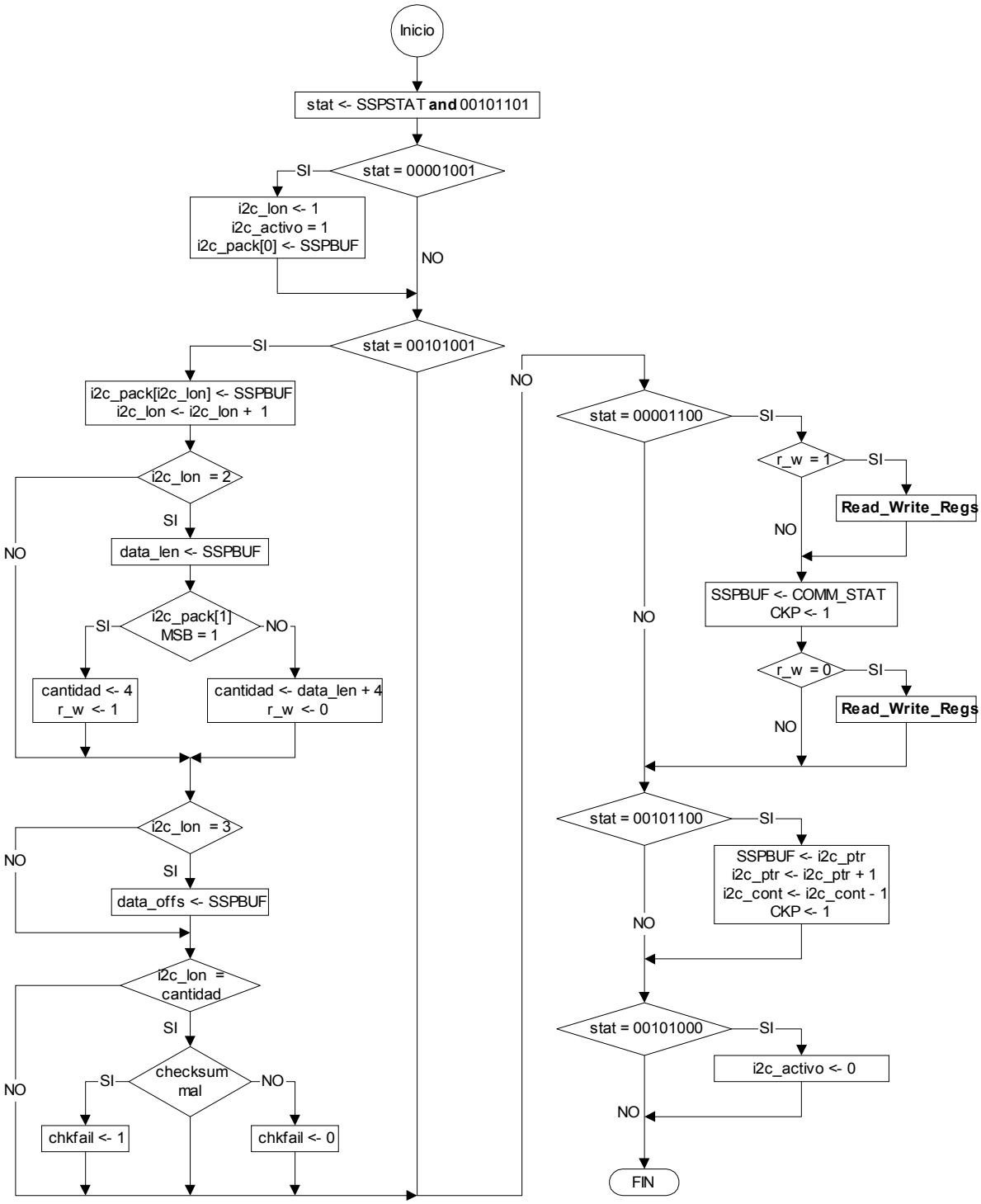


Figura 6.44 Diagrama de flujo de la rutina I2C_Read_Write.

En el primer estado, S = 1 y BF = 1, se realiza una operación de escritura, y se recibe la dirección del esclavo. Dicha dirección se agrega al arreglo i2c_pack y el

valor de i2c_lon se pone en 1. Además se pone en 1 la bandera de actividad en el bus I^2C.

En el segundo estado las banderas S, D_A y BF están en 1, lo que indica que se está ejecutando una operación de escritura y que el byte que se encuentra en el registro de recepción (SSPBUF) es un dato. El carácter recibido se agrega al arreglo i2c_pack, y se incrementa el valor de i2c_lon. Luego si el carácter recibido es el tercero del paquete, este byte corresponde al data_len del protocolo I^2C, se chequea el MSB para determinar si es un paquete de lectura o escritura, y de esta forma colocar el valor en la variable cantidad correspondiente al número de caracteres que se deben recibir. Si el carácter recibido es el cuarto en i2c_pack, este corresponde al data_offs del protocolo I^2C, y por ello se debe actualizar el valor del mismo. Y finalmente, cuando se han recibido todos los caracteres del paquete (i2c_lon = cantidad) se realiza el cálculo del checksum y se compara con el valor recibido, para generar el error correspondiente en caso de ser necesario.

Cuando las banderas S y R_W están en uno el esclavo se encuentra en el estado 3, en el cual se está realizando una operación de lectura y el byte en SSPBUF es la dirección del esclavo. En este estado si el paquete completo es de lectura, la primera acción que se realiza es un llamado a la rutina Read_Write_Regs, luego el esclavo carga el byte Comm_Stat en el registro SSPBUF y seguidamente activa la bandera CKP, para transmitir dicho byte al maestro. La acción que se realiza es cuando el paquete es de escritura, y se llama a la rutina Read_Write_Regs. Esta última rutina realiza acciones distintas cuando el paquete es de lectura o escritura. Si el maestro desea leer registros, la rutina carga en el arreglo i2c_pack los datos pedidos por el maestro, pero si la acción es para escribir en los registros internos del esclavo, entonces la rutina Read_Write_Regs captura los datos que envía el maestro en el arreglo i2c_pack.

En el estado 4 están activas las banderas S, R_W y D_A; indicando que se esta realizando una operación de lectura, y que el dispositivo esclavo esta listo para transmitir otro dato. Para ello el esclavo carga el valor apuntado por i2c_ptr en el

registro SSPBUF, y habilita la transmisión activando la bandera CKP, además, incrementa el puntero al siguiente elemento ha transmitir, y por último decrementa el conteo del número de elementos que tiene que enviar.

En el último estado, están activas las banderas S y D_A, e indican que se recibió un NACK por parte del maestro, y por lo tanto que la transacción por el bus I^2C ha terminado.

En cuanto a la rutina que reconstruye el paquete I^2C o bien que lo crea en el esclavo (ReadWrite_Regs), se presenta en la figura 6.45.

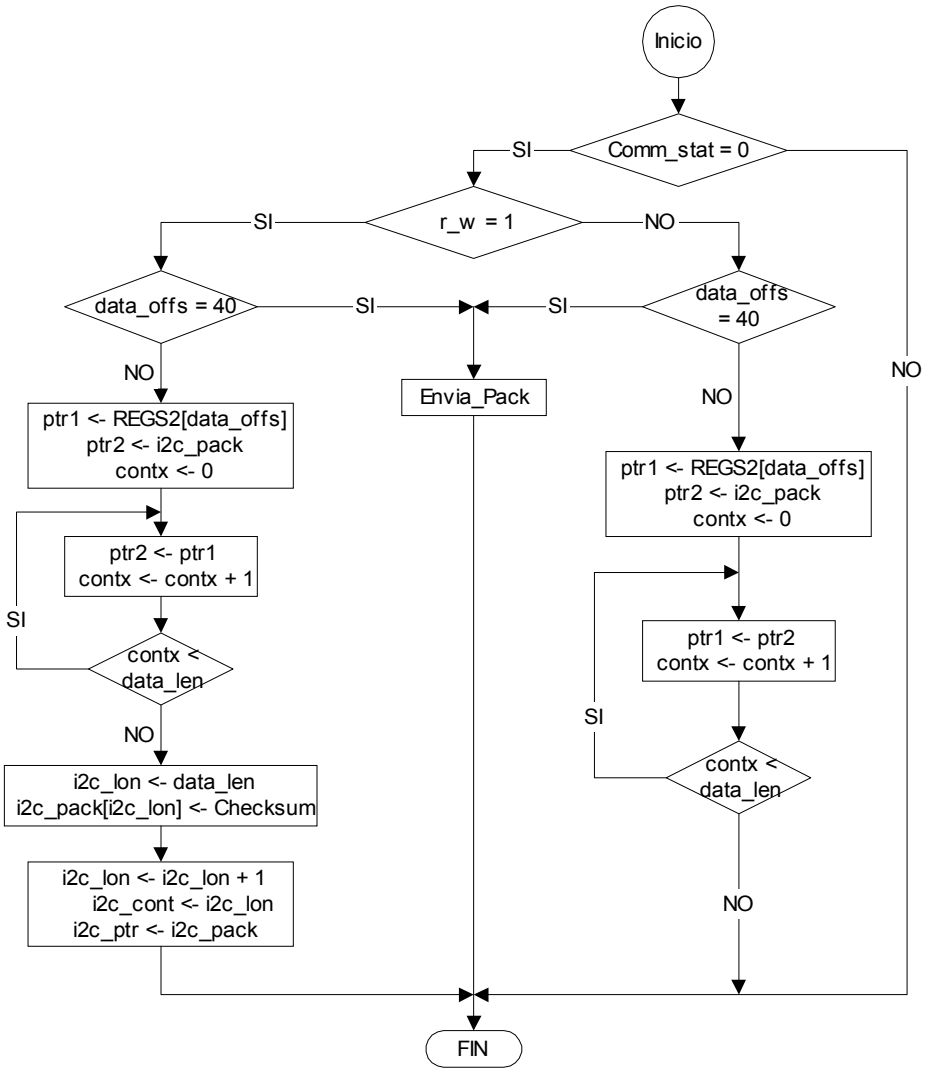


Figura 6.45 Diagrama de flujo de la rutina I2C_Read_Write.

Indice

En la rutina de la figura anterior, primero se chequea que no exista ningún error en paquete recibido de acuerdo al protocolo I^2C. Luego se determina si la acción que se va a realizar es de lectura o escritura. Si el paquete es de lectura, se deben cargar los valores almacenados en los registros internos en el arreglo i2c_pack, la carga se realiza a partir del registro data_offs con un total de data_len registros. Se debe calcular el checksum y colocarlo al final del paquete, i2c_lon contiene el número de elementos en i2c_pack. El puntero i2c_ptr apunta al primer elemento en i2c_pack y i2c_cont tiene el mismo valor que i2c_lon. Cuando se transmite un dato el apuntador se incrementa y i2c_cont se decrementa. Cuando la transacción que se recibe es de escritura, la rutina Read_Write_Regs es llamada cuando se han recibido todos los datos ha ser escritos en los registros internos. Y entonces se realiza la escritura de los datos en el arreglo i2c_pack al arreglo REGS.

Si se recibe un paquete que contiene 40 como data_offs, los bytes de datos incluidos en el paquete se deben retransmitir a la planta del sitio a través del puerto serie, llamando a la rutina Envia_Pack.

Volviendo al programa principal de la figura 6.35 la última rutina que se realiza es la actualización de los valores en los registros internos del PIC esclavo, para ello se llama a la rutina Actualizar_Alarmas, y el diagrama de flujo de la misma se presenta en la figura 6.46.

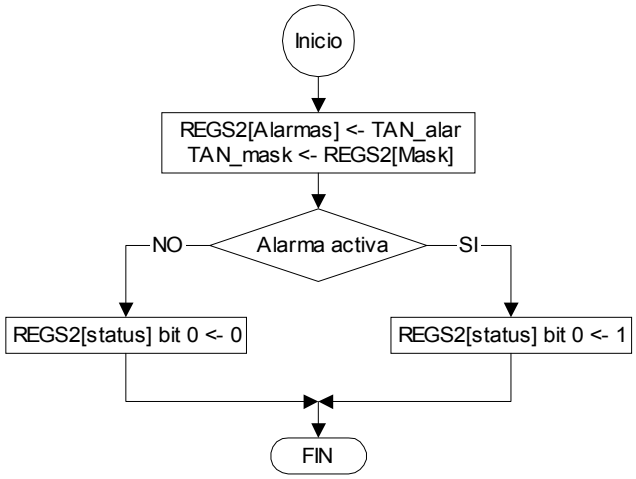


Figura 6.46 Diagrama de flujo de la rutina Actualizar_Alarmas.

Indice

En esta rutina se carga el valor del los registros temporales de alarmas (TAN_alar) en los registros internos del microcontrolador esclavo (REGS2[alasrmas]), y por último, si existe alguna alarma activa se pone en uno el bit 0 del registro de status (REGS2[status]).

6.2 Alcances y limitaciones

Una modificación importante al planteamiento inicial del proyecto se presentó al incorporarse una planta de emergencia a los equipos presentes en los sitios repetidores. Dicha planta comunica alarmas, fallas y datos generales de la misma por medio de un puerto serie por protocolo Modbus. La inclusión de las plantas en los equipos de los repetidores, se dio sobre la marcha del proyecto, provocando un retraso en el desarrollo del mismo ya que resulta de suma importancia considerar las nuevas condiciones como parte del proyecto. Al considerar las nuevas necesidades del sistema que se construyó, el nivel de dificultad en la solución del mismo aumentó, implicando una disminución en los alcances del proyecto.

Se presentaron graves problemas en la adquisición del equipo necesario, ya que el distribuidor tardó aproximadamente 8 semanas en poner a disposición el equipo, cuando se tenía estimado 3 semanas para dicha actividad.

Todas las rutinas que realizan las tareas básicas de la tarjeta electrónica están listas y probadas, estas tareas son: monitoreo de 16 señales digitales, monitoreo de 3 señales analógicas, comunicación Modbus con el sistema principal en el CENCE, comunicación Modbus con planta de emergencia, y comunicación entre los dos microcontroladores. Quedó por fuera de los alcances de este proyecto, el almacenamiento de datos en memoria no volátil.

La integración de las rutinas mencionadas se dio en forma parcial, el monitoreo de señales analógicas y digitales, junto con la comunicación Modbus con el sistema principal, y la comunicación entre microcontroladores se logró agrupar. No se contó con el tiempo necesario para integrar la comunicación con la planta de emergencia.

A pesar de que la mayoría de las rutinas se encuentran agrupadas, las pruebas al grupo de rutinas son muy pocas, por ello es recomendable realizar más pruebas y depurar las rutinas desarrolladas.

CAPÍTULO 7
CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

7.1.1 El protocolo de comunicación Modbus brinda al sistema en desarrollo una amplia compatibilidad con otros sistemas.

7.1.2 El protocolo I^2C brinda un medio de comunicación serial entre IC's confiable y de alta velocidad.

7.1.3 Se implementan las funciones de lectura y escritura Modbus, funciones 03 y 16 respectivamente, en al tarjeta electrónica.

7.1.4 Falta integrar la rutina de reenvío de paquetes Modbus del Centro de Control a la planta y la rutina de almacenamiento de información en memoria no volátil, para finalizar el hardware requerido en el proyecto de comunicación de alarmas remotas.

7.2 Recomendaciones

7.2.1 Para el desarrollo de aplicaciones con los microcontroladores de la familia Microchip®, es conveniente la utilización del compilador de C que suministra el fabricante.

7.2.2 Es conveniente cuidar los posibles conflictos de interrupciones, cuando se realice la integración de todas las rutinas diseñadas.

7.2.3 Es recomendable implementar nuevas funciones Modbus en el sistema, como las funciones 05, 22 y 23, que fuerzan bits individuales en un registro, modifican el contenido de un registro por medio de una mascara and y or y realizan una lectura y escritura en una sola transacción respectivamente, esto para darle mayor versatilidad al sistema.

7.2.4 En la implementación de almacenamiento de información en memoria no volátil, se recomiendan dos opciones:

- Microcontroladores PIC18F252 con memoria de programa flash: y se crea una tabla de almacenamiento en la memoria de programa.
- Memoria EEPROM serial de la familia de Microchip®: la cual se puede leer y escribir por medio del protocolo I^2C, que ya está implementado en el prototipo.

Bibliografía

- Becerra S, César. Lenguaje C. Computador Ltda, 1987.
- Microchip®. PIC18CXX2 Data sheet.
- Microchip®. MPLAB® -CXX Compiler user's guide.
- Microchip®. MPLAB®-CXX Reference guide libraries and precompiled object files.
- Microchip®. An I^2C™ Network Protocol for Environmental Monitoring.
- Microchip®. Using the PICmicro® MSSP for master I^2C™ communication.
- Microchip®. Using the PICmicro® SSP for slave I^2C™ communication.
- Modicon Modbus protocol reference guide.

Indice

APÉNDICES Y ANEXOS

Apéndice A.1: Abreviaturas.

ADC:
Analog to Digital Converter (Convertidor analógico digital).

CENCE:
Centro Nacional de Control de Energía.

EPROM:
Erasable Programmable Read Only Memory (Memoria de solo lectura programable y borrable).

I^2C:
Inter-Integrated Circuit.

ICE:
Instituto Costarricense de Electricidad.

ICELEC:
Sector ICE-Eléctrico.

MSSP:
Master Synchronous Serial Port (Puerto serie síncrono maestro).

PIC:
Microcontrolador de la familia de Microchip®, en este documento se refiere al PIC18C252.

RAM:
Ramdom access memory (Memoria de acceso aleatorio).

SEN:
Sistema eléctrico nacional.

Indice

SPI:

Serial port interface (interfaz de puerto serial).

UEN:

Unidad estratégica de negocio.

Apéndice A.2: Registros Modbus del PIC maestro.

Tabla A.2.1 Descripción del contenido de los registros Modbus 1 al 8 del PIC maestro

Registro Modbus	Contenido
1	- Byte bajo: Byte de control para habilitar la existencia de una planta en el sitio - Byte alto: Dirección Modbus de la tarjeta electrónica
2	- Byte alto: valor analógico que causa alarma por máximo 1 en señal 1 - Byte bajo: valor analógico que causa alarma por máximo 2 en señal 1
3	- Byte alto: valor analógico que causa alarma por mínimo 1 en señal 1 - Byte bajo: valor analógico que causa alarma por mínimo 2 en señal 1
4	- Byte alto: valor analógico que causa alarma por máximo 1 en señal 2 - Byte bajo: valor analógico que causa alarma por máximo 2 en señal 2
5	- Byte alto: valor analógico que causa alarma por mínimo 1 en señal 2 - Byte bajo: valor analógico que causa alarma por mínimo 2 en señal 2
6	- Byte alto: valor analógico que causa alarma por máximo 1 en señal 3 - Byte bajo: valor analógico que causa alarma por máximo 2 en señal 3
7	- Byte alto: valor analógico que causa alarma por mínimo 1 en señal 3 - Byte bajo: valor analógico que causa alarma por mínimo 2 en señal 3

8	- Byte alto: Máscara de habilitación de cada una de las 8 alarmas del bloque 1.
	  - Bit 0 = 1: Alarma del pin 1 habilitada.
	  - Bit 1 = 1: Alarma del pin 2 habilitada.
	  - Bit 2 = 1: Alarma del pin 3 habilitada.
	  - Bit 3 = 1: Alarma del pin 4 habilitada.
	  - Bit 4 = 1: Alarma del pin 5 habilitada.
	  - Bit 5 = 1: Alarma del pin 6 habilitada.
	  - Bit 6 = 1: Alarma del pin 7 habilitada.
	  - Bit 7 = 1: Alarma del pin 8 habilitada.
	- Byte bajo: Máscara de habilitación de cada una de las 8 alarmas del bloque 2.
	  - Bit 0 = 1: Alarma del pin 9 habilitada.
	  - Bit 1 = 1: Alarma del pin 10 habilitada.
	  - Bit 2 = 1: Alarma del pin 11 habilitada.
	  - Bit 3 = 1: Alarma del pin 12 habilitada.
	  - Bit 4 = 1: Alarma del pin 13 habilitada.
	  - Bit 5 = 1: Alarma del pin 14 habilitada.
	  - Bit 6 = 1: Alarma del pin 15 habilitada.
	  - Bit 7 = 1: Alarma del pin 16 habilitada.

Tabla A.2.2 Descripción del contenido de los registros Modbus 9 al 13 del PIC maestro

Registro Modbus	Contenido
9	- Byte alto: Máscara de habilitación de cada una de las 8 alarmas del bloque 3. Cada bit representa la alarma correspondiente. - Byte bajo: Máscara de habilitación de cada una de las 8 alarmas del bloque 4. Cada bit representa la alarma correspondiente.

10	- Byte alto: Máscara de habilitación de las 4 alarmas de las señales analógicas 1 y 2.
	  - Bit 0 = 1, máximo 1 de la señal analógica 1 habilitado.
	  - Bit 1 = 1, máximo 2 de la señal analógica 1 habilitado.
	  - Bit 2 = 1, mínimo 1 de la señal analógica 1 habilitado.
	  - Bit 3 = 1, mínimo 2 de la señal analógica 1 habilitado.
	  - Bit 4 = 1, máximo 1 de la señal analógica 2 habilitado.
	  - Bit 5 = 1, máximo 2 de la señal analógica 2 habilitado.
	  - Bit 6 = 1, mínimo 1 de la señal analógica 2 habilitado.
	  - Bit 7 = 1, mínimo 2 de la señal analógica 2 habilitado.
	- Byte bajo: Máscara de habilitación de las 4 alarmas de la señal analógica 3.
	  - Bit 0 = 1, máximo 1 de la señal analógica 3 habilitado.
	  - Bit 1 = 1, máximo 2 de la señal analógica 3 habilitado.
	  - Bit 2 = 1, mínimo 1 de la señal analógica 3 habilitado.
	  - Bit 3 = 1, mínimo 2 de la señal analógica 3 habilitado.

11	- Byte alto: Alarmas reconocidas del bloque 1. Cada bit, cuando está en 1, reconoce la alarma correspondiente en el bloque 1. - Byte bajo: Alarmas reconocidas del bloque 2. Cada bit, cuando está en 1, reconoce la alarma correspondiente en el bloque 2.
12	- Byte alto: Alarmas reconocidas del bloque 3. Cada bit, cuando está en 1, reconoce la alarma correspondiente en el bloque 3. - Byte bajo: Alarmas reconocidas del bloque 4. Cada bit, cuando está en 1, reconoce la alarma correspondiente en el bloque 4.

13	- Byte alto: Alarmas reconocidas de las señales analógicas 1 y 2.
	  - Bit 0 = 1, reconocimiento del máximo 1 de la señal analógica 1.
	  - Bit 1 = 1, reconocimiento del máximo 2 de la señal analógica 1.
	  - Bit 2 = 1, reconocimiento del mínimo 1 de la señal analógica 1.
	  - Bit 3 = 1, reconocimiento del mínimo 2 de la señal analógica 1.
	  - Bit 4 = 1, reconocimiento del máximo 1 de la señal analógica 2.
	  - Bit 5 = 1, reconocimiento del máximo 2 de la señal analógica 2.
	  - Bit 6 = 1, reconocimiento del mínimo 1 de la señal analógica 2.
	  - Bit 7 = 1, reconocimiento del mínimo 2 de la señal analógica 2.
	- Byte bajo: Alarmas reconocidas de la señal analógica 3.
	  - Bit 0 = 1, reconocimiento del máximo 1 de la señal analógica 3.
	  - Bit 1 = 1, reconocimiento del máximo 2 de la señal analógica 3.
	  - Bit 2 = 1, reconocimiento del mínimo 1 de la señal analógica 3.
	  - Bit 3 = 1, reconocimiento del mínimo 2 de la señal analógica 3.

Tabla A.2.3 Descripción del contenido de los registros Modbus 14 al 18 del PIC maestro

Registro Modbus	Contenido

14	- Byte alto: alarma del bloque 1 normalmente en bajo o en alto; cada bit corresponde a cada alarma del bloque 1.
	  - Bit = 0, alarma normalmente en bajo.
	  - Bit = 1, alarma normalmente en alto.
	- Byte bajo: alarma del bloque 2 normalmente en bajo o en alto; cada bit corresponde a cada alarma del bloque 2.
	  - Bit = 0, alarma normalmente en bajo.
	  - Bit = 1, alarma normalmente en alto.

15	- Byte alto: alarma del bloque 3 normalmente en bajo o en alto; cada bit corresponde a cada alarma del bloque 3.
	  - Bit = 0, alarma normalmente en bajo.
	  - Bit = 1, alarma normalmente en alto.
	- Byte bajo: alarma del bloque 4 normalmente en bajo o en alto; cada bit corresponde a cada alarma del bloque 4.
	  - Bit = 0, alarma normalmente en bajo.
	  - Bit = 1, alarma normalmente en alto.

16	- Byte alto: alarma de tipo intermitente o mantenida; cada bit corresponde a cada alarma del bloque 1.
	  - Bit = 0, alarma intermitente.
	  - Bit = 1, alarma se mantiene una vez activada.
	- Byte bajo: alarma de tipo intermitente o mantenida; cada bit corresponde a cada alarma del bloque 2.
	  - Bit = 0, alarma intermitente.
	  - Bit = 1, alarma se mantiene una vez activada.

17	- Byte alto: alarma de tipo intermitente o mantenida; cada bit corresponde a cada alarma del bloque 3.
	  - Bit = 0, alarma intermitente.
	  - Bit = 1, alarma se mantiene una vez activada.
	- Byte bajo: alarma de tipo intermitente o mantenida; cada bit corresponde a cada alarma del bloque 4.
	  - Bit = 0, alarma intermitente.
	  - Bit = 1, alarma se mantiene una vez activada.

18	- Byte alto: alarma de tipo intermitente o mantenida; cada bit corresponde a cada alarma de la señal analógica 1 y 2.
	  - Bit = 0, alarma intermitente.
	  - Bit = 1, alarma se mantiene una vez activada.
	- Byte bajo: alarma de tipo intermitente o mantenida; cada bit corresponde a cada alarma de la señal analógica 3.
	  - Bit = 0, alarma intermitente.
	  - Bit = 1, alarma se mantiene una vez activada.

Tabla A.2.4 Descripción del contenido de los registros Modbus 19 al 38 del PIC maestro

Registro Modbus	Contenido

19	- Byte alto: Número de alarmas en el stack de alarmas.
	- Byte bajo: Byte de estado.
	  - Bit 0 = 1, alarma activa.
	  - Bit 1 = 1, alarma no reconocida activa.
	  - Bit 2 = 1, alarma activa en la planta.
	  - Bit 3 = 1, Error en la tarjeta electrónica.
	  - Bit 6 = 1, Planta no responde.
	  - Bit 7 = 1, Hay planta en el sitio

20	- Byte alto: Estado de las alarmas del bloque 1. Cada bit corresponde a una alarma, si está en 1 la alarma está activa (donde el bit 0 corresponde a la alarma 1, etc). - Byte bajo: Estado de las alarmas del bloque 2. Cada bit corresponde a una alarma, si está en 1 la alarma está activa (donde el bit 0 corresponde a la alarma 9, etc).
21	- Byte alto: Estado de las alarmas del bloque 3 (cada bit corresponde a una alarma). - Byte bajo: Estado de las alarmas del bloque 4 (cada bit corresponde a una alarma).

22	- Byte alto: Estado de las alarmas de las señales analógicas 1 y 2.
	  - Bit 0: alarma del máximo 1 de la señal analógica 1.
	  - Bit 1: alarma del máximo 2 de la señal analógica 1.
	  - Bit 2: alarma del mínimo 1 de la señal analógica 1.
	  - Bit 3: alarma del mínimo 2 de la señal analógica 1.
	  - Bit 4: alarma del máximo 1 de la señal analógica 2.
	  - Bit 5: alarma del máximo 2 de la señal analógica 2.
	  - Bit 6: alarma del mínimo 1 de la señal analógica 2.
	  - Bit 7: alarma del mínimo 2 de la señal analógica 2.
	- Byte bajo: Estado de las alarmas de la señal analógica 3.
	  - Bit 0: alarma del máximo 1 de la señal analógica 3.
	  - Bit 1: alarma del máximo 2 de la señal analógica 3.
	  - Bit 2: alarma del mínimo 1 de la señal analógica 3.
	  - Bit 3: alarma del mínimo 2 de la señal analógica 3.

23	- Byte alto: valor de la señal analógica 1. - Byte bajo: valor de la señal analógica 2.
24	- Byte alto: valor de la señal analógica 3.
25 - 38	Cada byte de cada uno de estos registros puede contener un número que corresponde a una alarma activa. Este bloque de registros funciona como una pila o stack, donde la alarma más reciente de la pila se encuentra en el primer elemento del stack. La extensión de la misma va del byte alto del registro 23 al byte bajo del registro 36, para un total de 28 alarmas como se muestra en al figura A.2.1. Esta pila se utiliza para llevar un registro cronológico en forma temporal, tan solo por el tiempo necesario para reportar la alarma al Centro de Control, una vez reconocida la alarma, se extrae del stack.

Tabla A.2.5 Descripción del contenido de los registros Modbus 39 al 52 del PIC maestro

Registro Modbus	Contenido
39 - 52	Cada byte de cada uno de estos registros puede contener un número que corresponde a un contador de décimas de segundo, de la alarma ubicada en la misma posición pero en el stack de alarmas. Este bloque de registros funciona como una pila o stack, donde el contador de la alarma más reciente de la pila se encuentra en el primer elemento del stack. La extensión de la mismo va del byte alto del registro 37 al byte bajo del registro 50, para un total de 28 contadores de las alarmas como se muestra en al figura A.2.2. Esta pila se utiliza para que el registro cronológico temporal de las alarmas sea un poco más preciso, ya que es posible conocer en forma aproximada la diferencia de tiempo entre una alarma y otra.

Regsitros Modbus	Contenido	
25	Parte alta	Elemento 1 del stack
	Parte baja	Elemento 2 del stack
26	Parte alta	Elemento 4 del stack
	Parte baja	Elemento 5 del stack
* * *		
38	Parte alta	Elemento 27 del stack
	Parte baja	Elemento 28 del stack

Figura A.2.1 Stack de alarmas en los registros Modbus 25 al 38

Regsitros Modbus	Contenido	
39	Parte alta	Contador de décimas de segundo de la alarma en elemento 1 del stack
	Parte baja	Contador de décimas de segundo de la alarma en elemento 2 del stack
40	Parte alta	Contador de décimas de segundo de la alarma en elemento 3 del stack
	Parte baja	Contador de décimas de segundo de la alarma en elemento 4 del stack
* * *		
52	Parte alta	Contador de décimas de segundo de la alarma en elemento 27 del stack
	Parte baja	Contador de décimas de segundo de la alarma en elemento 28 del stack

Figura A.2.2 Stack de contadores de décimas de segundo de las alarmas, implementado en los registros Modbus 39 al 52

Indice

Apéndice A.3: Registros internos del PIC esclavo.

Tabla A.3.1 Descripción del contenido de los registros 1 al 16 del PIC esclavo

Número Registro	Contenido
1	- Byte bajo: Dirección del esclavo Modbus (planta de emergencia)
2	- Hay planta en el sitio
3	- Byte alto: valor analógico que causa alarma por máximo 1 en señal 1
4	- Byte bajo: valor analógico que causa alarma por máximo 2 en señal 1
5	- Byte alto: valor analógico que causa alarma por mínimo 1 en señal 1
6	- Byte bajo: valor analógico que causa alarma por mínimo 2 en señal 1
7	- Byte alto: valor analógico que causa alarma por máximo 1 en señal 2
8	- Byte bajo: valor analógico que causa alarma por máximo 2 en señal 2
9	- Byte alto: valor analógico que causa alarma por mínimo 1 en señal 2
10	- Byte bajo: valor analógico que causa alarma por mínimo 2 en señal 2
11	- Byte alto: valor analógico que causa alarma por máximo 1 en señal 3
12	- Byte bajo: valor analógico que causa alarma por máximo 2 en señal 3
13	- Byte alto: valor analógico que causa alarma por mínimo 1 en señal 3
14	- Byte bajo: valor analógico que causa alarma por mínimo 2 en señal 3
15	- Byte alto: Máscara de habilitación de las 4 alarmas de las señales analógicas 1 y 2. - Bit 0 = 1, máximo 1 de la señal analógica 1 habilitado. - Bit 1 = 1, máximo 2 de la señal analógica 1 habilitado. - Bit 2 = 1, mínimo 1 de la señal analógica 1 habilitado. - Bit 3 = 1, mínimo 2 de la señal analógica 1 habilitado. - Bit 4 = 1, máximo 1 de la señal analógica 2 habilitado. - Bit 5 = 1, máximo 2 de la señal analógica 2 habilitado. - Bit 6 = 1, mínimo 1 de la señal analógica 2 habilitado. - Bit 7 = 1, mínimo 2 de la señal analógica 2 habilitado.
16	- Byte abajo: Máscara de habilitación de las 4 alarmas de la señal analógica 3. - Bit 0 = 1, máximo 1 de la señal analógica 3 habilitado. - Bit 1 = 1, máximo 2 de la señal analógica 3 habilitado. - Bit 2 = 1, mínimo 1 de la señal analógica 3 habilitado. - Bit 3 = 1, mínimo 2 de la señal analógica 3 habilitado.

Tabla A.3.2 Descripción del contenido de los registros 17 al 36 del PIC esclavo

Número Registro	Contenido
17	- Byte bajo: Estado de la respuesta del esclavo modbus.

18	- Byte alto: Byte de estado.
	  - Bit 0 = 1, alarma activa.
	  - Bit 1 = 1, alarma activa en la planta.
	  - Bit 2 = 1, hay una planta en el sitio.
	  - Bit 3 = 1, no hay respuesta de la planta.
	  - Bit 4 = 1, error interno en el microcontrolador esclavo.

19	- Byte alto: Estado de las alarmas de las señales analógicas 1 y 2 (si el bit está en 1 la alarma respectiva está activa).
	  - Bit 0: alarma del máximo 1 de la señal analógica 1.
	  - Bit 1: alarma del máximo 2 de la señal analógica 1.
	  - Bit 2: alarma del mínimo 1 de la señal analógica 1.
	  - Bit 3: alarma del mínimo 2 de la señal analógica 1.
	  - Bit 4: alarma del máximo 1 de la señal analógica 2.
	  - Bit 5: alarma del máximo 2 de la señal analógica 2.
	  - Bit 6: alarma del mínimo 1 de la señal analógica 2.
	  - Bit 7: alarma del mínimo 2 de la señal analógica 2.

20	- Byte bajo: Estado de las alarmas de la señal analógica 3 (si el bit está en 1 la alarma respectiva está activa).
	  - Bit 0: alarma del máximo 1 de la señal analógica 3.
	  - Bit 1: alarma del máximo 2 de la señal analógica 3.
	  - Bit 2: alarma del mínimo 1 de la señal analógica 3.
	  - Bit 3: alarma del mínimo 2 de la señal analógica 3.

21	- Byte alto: valor de la señal analógica 1.
22	- Byte alto: valor de la señal analógica 2.
23	- Byte alto: valor de la señal analógica 3.
24	- *****
25 - 36	- Contadores de eventos de cada una de las alarmas de las señales analógicas. Cada elemento de este bloque lleva un conteo para cada alarma, de tal forma que para activar una alarma se deben haber presentado un número continuo de eventos iguales, es decir, la alarma se debe activar después de un número continuo de muestras donde la misma este activa.

Anexo B.1: Principales características de los circuitos integrados utilizados

MICROCHIP

PIC18CXX2

High-Performance Microcontrollers with 10-Bit A/D

High Performance RISC CPU:

- C-compiler optimized architecture/instruction set
  - Source code compatible with the PIC16CXX instruction set
- * Linear program memory addressing to 2M bytes
- * Linear data memory addressing to 4K bytes

Device	On-Chip Program Memory		On-Chip RAM (bytes)
	EPROM (bytes)	# Single Word Instructions	
PIC18C242	16K	8192	512
PIC18C252	32K	16384	1536
PIC18C442	16K	8192	512
PIC18C452	32K	16384	1536

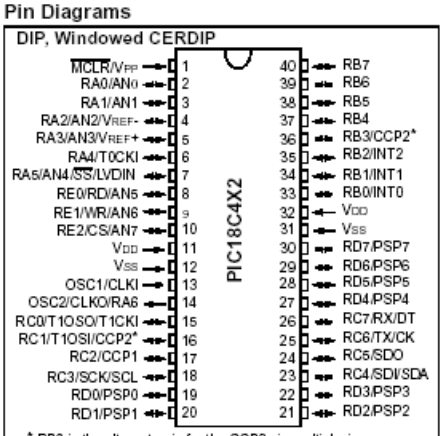
- * Up to 10 MIPs operation:
  - DC - 40 MHz osc./clock input
  - 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- * 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- **Timer0** module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- **Timer1** module: 16-bit timer/counter
- **Timer2** module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- * **Timer3** module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two **Capture/Compare/PWM** (CCP) modules. CCP pins that can be configured as:
  - Capture input: capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1- to 10-bit. Max. PWM freq. @:8-bit resolution = 156 kHz, 10-bit resolution = 39 kHz
- **Master Synchronous Serial Port** (MSSP) module. Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I^2C™ master and slave mode
- **Addressable USART** module:
  - Supports interrupt on Address bit
- **Parallel Slave Port** (PSP) module

Pin Diagrams

DIP, Windowed CERDIP



* RB3 is the alternate pin for the CCP2 pin multiplexing.

NOTE: Pin compatible with 40-pin PIC16C7X devices

Analog Features:

- **10-bit Analog-to-Digital Converter** module (A/D) with:
  - Fast sampling rate
  - Conversion available during sleep
  - DNL = ±1 LSb, INL = ±1 LSb
- Programmable **Low-Voltage Detection** (LVD) module
  - Supports interrupt on low voltage detection
- Programmable Brown-out Reset (BOR)

Special Microcontroller Features:

- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming (ICSP™) via two pins

CMOS Technology:

- Low-power, high-speed EPROM technology
- Fully static design
- Wide operating voltage range (2.5V to 5.5V)
- Industrial and Extended temperature ranges
- Low-power consumption

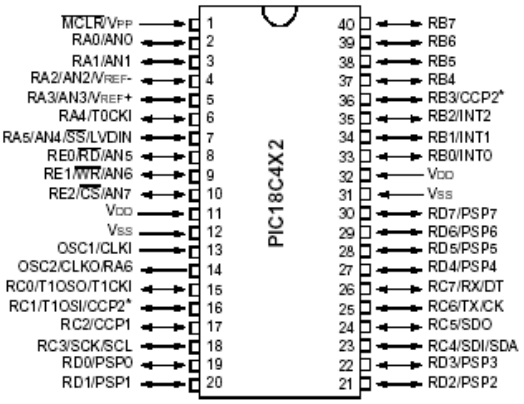
 Preliminary

Figura B.1.1 Descripción general de las características del microcontrolador PIC18C252.

Indice

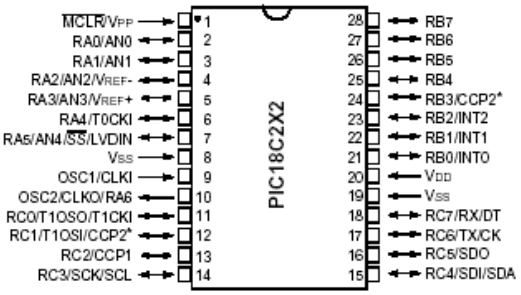
Pin Diagrams (Cont.'d)

DIP, JW



NOTE: Pin compatible with 40-pin PIC16C7X devices

DIP, SOIC, JW



* RB3 is the alternate pin for the CCP2 pin multiplexing.

NOTE: Pin compatible with 28-pin PIC16C7X devices

Figura B.1.2 Distribución de pines del microcontrolador PIC18C252.

Indice

FAIRCHILD SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS244
Octal 3-STATE Buffer/Line Driver/Line Receiver

General Description

These buffers/line drivers are designed to improve both the performance and PC board density of 3-STATE buffers/drivers employed as memory-address drivers, clock drivers, and bus-oriented transmitters/receivers. Featuring 400 mV of hysteresis at each low current PNP data line input, they provide improved noise rejection and high fanout outputs and can be used to drive terminated lines down to 133Ω.

Features

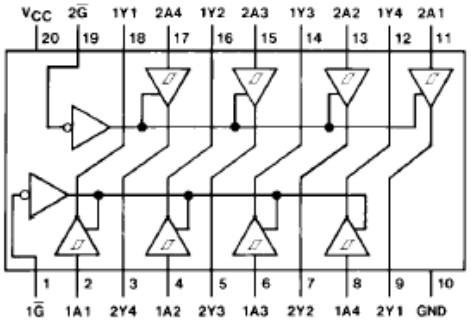
- 3-STATE outputs drive bus lines directly
- PNP inputs reduce DC loading on bus lines
- Hysteresis at data inputs improves noise margins
- Typical I_{OL} (sink current) 24 mA
- Typical I_{OH} (source current) −15 mA
- Typical propagation delay times
  - Inverting 10.5 ns
  - Noninverting 12 ns
- Typical enable/disable time 18 ns
- Typical power dissipation (enabled)
  - Inverting 130 mW
  - Noninverting 135 mW

Ordering Code:

Order Number	Package Number	Package Description
DM74LS244WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
DM74LS244SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS244N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram



Function Table

Inputs		Output
$\overline{G}$	A	Y
L	L	L
L	H	H
H	X	Z

L = LOW Logic Level
H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
Z = High Impedance

 DS008442 www.fairchildsemi.com

Figura B.1.3 Características generales y distribución de pines del buffer 74LS244.

Indice

19-4323; Rev 10; 8/01

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

Features

Superior to Bipolar

- Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)
- Low-Power Receive Mode in Shutdown (MAX223/MAX242)
- Meet All EIA/TIA-232E and V.28 Specifications
- Multiple Drivers and Receivers
- 3-State Driver and Receiver Outputs
- Open-Line Detection (MAX243)

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

*Contact factory for dice specifications.

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

Figura B.1.4 Descripción general del manejador de digital a RS-232, MAX233.

Indice

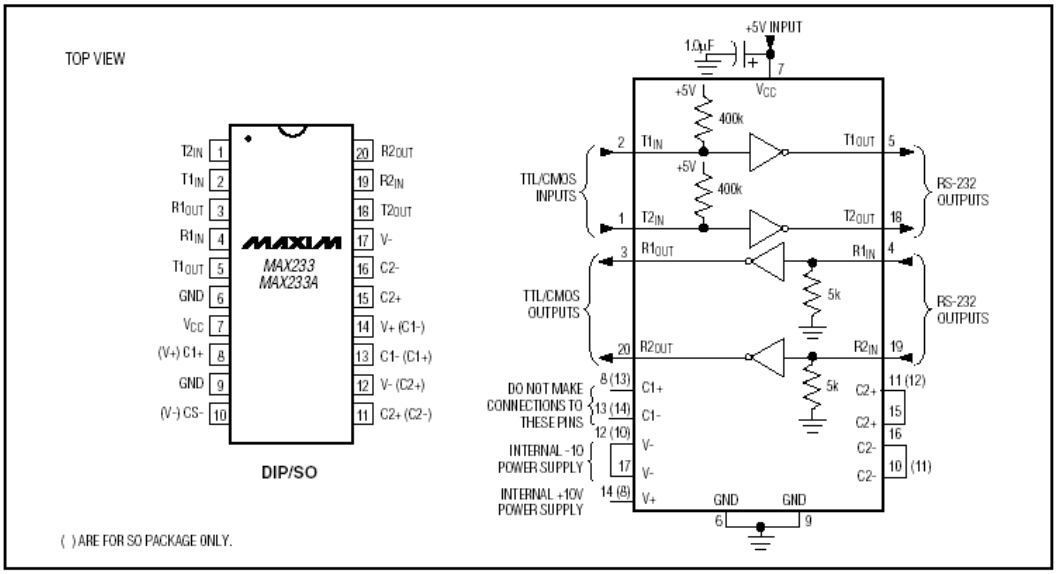


Figure 11. MAX233/MAX233A Pin Configuration and Typical Operating Circuit

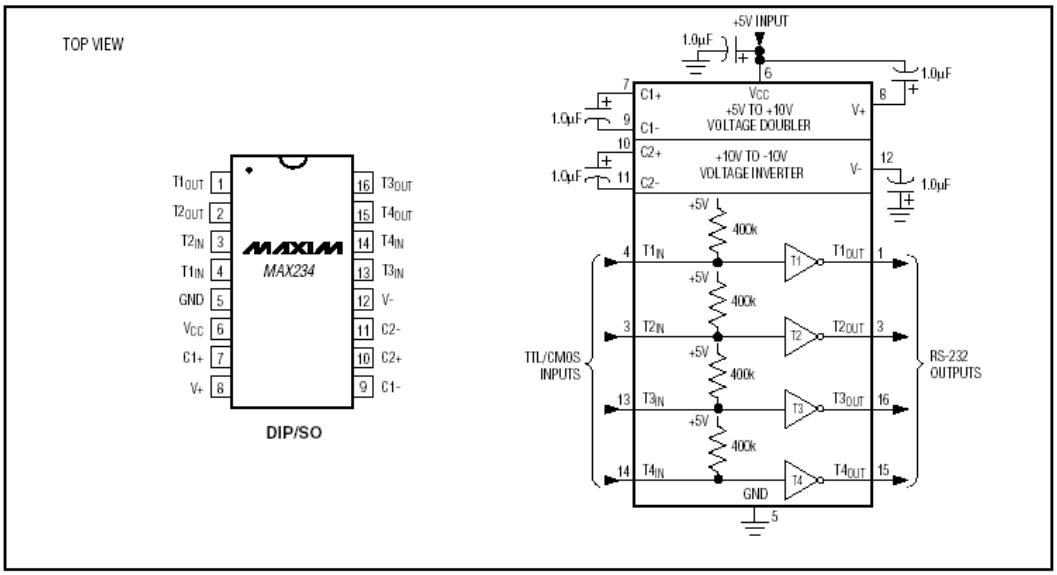


Figure 12. MAX234 Pin Configuration and Typical Operating Circuit

Figura B.1.5 Distribución de pines del manejador MAX233.

Indice