PROTOCOLOS EN REDES DE MICROCONTROLADORES

Autor: Ricardo A. López

Director: Fernando G. Tinetti

Tesis presentada para obtener el grado de Magister en Redes de Datos.

Facultad de Informática - Universidad Nacional de La Plata

Mayo de 2010

INDICE

CAPÍTULO 1: INTRODUCCIÓN, ALCANCES Y OBJETIVOS DE LA TESIS

RESUMEN

Los microcontroladores están inmersos en nuestra forma de vida. Los encontramos en automóviles, lavarropas, celulares, reproductores MP3, agendas y en un sinfín de sitios en nuestra vida cotidiana. La capacidad de integración a muy alta escala (VLSI) - con crecimiento casi exponencial en los últimos años-, hace que estos dispositivos cada día contengan más y más funciones que antes eran impensadas.

Debido a ello, una agrupación de estos dispositivos conectados en red, configura un sistema de control muy poderoso, que dotado de algún protocolo normalizado que permita su interconexión a Internet, le da un alcance prácticamente ilimitado y de gran escalabilidad.

Por lo expuesto, en esta tesis se estudiará la implementación de una red de microcontroladores, definiendo funciones de Control y adquisición de datos, equivalentes a los sistemas de Control y Adquisición de Datos (SCADA) de gran escala. Sobre la definición efectuada, surgirá un protocolo de aplicación que permitirá así un desarrollo Top-Down del sistema.

Sobre la base de la definición lograda, este capítulo describe en un modo general los alcances de la tesis, donde se estudiarán entre otros aspectos, los protocolos de capas de comunicaciones para llegar a dos de las implementaciones más populares utilizadas en los ambientes industriales: RS485 y Ethernet. Si bien la primera es mucho más antigua, sigue aún vigente y se ha potenciado a partir de la creación de interfaces compatibles, citando como ejemplo la inmunidad a ruido eléctrico que le provee una interfase transparente sobre fibra óptica. La segunda, más moderna, ya fija una tendencia debido a su ubicuidad y amplitud de prestaciones.

1.1. INTRODUCCIÓN.

La motivación que primó en la elección del temario, nace de alguna experiencia transitada en el campo de los dispositivos de microcómputo. A través de diversos trabajos efectuados, o por desarrollos experimentales efectuados dentro del ámbito de la universidad, se detectó que podría ser tema de interés el potenciar la relativa capacidad de un dispositivo microcontrolador aislado, mediante su asociación con otros en red. Un sistema así creado podría multiplicar las capacidades de control.

Ya en ámbito laboral de este tesista se había observado la dificultad existente en las Unidades Remotas de Telecontrol con tecnología de los años 90. Estos equipos poco distribuidos, requerían de grandes gabinetes concentradores de la información de campo, con el consiguiente alto costo de cableado, mano de obra y produciendo un sistema poco escalable y altamente propenso a fallas. Se primaba que la información convergiera a las Unidades Remotas y no tanto que estas fueran a recabarla a campo.

A fines de los 90' el paradigma cambió a sistemas más distribuidos: Sistemas que con una relativa capacidad de cómputo, y embebidos en placas de pequeño tamaño y bajo costo, que pudieran ser distribuidos en las instalaciones, todos ellos convergiendo en una red y aportando la misma información pero como componentes desagregados. Esta forma de instalación, sería muy escalable y de fácil reposición ante la falla de una parte. Esta interesante propiedad generó la motivación de estudiar los microcontroladores en red y observar su desempeño.

Por lo expuesto, esta tesis tiene como objetivo, llevar a cabo un estudio de protocolos en redes de microcontroladores, a efectos de internarse en el estado del arte de esta disciplina, para luego efectuar una definición y una implementación de referencia (también llamada "proof of concept").

El establecimiento de una red local de bajo costo implementada con Microcontroladores (uCs), contempla Definición y Diseño de la red, estableciendo conceptos y pautas que permitan en lo posible su extensión a cualquiera de las marcas existentes en el mercado. Bajo esas pautas de definición y diseño generales, se efectuará además el Desarrollo y Test del **Software** de un prototipo.

Desde el punto de vista del **Hardware**, se efectúa la implementación de una Red bajo norma RS-485 vinculando microcontroladores de 8 bits de diferente porte. La red se concibe como agrupaciones de Unidades de Control (UCs) interconectadas, cada una de ellas realizando procesos que interactúan con su propia entrada y salida (campo) y a la vez con un Centro de Control de mayor jerarquía a través de la red.

Existen dos actividades centrales que deben resolver las UCs:

- ✓ Procesos locales (control de lazo cerrado) de interacción con el campo, de mayor prioridad.
- ✓ Procesos relacionados con la red: Sincronización, de alta prioridad y de SCADA, de menor prioridad.

La ejecución concurrente de estos procesos lleva a la implementación de una **Multitarea Colaborativa**. Este es un cuasi - paradigma de Sistemas Operativos, donde cada tarea a realizar se concibe como una parte de un programa general, que debe ser resuelta en un tiempo relativamente corto y balanceado frente al resto de las tareas, impidiendo la apropiación de la CPU.

Los procesos relacionados con la red se implementan, como se verá luego en detalle, mediante un esquema de paso de mensajes establecidos por protocolo. El protocolo que se utiliza es de característica maestro – esclavo, que opera con un esquema equivalente a un protocolo cliente – servidor, en un modo **sincrónico**. En efecto, cada orden emitida desde el Centro de Control, posee un número de comando y una dirección de destino. Este comando se ejecuta en alguna de las UCs de la red, cumpliendo ésta un rol equivalente a un servidor que efectúa un proceso local, para devolver luego un resultado mediante un mensaje que es devuelto al Centro de Control.

La interconexión se realiza mediante una red, conforme a las siguientes variantes de sus capas inferiores:

1. En bus multipunto. (Rs485).
2. Radial (Ethernet).

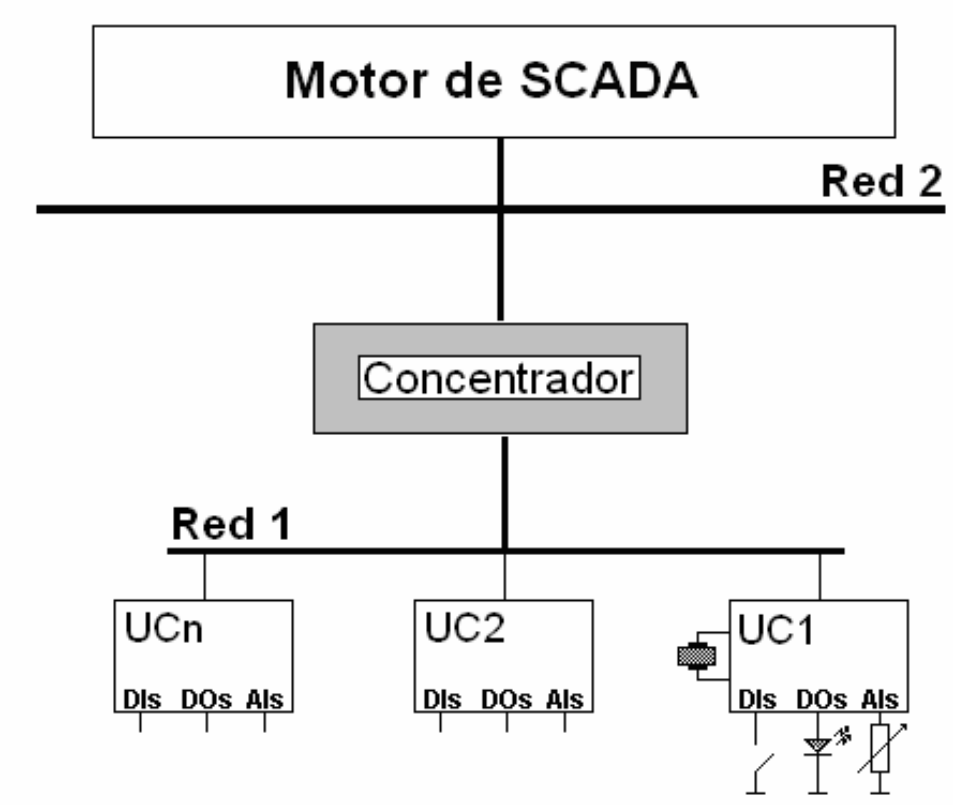


Figura 1.1: Topología de la Red de Microcontroladores y Módulos asociados

En la Figura 1.1, se otorga una disposición de los bloques constitutivos del sistema. Estos, interconectados mediante la utilización de los dos tipos de redes enunciadas, donde se destacan los siguientes componentes:

a) Chips microcontroladores, denominados Unidades de Control (Uc1, Uc2…Ucn), interactuando cada una de ellas con una porción del campo de

entrada y salida y ejecutando cada una su aplicación, sin conexión entre sí, pero todas ellas vinculadas a un dispositivo “maestro” que las concentra.

b) Concentrador de comunicaciones, con su aplicación específica, vinculado a las UCs anteriormente mencionadas, mediante una red multipunto.

c) La interconexión de varias UCs y un concentrador conforma una red, denominada Red 1. Como se verá luego, pueden existir varias de estas Redes 1 por lo que por conveniencia, se adoptará la denominación de Subredes 1,2,…n cuando sea necesario.

d) En la parte superior del dibujo, el Concentrador se vincula mediante otra interconexión al Hardware de un Centro de Control, conformando la Red 2. Comúnmente esta es una red Ethernet.

e) Un Centro de Control (CC) con un motor de Control de Supervisión y Adquisición de Datos (SCADA), aplicación que corre sobre un Host (PC), que se vincula a la Red 2.

No obstante ser el CC un integrante necesario en la red - y dado que sus especificaciones y desarrollo exceden el alcance de esta tesis -, el mismo será simulado mediante un software que emula una terminal de trabajo. Ésta permite la emisión y recepción de paquetes en el protocolo cuyas definiciones se efectúan más adelante. Asimismo, esta terminal permitirá funciones de diagnóstico y test.

La red así configurada, posee todas las bondades, atributos y también las dificultades de un Sistema Distribuido: Ausencia de tiempo global, pérdida de mensajes, fallo de unidades y otros. Por lo expuesto, muchos de los razonamientos y políticas a implementar serán análogos a efectuados ante este tipo de sistemas. En la elección del tipo de red a utilizar en la vinculación entre UCs y el Concentrador, ha primado el hecho que - si bien es importante que la red tenga la mayor capacidad de transmisión -, se considera más importante aún, su bajo costo. Esto es debido al hecho que cada UC particular, no posee habitualmente muchos puntos de vinculación al campo (menos de 50), por lo que no es un factor importante la velocidad. Sí es importante lograr una red que poseyendo bajo costo, tenga a la vez una confiabilidad elevada y sencillez de instalación. Con estas premisas, se cae casi invariablemente en la elección de una conexión multipunto con cable balanceado. Con los avances tecnológicos, es posible la incorporación de segmentos de fibra óptica, siendo su inserción totalmente transparente para los protocolos.

1.2. ALCANCE

Se otorga aquí, descripción y definiciones generales de los distintos módulos que componen el sistema, y que perfilan el alcance del trabajo a desarrollar. Otras definiciones de mayor detalle, se otorgan en el capítulo correspondiente al módulo que se trate. Esas definiciones, son las que en definitiva actúan sobre el diseño del Software y del hardware de las redes.

Unidades de Control

El microcontrolador constitutivo de una unidad de control, deberá contar con la cantidad de Entradas y Salidas, Digitales y Analógicas necesarias, para cumplir con los requerimientos de control de lazo cerrado que exija el campo.

También será posible mediante la selección del componente adecuado, la interacción con otros dispositivos a través de puertos serie sincrónico y asincrónico, en modo punto a punto o multipunto. Estas facilidades resultan interesantes para conexión con trasductores que efectúan la medición de cierta magnitud (temperatura, presión, tensión eléctrica) y se exteriorizan mediante un canal de comunicaciones (típicamente Rs232 o RS485) en un protocolo. En este sentido y como una línea adicional de investigación relacionada con esta tesis, pero no incluida en ella, se estima una sub-red de procesadores digitales de señales (DSP) [13], que conteniendo varias unidades de estos dispositivos (cada una abocada a una medición en particular), se vinculen a una UC que las concentre y recabe sus valores de medición. Esta concepción, evita la concentración de cables en una única UC, lográndose un procesamiento más distribuido, de más bajo costo y mayor facilidad de instalación.

La UCs, contarán con una aplicación, que bajo un esquema general dado de acciones de control, adquisición de datos y comunicaciones con la red, podrá embeber el código específico con la funcionalidad particular que se quiera instrumentar. Sobre este código, por corresponder a una funcionalidad específica y amplia, no se brindan pautas de diseño específicas, aunque sí las generales que le permiten su contención en el sistema embebido. Las UCs tienen la misión de interactuar con el campo, correspondiendo las variaciones de cantidad de individuos y configuración particular de cada uno con la complejidad del sistema a tratar. Como criterio general, no exhaustivo, se tendrán las siguientes características dentro de una UC:

- Adquisición de estados binarios. (Digital Input).
- Adquisición de valores analógicos. (Analog Input).
- Adquisición de Secuencia de Eventos (SOE), con precisión por debajo de 10^{-3} segundo.
- Aceptación de Sincronización horaria interna.
- Acciones de Control sobre el campo. (Digital Output)
- Utilización del Puerto Sincrónico (SSP) como interface de comunicación con dispositivos serie sincrónicos (memorias y otros).
- Utilización del Puerto Asincrónico (UART), con conversión a red Half-Duplex, RS485, para comunicación con dispositivos serie asincrónica (DSP, trasductores y otros).
- Utilización del Puerto Asincrónico (UART), con conversión a red Half-Duplex, RS485, para comunicación con un concentrador de comunicaciones.

Concentrador de Comunicaciones

Tal cual se ha definido más arriba, el Concentrador tiene como misión esencial la de ser:

1. **Puente** o Gateway, entre la Red 1 (bajo norma RS485) que aglutina las Unidades de Control, con la Red 2, bajo norma Ethernet 2, que puede aglutinar varios concentradores y el CC.

Este esquema permite una configuración como la que se otorga en la Figura 1.2 En la misma puede apreciarse cómo las UCs se agrupan en un esquema en árbol con

dispositivos concentradores en su raíz. A su vez una agrupación de estos, se vincula a uno o varios centros de control. Lógicamente, la agrupación de una cantidad importante de UCs por cada concentrador, implica la aparición de un flujo importante de datos que fluye virtualmente desde cada concentrador. Esta situación motiva que la Red 1 posea una capacidad importante. Debido a ello, se hace uso de ethernet.

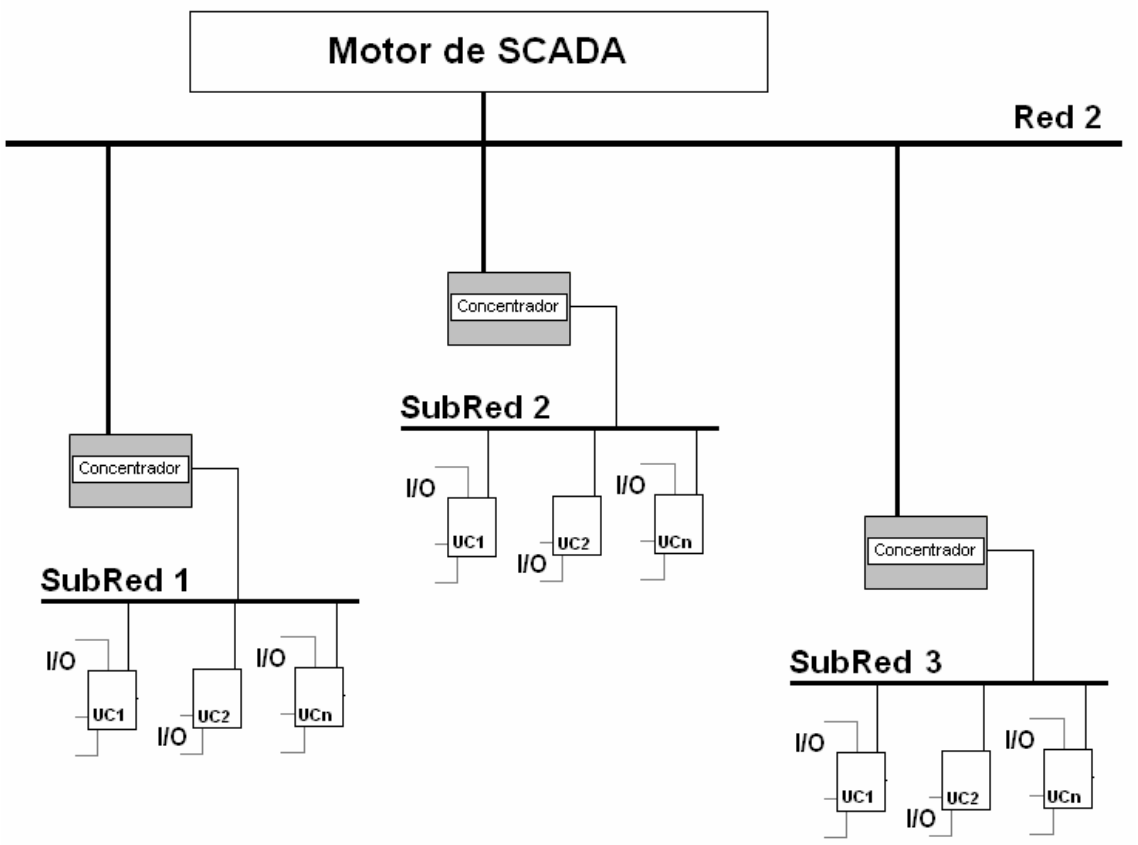


Figura 1.2: Topología en subredes.

Como puede apreciarse en la Figura 1.2, la concentración de UCs mediante un enlace multipunto, determina que deba utilizarse un protocolo maestro – esclavo para arbitrio de las comunicaciones en el bus. Esta situación hace que se determine al Concentrador como maestro, de forma tal que las comunicaciones en el bus son canalizadas por él, en tránsito hacia la Red 1. Esta situación, permite una interesante propiedad del bus controlado, que puede aprovecharse para otra función esencial del Concentrador, a saber:

2. Sincronización interna de las Unidades de Control que se encuentran en las distintas Subredes 1, debajo de cada concentrador, mediante una técnica de **Broadcast Embebido**.

Esta acción de sincronización que ejerce cada concentrador sobre su red dependiente, permite una elevada precisión de los relojes individuales de cada UC. En función de ello, puede lograse sincronización externa con el UTC, adicionando a cada concentrador un dispositivo GPS. La segmentación de la red en subredes es

beneficiosa, dado que el agregado de sincronización externa a cada segmento de red, permite un grado alto de confiabilidad ante fallas en alguna de las subredes y/o sus concentradores.

En la figura también se vislumbra, de qué manera el sistema SCADA situado en un host – constitutivo del CC y sobre una red ethernet, normalizada -, puede tener vinculación a un WEB Server o dispositivo equivalente, que permita la interacción con Internet.

Además de las dos funciones esenciales ya asignadas, la inclusión del Concentrador se encuentra justificada por la necesidad de disponer de un hardware, que establezca una relación peer-to-peer con las UCs integrantes de una subred. En efecto - como se verá luego en los detalles de implementación de la capa física -, es necesario poseer un control íntimo e inmediato de las líneas de handshake para arbitrio del bus Half-Duplex, ya que de otra manera, con una aplicación en más "alto nivel" residente en un Host, podrían acumularse demasiados errores de comunicación o no aprovechar al máximo la capacidad ya austera del bus. En otra instalación de Red (Ethernet en ambas redes), tal Concentrador podría ser un simple Hub o Switch Ethernet.

Redes

La Red1 puede implementarse en dos variantes de interés:

- Bajo norma RS485.
- Bajo norma Ethernet 2.

Respecto a la Red 2, también se tendrán dos posibilidades:

- Establecer un enlace Rs232 con un concentrador y una PC (único cliente). Aprovechable en instalaciones pequeñas.
- Establecer un enlace Ethernet de varios concentradores con una o varias PCs que fuera necesario situar en la Red 2. Hay ventajas apreciables de velocidad ya que tendríamos al menos 10 Mbps frente a los 0.1 Mbps que ofrece RS-232 o RS-485.

Centro de Control con Motor de Control de Supervisión y Adquisición de Datos

La Aplicación Motor del SCADA es la responsable de adquirir datos desde campo, a través de los concentradores de comunicaciones y depositarlos en el Servidor de BDD, a efectos que las variadas aplicaciones de Interfase Hombre – Maquina (MMI) puedan nutrirse de éstos. Para el logro de este objetivo, el motor correrá sobre un Host, sobre cualquier Sistema Operativo, por lo que deberá emplearse un lenguaje de alto nivel que cumplimente las características necesarias que aseguren su portabilidad.

El CC, es otro proceso que corre normalmente en el mismo Host y que se caracteriza por ser una aplicación que contiene una interfase con el usuario, operando sobre datos que le suministra el motor del SCADA. Yendo más a detalle, puede decirse que en algunos casos esos datos son suministrados mediante una comunicación directa entre los procesos y en otros casos los adquiere desde la base de datos.

Protocolos de Comunicaciones

En base a la topología mostrada en la Figura 1.2 y conforme a la selección de tipos de redes a implementar, también fue necesario tomar alguna decisión acerca de los protocolos de comunicaciones a instrumentar sobre la Red 1.

Caso de capa física en Ethernet:

Para las capas superiores de comunicaciones se adopta TCP/IP. Esto es debido a las ventajas que posee este protocolo, especialmente por ser abierto, su ubicuidad y sus facilidades, de las cuales no hace falta extenderse en esta tesis por la existencia abrumadora de información al respecto. La capa de aplicación fue definida a efectos de adaptarla a las necesidades de este proyecto.

Caso de capa física en RS485:

Aquí se optó por efectuar una definición completa de protocolo teniendo en cuenta los siguientes aspectos:

- Contar con una definición completamente abierta y no comercial, pudiendo definir soluciones propias y expandibles.
- Formular un protocolo sencillo y robusto a efectos de estudio, experimentación y evaluación de rendimiento.
- Es una de las redes físicas más confiables, para cableados de buena extensión (superior a 1000 metros), en bus y balanceada lo que la hace altamente inmune al ruido.
- Es de muy bajo costo y alta disponibilidad.
- La utilización de esta norma física es muy sencilla a partir de la disponibilidad de hardware con el que es posible conectar UARTs de las UCs.
- Se implementa con un bus físico y se puede implementar un protocolo Maestro - Esclavo de manera bastante sencilla para arbitrio del bus, obteniendo grados de libertad para implementación de un sistema broadcast de sincronización para sistemas embebidos.

Por estas características, en los capítulos correspondientes se efectuará una descripción de detalle de las definiciones adoptadas para un protocolo para red Multi-Drop, que se ha dado en llamar **Mara-1**.

1.3. COMUNICACIÓN DE LAS UCS CON EL CENTRO DE CONTROL

Debido al hecho que en casi todo el universo de microcontroladores de distintos fabricantes, existe la capacidad de comunicación serie bajo norma RS232 y teniendo en cuenta que este paradigma no permite comunicaciones multipunto, en las comunicaciones de las UCs con el CC, preferentemente se opta por una técnica de comunicaciones que, - con conversión a norma RS-485 -, admite direccionamiento (unicast) de aquellas Unidades integrantes de una red. Las UCs guardan una condición de Slave, de modo tal que -mediante un protocolo Master / Slave, en Half-

Duplex-, reportan a un CC de SCADA en su condición de Master, utilizando el Concentrador ya anunciado como puente entre dos redes.

En estas condiciones, a través de una conexión bipolar balanceada, el Master emite un mensaje que todos los Slaves escuchan y decodifican para saber, si de acuerdo a la dirección de destino, el mensaje es para alguno de ellos. Verificada la coincidencia de dirección de destino con la propia por parte de una UC, además de la integridad del mensaje, aquella lo transfiere hacia sus capas superiores de software para su tratamiento, mientras que el resto de las UCs de la red, desecha el mensaje quedando en escucha.

El protocolo Master / Slave que se diseñó, posee un sólo Master y múltiples Slaves. En función de ello, debe existir sincronismo entre consultas y respuestas, como en todo protocolo Master / Slave, ya que en una red de estas características existirán eventualmente varios Slaves a consultar. En este ambiente, el Master necesita de datos que los Slaves adquieren en forma periódica y eventual acumulándolos en su propia memoria. Debido a ello, mediante un esquema de mensajes establecido por protocolo, el Master efectúa requerimientos que disparan procesos que se ejecutan en alguno de los Slaves, devolviendo luego algún resultado.

Debido al hecho que la consulta se efectúa normalmente en round –robin, es un objetivo vital obtener la más rápida respuesta por parte de una UC direccionada. Por ello, finalizada la recepción del último carácter del mensaje entrante y verificada su integridad, se disparan en cascada:

a) El análisis en capa de aplicación del mensaje recibido,

b) La ejecución del proceso remoto encomendado en la llamada,

c) La preparación de la respuesta y,

d) El envío de esta última a la red, a la dirección del Master.

El protocolo también posibilita comunicación en broadcast. Cuando se utiliza esta facilidad (por ejemplo en sincronización), los mensajes se emiten a una dirección especial (FFh) y no hay respuesta por las UCs.

1.4. ORGANIZACIÓN DE LOS CAPÍTULOS

El resto de la tesis está organizado de la siguiente forma:

El capítulo 2 se describe a detalle los requerimientos que usualmente posee el campo de control donde pueden integrarse los sistemas embebidos. Se otorgan algunas descripciones de los tipos de señales presentes y en consecuencia se formulan algunos requerimientos para los sistemas de control.

El capítulo 3 describe la capa de aplicación del protocolo de comunicaciones. En esta capa se definen entidades que se han denominado comandos y que poseen argumentos para llamar funciones que se ejecutan en las UCs.

En el capítulo 4 se describe en general la capa de comunicaciones del sistema y particularmente la capa de transporte, específicamente aplicada para una capa física de una red bajo norma RS-485 y consideraciones cuando se utiliza TCP/IP..

El capítulo 5 describe a detalle las implicancias relacionadas con la adopción de una capa física bajo norma RS-485 y para la norma Ethernet

En el capítulo 6 se efectúa una descripción de la aplicación desarrollada en la UC sobre la base de requerimientos planteada en el capítulo 2. Esta descripción se complementa con el fuente software desarrollado, que se utiliza luego en el prototipo de presentación.

El capítulo 7 efectúa una descripción de un integrante vital de la red: El Concentrador de comunicaciones.

El capítulo 8 describe el desarrollo y adecuación de una aplicación de test de toda la tesis. Se aportan algunos guarismos que permiten mesurar el trabajo efectuado.

En el capítulo 9 se efectúa una descripción del modelo de sincronización elegido para la red 1, otorgando valores de mérito de la sincronización.

En el capítulo 10 se otorgan conclusiones del desarrollo de la tesis y se aventuran algunas líneas de investigación futuras.

CAPÍTULO 2: REQUERIMIENTOS DE LAS APLICACIONES

RESUMEN

En el presente capítulo se otorgan definiciones y se describen los requerimientos que demandarán las aplicaciones hacia los dispositivos microcontroladores.

Dado que estos últimos comúnmente componen Sistemas Embebidos, que poseen una aplicación dedicada al control de un proceso particular, sea eléctrico, industrial o de otra índole, los mencionados requerimientos son de muy amplio espectro. Por ello, veremos que existe un campo bastante heterogéneo de situaciones a cubrir que deberán ser traducidas a variables de registro en cada sistema. Estas variables deberán ser conformadas para su tratamiento en un modo regular en el CC.

La situación apuntada hace necesario acotar en alguna medida el campo de estudio, considerando un escenario particular de la ingeniería como por ejemplo el ámbito eléctrico. Consecuentemente, este ámbito acotado hará surgir definiciones que impactarán en el diseño de los protocolos de aplicación.

Por ello, se considera oportuno describir el alcance de cada concepto que luego, en el desarrollo, será considerado como premisa de entrada para realización de cada etapa.

2.1. INTRODUCCIÓN

A efectos de acotar el espectro de las aplicaciones posibles y de rescatar los aspectos preponderantes en alguna disciplina, se elige un escenario para la toma de definiciones y este es el campo de las mediciones eléctricas.

Esto resulta ser así por dos razones que se consideran con alguna relevancia:

1. El campo de variables eléctricas es inherentemente de un grado elevado de exigencia. En algunos casos es considerado de misión crítica. Esto es así por el hecho que las variables eléctricas poseen cierta velocidad de ocurrencia o en otros casos se disparan eventos en cascada y con una secuencia veloz.
2. Este campo es en el que este tesista se ha desempeñado en los 30 últimos años como profesional y si bien ello no configura ninguna situación que deba ser considerada, es importante poseer alguna experiencia para detectar los puntos débiles de una instalación o aquellos donde es de interés actuar y con algún grado de precisión en la solución a adoptar. Además puede tener cierta importancia ejemplificar el campo real teniendo en consideración la imposición del marco regulatorio eléctrico, que vuelve el escenario un tanto más complicado de abordar.

En recuadro, se remarcan algunos ejemplos puntuales de detalle, que no son necesarios para continuar el hilo de la lectura, pero que ilustran acerca de las particularidades que pueden suscitarse en cada caso.

Los sistemas SCADA (Supervisory Control and Data Adquisition), vinculados a sistemas eléctricos han tenido desde muchos años atrás gran importancia para el manejo de redes de potencia. Los sistemas computados y de comunicaciones han ido evolucionando en velocidad y capacidad de almacenamiento y todo ello ha redundado en confiabilidad en la operación de un sistema eléctrico.

> En los años 70 - previo a la aparición de las computadoras personales y las estaciones de trabajo-, estos sistemas se desarrollaban en torno a minicomputadoras como la PDP 11 de Digital Equipment o ModCom Classic 75. Estos computadores de propósito general eran adaptados al campo de control mediante interfaces y Unidades Remotas de Telecontrol (RTUs), mediante desarrollos que efectuaban las compañias proveedoras del mercado por ejemplo: ABB con su INDACTIC 11, SIEMENS, o FERRANTI con su sistema VANGUARD.
>
> Como ejemplo de estos sistemas se puede citar el Sistema de Telecontrol de la línea Futaleufú - Puerto Madryn construido en 1975/78 o más anteriormente el Despacho Nacional de Cargas, en base a equipamiento Vanguard con computadoras ModCom.

Estos sistemas poseían Unidades Remotas de Telecontrol (RTUs) que permitían la inclusión de placas destinadas a distintas funciones: Adquisición de variables digitales, Analógicas y Emisión de comandos binarios. Las salidas analógicas o las digitales PWM (Pulse Width Modulation) eran de muy rara aparición, pero tenían como utilidad el movimiento de motores o el cambio de posición de los reguladores de tensión en Trasformadores de potencia.

En el ambiente de la generación, transporte y distribución eléctrica, se requieren cierto grado de exactitud de los valores de medida. Estos valores son impuestos por

el Ente Regulador de la Electricidad (ENRE) y las condiciones más relevantes se detallan en el siguiente listado:

- En mediciones para la operación: 0,5%
- En mediciones comerciales entre agentes del mercado: 0,2%
- Como constantes de tiempo en trasductores de medición: 300 ms.
- Oscilaciones de la onda eléctrica: considerar 0,5 a 3 Hz.
- Discriminación de eventos: 1 ms.
- Discriminación de propagación de fallas: 1 µS.

Como se verá luego en más detalle, estos son algunos de los factores de interés para el análisis y seguimiento de operación de un sistema. Por ello, estos son algunos de los guarismos que deberían tenerse en cuenta para la definición y desarrollo del software y protocolos, si quiere obtenerse un grado de éxito en el tratamiento del problema.

2.2. DEFINICIÓN DE ENTRADAS Y SALIDAS

Eventos

En los sistemas de control existe un flujo de información que se propaga entre componentes de un circuito (mecánico, hidráulico, eléctrico, etc.) conformando lo que da en denominarse: **Lazo**. El lazo puede ser **abierto**, donde cada acción se desarrolla en cascada como una consecuencia de causas-efectos, sin que exista ningún vínculo o relación entre la primera que dispara la secuencia y la última que se considera como *efecto o resultante* de todas las anteriores. En otro caso, existe una *realimentación o feedback* entre el último efecto desarrollado y el primero iniciador, conformando un lazo **cerrado**, de un modo tal que éste puede corregirse o modularse gracias a la realimentación, para desarrollar una nueva secuencia. En general, los ciclos de control suelen ser de lazo cerrado, dado que en este caso la acción de control es más segura, precisa y predecible.

En particular, los sistemas de de control eléctrico son casi siempre de lazo cerrado. Solamente se deja librado a sistemas de control de lazo abierto, a una serie de procesos individuales y menores, donde el riesgo es pequeño y conocido de antemano. En esos sistemas de lazo cerrado muchas veces el hombre es parte componente del lazo. Esto es así porque en determinadas circunstancias, debido a su complejidad, se requiere una decisión que sólo el hombre puede resolver (aunque para determinadas acciones sistemáticas, cada vez más, las decisiones son tomadas por computadores). En estos casos el hombre actúa con un rol de **Operador** del sistema.

En estos sistemas realimentados es importante tener en cuenta el orden de magnitud de los tiempos involucrados en el retorno de realimentación que permite efectuar la corrección. En los sistemas eléctricos estos órdenes son bastante dispares y puede compararse los tiempos involucrados en los ejemplos que se exhiben en la Tabla 1.

En cualquiera de los casos apuntados, debe existir el registro de una acción de campo, por parte de un componente electrónico que realiza una traducción del evento a alguna variable eléctrica. A su vez la acción o una serie de acciones simultáneas con o sin relación entre sí, deben ser registradas debido a que es

importante su análisis ulterior y debe estar involucrada en ese análisis, la hora de ocurrencia de cada uno de los eventos apuntados.

El registro, muchas veces se efectúa con equipos computados que están a alguna distancia entre sí (miles de kilómetros o pocos decímetros) por lo que al ser procesos diferentes en distintos computadores, debe haber alguna sincronización entre ellos.

Acciones	Ejemplos	Orden de tiempos involucrados
1) Acciones donde el hombre es parte del lazo cerrado	Apertura y cierre de interruptores. Cambios de puntos de operación de transformadores. Cierres de compuertas hidráulicas.	20 segundos a minutos.
2) Proceso automático (sin intervención del hombre)	Actuación de protecciones eléctricas. Vuelco de álabes de turbinas ante rechazo de carga.	60 a 400 milisegundos
3) Eventos eléctricos con o sin relación causa - efecto	Disparos de protecciones. Apertura simultánea de interruptores.	1 ms a 80 ms
4) Eventos Eléctricos por causas naturales	Propagación de fallas en líneas	1 microsegundo a varios microsegundos.

Tabla 1: Comparación de tiempos de ocurrencia de distintos eventos eléctricos.

Un ejemplo práctico de falla en líneas de Alta Tensión:

En el último caso de la tabla, se presenta para clarificar el escenario, la situación hipotética ante la caída de una descarga eléctrica atmosférica sobre una línea de 1000 kmts, provocando la rotura de un aislador por la ocurrencia de un arco eléctrico y la consecuente salida de servicio de la línea. Como tal falla puede ocurrir en cualquier punto, es deseable conocer ese punto con alguna precisión a efectos que la dotación de mantenimiento pueda ir rápidamente al sitio a efectuar la reparación.

Para efectuar la detección del punto de ocurrencia del evento, existen equipos localizadores de fallas que situados en cada extremo de la línea, detectan el tiempo de llegada del frente de onda del disturbio que se propaga por la línea de Alta Tensión y que viaja a velocidades cercanas a la de la luz (se propaga en el aire y el conductor). Obviamente, dicho frente de onda llegará en general con distintos tiempos a cada uno de los extremos y el principio de detección implica la correlación del registro temporal de la llegada de este mismo evento a cada extremo. Lógicamente los relojes de cada equipo deben estar sincronizados mediante algún método, pero si supusiéramos por un momento que el error

máximo en esta sincronización es de 1 microsegundo, calculando la distancia que la luz recorre en ese tiempo, tenemos:

D = 300 000 Kmts/ seg * 1 µS = 0,3 Kmts = 300 metros

Este es el grado de incertidumbre que tendremos en la localización de la falla y es aceptable para la compañía.

Para lograr esas precisiones en la sincronización de los equipos detectores se utilizan GPS de cierta calidad que recaban información de varios satélites que les otorgan la precisión deseada.

Un ejemplo práctico de falla en Estaciones Transformadoras de Alta Tensión:

Como un ejemplo de la acción crítica que se posee en el caso 2) de la Tabla 1 y que claramente justifica la adopción de técnicas donde el hombre no es parte integrante del lazo de control, podemos citar el despeje de fallas en el caso de transformadores.

En este tipo de fallas, es común que se produzca arco eléctrico por algún detrimento o contaminación del aceite aislante entre bobinados de un transformador. En el momento del arco, se calienta al extremo gran parte de la masa de aceite aislante existente en la cuba de un transformador y se libera hidrógeno. El no despeje del arco en tiempos que rondan en 60 milisegundos convierten al transformador literalmente en una bomba, que en su explosión provocaría gran daño en si mismo y en equipos aledaños y en ciertas circunstancias al personal de mantenimiento que podría encontrarse en los alrededores.

En esta tesis nos focalizamos en el tercer renglón de la Tabla 1, donde las necesidades de sincronización rondan en algunas **décimas de ms**. En estos casos, el poseer un listado de secuencia de eventos con estampa de tiempo que contemplen la exactitud apuntada, permite a los profesionales analistas efectuar un análisis pormenorizado de las fallas que se presentan, aún habiendo sido registradas por distintos computadores, normalmente ubicados a mucha distancia entre sí.

En el mercado eléctrico existe una regulación que ejerce la Compañía Administradora del Mercado Mayorista Eléctrico (CAMMESA), que requiere que la información de todos los eventos de importancia que produzcan los actores del mercado (Generadores, transportistas y distribuidores de energía eléctrica), sea comunicada en **tiempo real** mediante enlaces directos de comunicaciones con esta empresa, bajo las siguientes condiciones:

a) **Discriminación de 1 ms entre eventos.**
b) **Antigüedad del evento desde su ocurrencia hasta estar en las bases de datos de CAMMESA, no mayor a 20 segundos.**

El no cumplimiento de estas normas impuestas por el ENRE conlleva a importantes multas. Debido a ello, los actores del mercado se ven obligados a configurar enlaces de comunicaciones redundantes (satelitales, fibra óptica, radioenlaces, etc.)

Observemos que con esta modalidad del manejo de la información producida por varios agentes del mercado (en muchos casos con suficiente redundancia) y teniendo en un mismo punto (CAMMESA) la información concentrada para su análisis en tiempo ulterior al ocurrido, pero guardando una correlación muy precisa entre los eventos, es posible laudar en situaciones donde puede ocurrir conflicto de

intereses económicos, ya que la salida de servicio de equipos suele ser causal de fuertes multas a los agentes originadores de las fallas.

Resumiendo entonces, será necesario nutrir a las UCs que conforman la red, de mecanismos que reaccionen ante la ocurrencia de eventos externos que sea deseable obtener, registrando los mismos en una base de datos específica. Típicamente, esta BDD es una cola cíclica de un cierto tamaño residente en la memoria de una UC, de modo tal que los eventos se almacenan en ella a la espera de ser recabados por el CC, que mediante un proceso cíclico aglutinará los eventos recabados en todas las UCs del sistema independientemente de la Red en que estas se encuentren. El tamaño de la cola de eventos que puede mantener una UC, guarda compromiso entre la cantidad de eventos que pueden producirse dentro de un intervalo y las consultas que se efectúan por parte del CC en tal intervalo en condiciones de funcionamiento normal de las comunicaciones, de modo tal que no se produzca sobreescritura de eventos que aún no fueren recabados.

Por otro lado, las UCs deberán llevar la hora en su propio RTC (Real Time Clock), con una exactitud tal que sea posible aplicar a cada evento que la UC registre, la estampa de tiempo del momento de ocurrencia del mismo. Esto teniendo en cuenta que el proceso que efectúa el registro de cualquier evento posee tiempos inherentes que rondan en los 50 microsegundos, lo que hace suponer - teniendo en cuenta la discriminación exigida -, que el registro es instantáneo a fines prácticos.

Estas pautas apuntadas, son condiciones de entrada al diseño y que serán consideradas en la etapa correspondiente.

Estados

La información de eventos aludida en el apartado anterior, tiene su soporte en el estado que se mantiene dentro de cada sistema y que a su vez se conforma sobre la base de los estados individuales de cada uno de los componentes.

El estado de un sistema lo conforman una serie de variables que se recaban con cierta periodicidad en cada UC. Efectivamente, con un periodo que va de varios centisegundos hasta algunos segundos, según el caso, la UC tiene previsto una rutina que recaba estados y los coloca como variables en memoria. Estos estados generalmente se conforman con entradas digitales de campo, entradas analógicas, valores registrados por los subsistemas de comunicaciones o por el propio procesador de cada UC.

Los estados tendrán su correlato en variables que el procesador escribirá en su memoria de lecto-escritura y a diferencia de los eventos que se guardan en una cola rotativa, la toma de un nuevo estado **sobreescribe** los valores que las variables poseían como consecuencia de la toma de estado inmediata anterior.

En general, ese estado global de una UC es recabado también periódicamente mediante un comando de protocolo emitido por parte del CC. Este estado normalmente sirve con dos objetivos bastante diferenciados:

a) El mantenimiento de **mímicos** del sistema, en pantallas de las terminales del CC y que otorgan al operador una información visual que le permite la toma de decisiones en un tiempo relativamente corto (algunos segundos).

b) La **acumulación** paulatina de estos estados sucesivos con periodos de algunos segundos entre sí y con marcas temporales, lo que permiten la conformación de la base de datos **histórica** del sistema.

Como puede notarse, la toma de estado de estas variables tiene relación con los tiempos involucrados en la situación 1) de la Tabla 1, donde el operador es parte del lazo cerrado de control. Aquí también vale efectuar un distingo con los eventos dado que a estos se los trata comúnmente como listados acotados y secuenciales de ocurrencia (SOE: Sequence of events). En el caso de los eventos, es muy importante observar el tiempo de ocurrencia por lo que la forma de presentación surge mayormente de **Querys** que se le efectúan a la BDD del CC y que se presentan como listados, en lugar de mímicos gráficos que sólo representan una imagen poco precisa del sistema en **algún** momento.

En general, las variables de estado tienen conformación bastante heterogénea, debido al hecho que provienen de distintas fuentes externas que retornan información variable en formato. Por ejemplo, las entradas digitales tendrán una representación en 1 ó 2 bits, las entradas analógicas pueden variar en tamaños que oscilan entre 8 y 16 bits, las entradas de ciertos dispositivos compactos que las pueden proveer por puerta serie podrían estar codificadas con el estándar IEEE de simple o doble precisión. Otras variables de estado de la CPU podrán registrarse con un tamaño de 1 byte (por ejemplo, sesgos de sincronización de relojes).

En resumen, será necesario conformar en la UC, rutinas de toma de estado, normalmente incluidas en bucle principal del software, a efectos del registro de las variables apuntadas.

Consecuentemente, será necesario instrumentar un comando de protocolo que se encargue de recabar estas variables a partir de cada UC. Ese comando deberá distinguir mediante el uso de **campos** de distinto tamaño el formato correcto de cada variable y deberá tener también un campo especial que indique cuantas variables se introduce por tipo. Estas condiciones deberán ser tomadas en cuenta en el diseño tanto del software interno de cada UC, como en la definición del protocolo.

Entradas Digitales

Las entradas digitales son uno de los tipos de registro traducidos de fenómenos externos a tener en cuenta. Provienen de fenómenos que tienen característica ON/OFF o abierto/cerrado. Cualquiera que sea la complejidad del fenómeno externo, el mismo, luego de una cadena de mayor o menor largo de adaptaciones se convertirá al menos en un bit dentro de una de las puertas de entrada/salida del microcontrolador componente de una UC. Esto se logra por la existencia de PORTS que embebidos en el chip del microcontrolador, tienen vínculo eléctrico con ciertos terminales (PINS) del chip y permiten conexión eléctrica a circuitos adaptadores en el exterior.

En los sistemas eléctricos, tales tipos de señales provienen de un muy variado tipo de fuentes de las que citamos algunas:

- Apertura/Cierre de interruptores o seccionadores, detectados por contactos fines de carrera de elementos mecánicos.

- Apertura/Cierre de contactos de detectores de gases en transformadores (Sistema Buchholz de detección de descarga interna) por aumento de la presión interna.
- Apertura/Cierre de contactos con par bi-metálico para medición de temperatura en transformadores, motores y otras máquinas eléctricas.
- Apertura/Cierre de contactos traducidos por la circulación de corriente como elemento de detección (cambios de resistencia por temperatura, por posición, presión, etc).

A esta altura no debe confundirse el estado digital (o analógico) con un evento. En el primero tiene primordial importancia **cuál** es el valor de la variable sin importar **cuándo** este cambio ocurrió. En los eventos en cambio, lo primordial es el registro del tiempo de ocurrencia, necesario para determinar la secuencia de fenómenos que ocurrieron en el campo y en un momento totalmente diferido de análisis ulterior. Estas dos entidades tienen un distingo muy claro en el ambiente eléctrico fundamentalmente por la instancia en que son útiles. Los estados permiten al operador tomar una decisión de operación en tiempo interactuando con el sistema. Los listados de eventos en cambio, permiten un análisis ulterior de falla, ya por un profesional más capacitado y a efectos de efectuar modificaciones de mayor envergadura en el propio sistema o para el análisis de la falla con fines contractuales (multas).

En este tipo de entradas es muy común la existencia de múltiples y variados componentes que contribuyen a la **adaptación** de la señal que proviene del fenómeno en campo. Es común el uso de relés repetidores u optoacopladores para lograr dos efectos importantes:

1) **Reducir** la tensión eléctrica a que se somete las puertas de entrada del microcontrolador. Estas raramente admiten tensiones que superan el margen dado por TTL (transistor-transistor logic) que no supera los 5.5 volts.
2) **Aislar** el circuito externo del interno del microcontrolador y sus componentes, a efectos que la eventuales fallas eléctricas que pudieren ocurrir en circuitos externos, avancen sobre el delicado circuito interno del microcontrolador donde las tensiones son muy bajas (por ejemplo 3.3 volts) y las impedancias eléctricas muy altas.

Cualquiera sea el circuito externo con el cumplimiento de las dos premisas anteriormente apuntadas, tendremos que el fenómeno externo estará representado por un 1 (uno) o 0 (cero) digital, determinado por la tensión de una fuente de alimentación vinculada al circuito interno del microcontrolador.

Como muchas veces no es posible lograr con facilidad esta situación en el circuito externo (carencia de contactos auxiliares), comúnmente es necesaria la utilización de relés repetidores que a partir de un contacto o de detección de tensión, derivan múltiples contactos para su utilización por parte de los circuitos de detección y aislación de los microcontroladores.

En resumen, será necesario dotar a las UC, de puertos digitales para el registro binario de fenómenos ocurridos en campo. Estas variables podrán ser utilizadas tanto por las rutinas de toma de estado como por las de registro de eventos, dentro del software de una UC.

Entradas Analógicas

Para la detección de este tipo de señales es muy común la adopción de los conversores analógico-digitales (ADC: Analogic Digital Converter). Este dispositivo convierte una entrada (analógica) eléctrica de tensión en un valor binario (digital) representativo de esa tensión. Para ello la señal analógica - que posee una variación continua en el tiempo -, se la vincula con la entrada del ADC y se somete a ésta a un **muestreo**. Los muestreos de la señal se efectúan a intervalos regulares, normalmente en el ciclo de toma de estado que realiza cada UC. Estos estados pueden tomarse con dos modalidades: Constituyendo un listado consecutivo de estados de cada variable o que cada estado vaya sobrescribiendo su correspondiente anterior.

En estos dispositivos interesa conocer la resolución, típicamente relacionada con la cantidad de bits que posee la muestra digital. Son comunes 10 a 12 bits más signo. Los conversores más usuales en los microcontroladores son los de aproximaciones sucesivas que poseen un compromiso óptimo entre velocidad de conversión y resolución, redundando en un bajo costo al ser un componente que se lo encuentra embebido en el chip del microcontrolador.

En la actividad eléctrica hay dos parámetros que se destacaron al principio y que están directamente relacionados con la calidad de estos dispositivos, a saber:

1. Un primer parámetro es el dado por la normativa de la medición comercial, donde se deben efectuar las mediciones con una **exactitud de 0,2 %** y se debe tener el signo de la magnitud por ejemplo para conocer si una potencia es negativa (saliente desde un generador) o positiva (entrante a una carga). Efectuando cálculos vemos que:

   $$\pm 0{,}2\ \% = 1 / 500 \Rightarrow 9 \text{ bits + signo}$$

   En la actividad es común la utilización de 10 bits con signo que permitiría discriminaciones del 0,1 %.

2. Las constantes de tiempo de los trasductores de medida que deben ser **inferiores a 300 ms**. De otra manera, si la inercia de los trasductores es superior a esa cifra, difícilmente pueda efectuarse el seguimiento de la amplitud de la tensión, especialmente en aquellos casos que haya oscilación del sistema con periodos de oscilación de hasta 300 ms.

Trasductores de Medida:

Los trasductores de medida son dispositivos que efectúan una cuantificación de la onda de tensión o corriente a partir de transformadores de medida. La magnitud eléctrica a medir suele ser en muchos caso muy elevada (500 Kvolts en el SIN: Sistema Interconectado Nacional), debido a ello, debe ser reducida con transformadores de tensión de exactitudes compatibles con el valor de exactitud deseado. Son comunes exactitudes por debajo del 0,1% dado que en un sistema de medición el error final es la sumatoria de los errores relativos de los componentes involucrados en la cascada.

La tensión reducida comúnmente es llevada a amplitudes 110 volt RMS * 1,41. La tensión que se obtiene es reducida pero una copia fiel de la original y en tiempo real conteniendo su contenido armónico. Con el valor analógico a su entrada el trasductor efectúa un cálculo y en un tiempo relativamente corto (300

ms, el orden de su constante de tiempo), entrega una magnitud continua linealmente relacionada con el valor de la entrada.

Comercialmente los trasductores de medida son específicos de una magnitud eléctrica a medir (tensión, corriente, potencia activa y reactiva, frecuencia) y sus salidas están normalizadas con magnitudes que varían como:

0 – 5 Volts

0 – 1 mA

0 – 20 mA

4 – 20 mA

En estos dispositivos existe una relación directa y lineal de la magnitud que se mide a la entrada y el valor que entrega el trasductor. En el último caso por ejemplo tenemos que el límite inferior de 4 mA se condice por ejemplo con 0 Volts y el valor máximo de 20 mA con una tensión de 130 volts.

Esta elección tiene dos ventajas: Por un lado, si el circuito se avería la corriente de salida no estará entre 4 mA y 20 mA y no podrá ser confundida con un valor cero de la magnitud principal. Por el otro, se deja un margen superior por encima de los 110 volts nominales para detectar sobretensiones en el sistema sobre el que se efectúa la medición.

Esta magnitud continua que entrega el trasductor es aceptada por el Conversor AD de la UC y se efectúa una medición de un valor estable durante el tiempo que dura la conversión (en el orden de 10 μS). Los microcontroladores poseen varios PINs del Chip de la unidad, dedicados a entradas digitales aunque esas entradas son **multiplexadas** conformando **canales** que son derivados correctamente a la entrada de un único conversor con el que normalmente cuenta cada unidad.

Los trasductores también contemplan una distorsión armónica máxima de la onda de tensión sobre la cual basan su índice de clase de medición. La deformación armónica **THD** no debe ser mayor al 2,5 %. Eso se entiende de la siguiente forma:

$$THD = \left(\sum e_i^{\,2} \right)^{1/2} / e1 \leq 0{,}025 \qquad \text{con i entre 2 y 50}$$

O sea, la raíz cuadrada de la sumatoria de los componentes de frecuencia desde la armónica 2 a la 50 elevados al cuadrado, dividido la componente fundamental debe ser menor o igual al 2,5%.

En resumen, será necesario dotar a la UC de varios canales de conversión Analógico-Digital, con una resolución de al menos 10 bits más signo.

Salidas Digitales

Este tipo de información es la que refleja las realimentaciones que efectúa el control sobre el sistema, ya sea el control automático o el que involucra a un Operador como parte del lazo.

En los sistemas eléctricos de potencia son muy raras las intervenciones más complejas que la **unaria** sobre el sistema. Esto es, las acciones de salida están perfectamente individualizadas y por seguridad de operación de la red, es normal que cada comando sea una acción aislada y distinguible de otras acciones de salida

similares, como parte integrante de un protocolo de operación paso a paso que debe seguir el operador del sistema.

No existen para estas salidas mayores requerimientos y comúnmente se adicionan aquí los opto-acopladores y relés repetidores para aumentar el margen de potencia de control exiguo que posee una salida digital (típicamente 5 Volts y no más allá de 25 mA).

En las UC, será necesario dotarlas de este tipo de puertos y es normal en los microcontroladores - que si bien cada salida digital está agrupada en un puerto con tamaño de un byte de datos-, existan instrucciones que permiten el manejo individual de bits como por ejemplo las funciones:

BitSet **Puerto, Nro de bit**

BitReset **Puerto, Nro de bit**

Como puede apreciarse en este pseudocódigo de instrucciones de cualquier microcontrolador comercial, estas instrucciones pueden poner a uno (set) o poner a cero (reset), el bit dado por el numero de bit que figura como segundo argumento de la función cuya dirección esta dada por el puerto, que figura como primer argumento de la función. Estas operación garantizan la operación unaria ya apuntada.

Salidas PWM

Este tipo de salida comúnmente es utilizada para el control de velocidad de motores, por modulación del valor medio de un tren de pulsos de tensión aplicada, que regula bastante eficientemente la velocidad en motores de Corriente Continua. En otros casos de necesidad de mayor precisión puede recurrirse a mecanismos de control para motores paso a paso.

Las etapas de control mediante estás técnicas dentro de los microcontroladores, se limitan a drivers electrónicos con muy bajo nivel de potencia y con tensiones que no superan los 5 volts y corrientes de hasta 25 mA, como las típicas salidas digitales. Debido a ello, casi siempre es necesaria la instrumentación de etapas de electrónica de potencia, para manejo de las corrientes que demandan los servo-motores.

Comunicaciones

Los módulos de comunicaciones son también dispositivos de entrada y salida. Dado que se definen con bastante profundidad en los capítulos dedicados al protocolo, no surgen aquí mayores requerimientos por encima de los que en su momento se definan para la capa física, en función de las demandas de las capas superiores.

En general, el único requerimiento que fija un sistema eléctrico está relacionado con los tiempos en que la información debe llegar al CC, lo que determina una velocidad de trabajo en bps. En las comunicaciones entre las UC y el Concentrador se ha optado primariamente por canales bajo norma RS485 que poseen una velocidad máxima de operación de 115200 bps, la cual es suficiente mente elevada en la gran mayoría de los casos, en relación con la cantidad de información que se maneja. En alguna situación este valor puede no ser suficiente dado que se trata de comunicación en Half Duplex y el canal es compartido por varios individuos en un segmento de red que desemboca en un concentrador.

En casos en que la cantidad de dispositivos compartiendo el canal y/o la velocidad de transmisión no sean suficientes para los requerimientos de una determinada aplicación, siempre es posible la utilización de Ethernet, que en microcontroladores el valor de velocidad puede elevarse a 10 Mbps o aún más.

En otros casos las puertas de comunicaciones serie (tanto asincrónica como sincrónica), suelen utilizarse para comunicación con otros dispositivos más complejos de adquisición de señales, como por ejemplo traductores de medida que entregan la información bajo un protocolo particular sobre comunicación serie RS232 o RS485. Estas puertas también pueden ser utilizadas para vincularlas con procesadores digitales de señales (DSP), cuando el existe alguna complejidad de la señal a medir. Los trasductores de medida son muchas veces DSP que entregan varias magnitudes de una misma onda de tensión o corriente.

Por todo lo expuesto, deberá dotarse a las UCs de uno o dos puertas serie a efectos de la comunicación, más el agregado de otros componentes de interfase.

Conclusiones

Hasta aquí se ha querido otorgar una visión lo más amplia posible pero acotada de la problemática que se presenta en uno de los escenarios posibles de utilización de un sistema como el que se trata en esta tesis.

Los problemas requieren en general un estudio pormenorizado cuando se escapan de las situaciones generales que se ha pretendido exhibir. En lo que sigue, se efectuará definiciones de diseño sobre el sistema que tendrán a su momento fundamento en las características de la entrada y salida desplegada en este capítulo.

CAPÍTULO 3: COMUNICACIONES Y CAPA DE APLICACIÓN

RESUMEN

En el presente capítulo se otorga una descripción general del protocolo de comunicaciones y en particular se describe su capa de aplicación.

En esta capa se definen las entidades que se han dado en llamar comandos y que poseyendo argumentos, brindan una analogía natural con el paradigma de llamadas a procedimientos remotos de un sistema distribuido.

Se define cada uno de los comandos con su objetivo y su descripción a detalle. No obstante, dado que el sistema presenta una considerable capacidad de ampliación de comandos en la medida que los requerimientos de las aplicaciones lo demanden, la definición se limita a los comandos que se han considerado esenciales y que se implementaron en la aplicación de ejemplo que se describe al final de esta tesis.

3.1. INTRODUCCIÓN.

En la construcción del protocolo de comunicaciones se ha seguido - aunque en un modo más simplificado -, el paradigma del Modelo de Intercambio entre Sistemas Abiertos (Modelo OSI) de la ISO, específicamente en la definición de tareas a realizar en cada capa.

Se ha buscado desde el principio la mayor simplicidad posible por varias razones:

1. Normalmente a mayor simplicidad de la definición también corresponde menor complejidad de implementación.
2. Los microcontroladores a utilizar suelen tener muy poca capacidad de procesamiento y de almacenamiento, por lo que no podría imponérseles una carga excesiva solamente para mantener las comunicaciones.
3. Tanto para investigación como para docencia, conviene tener en principio una definición sencilla, para luego adaptarla o hacerla más compleja, dependiendo de aplicaciones y/o conceptos específicos.

La capa de aplicación se mantiene constante para cualquier cambio a efectuar en las capas inferiores de comunicaciones. Este razonamiento, permite mayor simplicidad ante el intercambio de capas inferiores, dependiendo el ámbito de uso.

En el caso de capa física ethernet se vio como natural implementar TCP/IP en las capas intermedias. Esta combinación es un estándar en sistemas embebidos y la complejidad inherente se estudia en el capítulo respectivo. En el caso de capa física bajo norma RS-485, se vio como razonable instrumentar el modelo básico en tres capas recomendado por muchos autores [11] [2].

Este modelo de comunicaciones en tres capas, define de arriba hacia abajo: 1. Las aplicaciones (capa de aplicación), 2. Las comunicaciones (capa de transporte) y 3 Los computadores (capa física). Bajo estas condiciones y siguiendo este paradigma, en nuestro sistema se tiene:

- Una ***Capa de Aplicación***, que tiene definido lo necesario para la encuesta del estado, la ejecución de un control en campo y la configuración de los microcontroladores, además de incluir lo referente a la sincronización de los mismos. En general, se define un protocolo master/slave donde el CC es el master y las Unidades de Control son slaves. Por la estructura jerárquica del sistema (mostrada en la Figura 1.1 y en la Figura 1.2), el CC evidentemente es quien define y sincroniza toda la actividad de red del conjunto.
- Una ***Capa de Transporte***, donde se define todo lo necesario para que las comunicaciones de la capa de aplicación se realicen correctamente, tanto en la secuencia de mensajes, como en el control de errores que podrían acontecer en la capa inferior de la red. En la trama de protocolo - como se verá en su momento-, se distinguen campos que están definidos dentro del segmento de transporte y están asociados a garantizar mensajes libres de errores y hacia el destino adecuado.
- Una ***Capa Física****,* que dependerá de lo existente en Hardware. Esta capa no posee software y para el programador típicamente está representada por los registros de datos, control y status que posee el Hardware específico. Existe entonces a nivel subyacente y ya en software, el código de la capa inmediata superior que es en definitiva, la interfase sobre la que se lee/escribe.

3.2. COMANDOS.

La capa de aplicación es una rutina de software que es incluida a cada lado de la aplicación del maestro o el esclavo. El maestro emite comandos que poseen un significado específico y éstos originan en el esclavo, la ejecución de una rutina dentro de su proceso colaborativo a efectos de cumplimentar el requerimiento efectuado.

En la mayoría de los casos, la rutina que se dispara dentro del esclavo efectúa una acción concreta (rescate de un valor de estados, acción de control sobre un puerto, configuración, etc.) que además implica alguna devolución para indicar el grado o la calidad de la acción ejecutada.

En este sentido, cada comando emitido por el maestro, dispara una rutina concreta dentro del esclavo. Esto es, la acción de comandar, tiene sentido desde el punto de vista del maestro, pues es él quien dispara o comanda la operación. Esto implica que las acciones específicas en el esclavo serán como consecuencia o reacción a un comando ejercido por el maestro.

Debido a esta acción de “comandar” que posee el CC, maestro sobre las unidades remotas, se efectúa una analogía con la llamada a procedimientos remotos que se efectúa en un esquema cliente – servidor, dado que el servidor siempre actúa ejecutando un proceso como acto reflejo a la demanda de un cliente.

Los Comandos son entidades de un byte, que en muchos casos están seguidos de uno o varios bytes que actúan como argumentos. El byte de comando es el primero de la serie de bytes que conforman el mensaje de aplicación. Este mensaje, ya sea que se haya conformado en el maestro o en el esclavo, es la secuencia ordenada de bytes que es encapsulada / desencapsulada por las capas inferiores del protocolo.

Los comandos que se han considerado, dependen de las acciones que se desee ejercer en la Unidad de Control. La combinatoria ofrecida por un byte representativo de un comando es de 256, luego, sería posible ejercer con este esquema 256 acciones de distinto tipo en la Unidad de Control.

A los fines prácticos se ha limitado los comandos válidos a los valores comprendidos entre 0 (00h) y 250 (FAh) dejando el resto de los valores numéricos para ciertas funciones especiales.

Dentro de esta concepción se han diferenciado dos tipos de comandos en congruencia con las definiciones que aquí se describen en relación con las aplicaciones de las Unidades de Control: Comandos de Sistema y de Usuario. Los comandos de sistema son comandos generales, considerados necesarios para el funcionamiento consistente de las aplicaciones. Por otro lado, los comandos de usuario son originados en la necesidad de instrumentar ciertos mensajes, que tienen que ver más con la aplicación específica, definida por la acción de control que ejercerá el microcontrolador. Podría decirse que los comandos de sistema son funciones o servicios de los que puede valerse la aplicación del usuario.

Para el funcionamiento del sistema con el marco definido para esta tesis, fue necesario implementar alrededor de unos 12 comandos de sistema y sólo uno de usuario, relacionado con la aplicación ejemplo. Esta aplicación es un semáforo, que si bien no es una aplicación eléctrica, tiene algunos aspectos congruentes con éstas y además ilustra como ejemplo, sobre la diversidad de aplicaciones admitidas por los microcontroladores.

Descripción general de los comandos

En lo que sigue se otorga una descripción conceptual de cada comando a efectos de otorgar una semblanza de la concepción global del protocolo para más adelante, entrar en el detalle de cada uno de los comandos implementados, de sus variantes y de los argumentos necesarios en cada caso.

Estos comandos deben interpretarse como un menú o set, que la aplicación del CC puede emplear a voluntad conforme al objetivo que se trate. Por ello, algunos comandos son complementarios o de mayor especificidad que otros.

De acuerdo a lo consignado en el capítulo 2 y teniendo en cuenta que son los requerimientos de las aplicaciones los que definen la conformación global del protocolo, los comandos que el CC puede ejercer sobre las UCs (o eventualmente los concentradores) pueden subdividirse como:

1. Comandos para el control y la adquisición de datos.
2. Comandos de configuración.
3. Comandos para Test y diagnóstico.
4. Comandos de Usuario.

Comandos para el control y la adquisición de datos

- *Requerimiento general de estado:* Este comando se utiliza para conocer el estado global de una UC. Ante el requerimiento del CC, la UC devuelve el estado de puertos digitales, puertos analógicos y ciertas variables de estado de interés para el funcionamiento. Si queda espacio en la trama (no se alcanza el valor máximo del segmento), se adicionan los eventos acumulados en la pila. Si aún quedaran más eventos, se lo indica con el campo **MoreEv** de una de las variables de estado y a continuación se puede ejercer un comando específico que recaba eventos.
- *Requerimiento de Eventos:* Este comando se utiliza para requerir eventos acumulados en la UC si los hay. Como se describiera en el capítulo anterior, los eventos son entidades de 8 bytes y se acumulan en una pila. La operación de adquisición de eventos **no es Idempotente**, dado que al colocarse un evento en el buffer de emisión, se actualiza el puntero consumidor. Esto significa que para recuperar eventos que pudieren haberse perdido por comunicaciones, se debe provocar una acción de reenvío (retry), previo a un nuevo requerimiento, desde el buffer respectivo en la UC.
- *Puesta en Hora (PeH):* Este comando se utiliza para imponer el tiempo en una UC desde el CC o desde un concentrador. En la acción se impone el año, el día del año, la hora del día, el segundo de la hora, la fracción de segundo (con una granularidad de 1/32768 de segundo) y el día de la semana.
- *Control de Puerto digital:* Este comando se utiliza para la puesta en alto o bajo de un bit de un puerto digital. Se prefiere el accionamiento de un bit por comando, debido a razones de seguridad en el control.
- *Reinicio de UC (Reset):* Este comando se utiliza para reiniciar una UC determinada. Normalmente se provoca deteniendo el refresco del

WatchDog o alguna técnica similar dependiendo del microcontrolador que se trate, típicamente estableciendo un retardo de una fracción de segundo (típico 512 ms), tiempo que le permite a la UC devolver una respuesta de aceptación previo a su reinicio. El reinicio comienza desde el vector correspondiente al microcontrolador que se trate y las condiciones de reinico las establece el programador.

- *Sincronismo (Sync):* Este es un comando reducido de puesta en hora que permite un ajuste fino en el entorno del segundo. En general lo produce un concentrador o el dispositivo que tenga la misión de llevar el sincronismo. Es un comando para ser emitido en broadcast y es emitido en el instante en que se recibe la información de pasaje por cero del segundo en un GPS. Debido a ello, normalmente se ponen a cero los contadores que llevan la hora en la UC, adicionando un tiempo calculado de retardo debido a la transmisión del mensaje a la velocidad de transmisión de la red. Este tiempo de propagación puede rondar en los 8 ms a una velocidad de 9600 bps.

Comandos de configuración

Los comandos que siguen se utilizan para forzar en tiempo real, valores de variables en la aplicación que se esté ejecutando en la UC. Sirven para múltiples fines pero en general tiene como misión la configuración de la aplicación. Son complementarios dados que ejercen la lectura o escritura de los distintos tipos de memoria con que normalmente cuenta la UC. Lamentablemente no poseen una generalidad importante porque se es cautivo de los distintos formatos de memoria existente en los microcontroladores. En este caso se ha optado por argumentos que se relacionan con un esquema **paginado** de memoria, contándose con un número de página y un offset dentro de la misma. Como podrá advertirse, en algún caso los comandos necesitaran leves modificaciones tanto en el protocolo como en la instrumentación en la UC que se trate, dado que solo permiten manejar 256 páginas de 256 bytes cada una.

- *Escritura de Registro en RAM.*
- *Lectura de Registros de RAM.*
- *Escritura de Registro en EEPROM.*
- *Lectura de Registros de EEPROM.*

Comandos para Test y diagnóstico

- *Requerimiento de Fecha, Hora y sesgos:* Este comando se utiliza para recabar el tiempo actual almacenado en la UC. Adicionalmente se devuelve un valor de 16 bits que es el sesgo existente en el momento del último sincronismo recibido por la UC. Estos valores permiten efectuar un diagnóstico global relacionado con el sincronismo de las UCs. La información del sesgo obtenido el la ultima sincronización permite saber si la frecuencia de emisión del sincronismo es la adecuada o si los cristales utilizados poseen la calidad acorde al requerimiento de precisión por parte de la aplicación.

- *Provocar eventos en UCs:* Este comando se utiliza para provocar eventos ficticios en una o varias UCs. Cómo puede ser transmitido en broadcast, está garantizada la llegada simultánea a cada una de las UCs de una subred. El corrimiento en la llegada puede deberse a pequeñas diferencias de lectura por la UART receptora y diferencias en los procesos que estén cursando las UCs involucradas (Una Uc puede estar cursando una interrupción y otra no). No obstante ello, las UCs que reciben el comando en forma simultánea provocan un evento de un cierto tipo que es acumulado en la pila de eventos y que puede ser recabado más tarde por el CC a efectos de comparación de la estampa de tiempo (Time Stamp) conque fueron registrados.

Comandos de Usuario

- *Cambio de estado en aplicación de usuario:* Este es un comando general de usuario y en este caso se utiliza para efectuar un cambio de estado de la aplicación por la cual se ha optado como ejemplo en esta tesis (un semáforo).

Descripción detallada de los comandos

A continuación se otorga un detalle de cada uno de los comandos implementados, de sus variantes y de los argumentos necesarios en cada caso. Los formatos y ejemplos numéricos se otorgan en **Hexadecimal**.

Comando 00h: Requerimiento general de Estado de UC.

Objeto: El CC encuesta el estado de una UC.

Argumentos: No posee

Formato (Master):

00

Devolución: La UC devuelve el primer byte de comando y a continuación devuelve bytes de su estado de proceso. El estado se expresa en cuatro campos de bytes de tamaño variable. Cada campo se inicia con un byte que indica la longitud total del campo (incluido el propio byte de longitud), excepto el último campo que algunas veces no existe pero cuando lo hace es una cantidad MOD 8 de bytes. Los campos son:

1) Valores de variables de la RAM/ROM de la UC. (VarSys)

2) Valores de los puertos binarios. (DIs)

3) Valores de los canales analógicos. (AIs)

4) Eventos ocurridos desde la última lectura. (no lleva campo long.).

La suma de longitud de los cuatro campos no puede superar **246 bytes**.

Formato (Slave):

00 03 07 08 04 31 32 4A 03 22 7F 84 22 33 66 02 23 05 32

Observaciones

En el ejemplo de formato del slave puede observarse:

En primer lugar el byte de comando (00h) y luego: 1) (campo sombreado) un campo de tres bytes, con el indicador de longitud en primer lugar y luego dos valores de variables: VarSys. 2) A continuación un campo de 4 bytes, con el indicador de longitud en primer lugar y luego tres valores de estados de los puertos digitales de un byte: DIs. 3) A continuación (campo sombreado) un campo de 3 bytes, con el indicador de longitud en primer lugar y luego un valor de estado de un canal analógico de 2 bytes: AIs. 4) A continuación un grupo de 8 bytes constitutivo de un evento: Evs.

El último campo de eventos es condicional. Esto es, que el mismo está presente si la UC poseía un evento acumulado al producirse la consulta. La UC posee una pila de eventos de un largo determinado pero siempre múltiplo de 8 bytes (Tamaño necesario para el almacenamiento de un evento). Como este campo es el último y su existencia es condicional a la existencia de eventos en la UC y de existir siempre tiene un largo fijo (8 bytes), no es necesario introducir el valor de longitud para este campo.

Comando 01h: Requerimiento de eventos.

Objeto: El CC encuesta por eventos existentes en una UC.

Argumentos: No posee

Formato (Master):

01

Devolución: La UC devuelve todos los eventos acumulados que posee o hasta completar el largo máximo de un mensaje de aplicación (246 bytes). El primer byte es el comando. A continuación se encuentra un byte que en la aplicación se denomina **SysFlags**. Este byte contiene una bandera **(MoreEv)** que puesta en uno indica la existencia de más eventos en la UC (los cuales no pudieron ser enviados por excederse el tamaño del buffer). Esta bandera en cero indica que no quedan eventos acumulados en la UC. A continuación se tendrán varios campos de **8 bytes**, donde cada campo es un evento.

Formato (Slave):

01 01 84 16 22 33 66 02 23 05 01 73 22 33 66 02 23 05

Observaciones

En el ejemplo de formato del slave puede observarse:

En primer lugar el byte de comando (01h) y luego (campo sombreado) la variable SysFlags en la que se ha puesto el campo MoreEv en cero. Luego se observan dos campos de 8 bytes cada uno.

Cada campo de 8 bytes posee el siguiente significado:

1er byte: TUC (Tipo, Urgencia, Código)

Bit 7-6 : Tipo de evento

00: Evento de campo (físico)

01: Evento de Usuario

10: Evento de Sistema (comunicaciones y otros)

11: Libre

Bit 5-4 : Prioridad del evento

00..11: Muy Alta, Alta, Baja y Muy baja.

Bit 3-0 : Códigos de Evento

0000: Evento rápido (por interrupción)

0001: Evento lento (por polling)

0010: Escritura diferida de EEPROM (falla)

……: Libre

1000: Detección de falta de Com con el CC

……:

1001: Lámpara quemada (Evento de usuario)

2do byte: PBE (Puerto, Bit, Estado)

Bit 7-4 : Puerto Lógico (de 0 a 15) de 8 bits.

Bit 3-1 : Bit del puerto (de 0 a 7)

Bit 0 : Estado (0 ó 1)

3ro al 6to bytes (4 bytes): Time Stamp1

Se interpreta como un número Big-Endian de 32 bits que contiene:

Bit 31- 26 (6 bits): Año (00 a 63. Años 2000 al 2063).

Bit 25-17 (9 bits): Día del año (01 al 366. Incluye bisiestos)

Bit 16 -12 (5 bits): Hora del día (00 a 23)

Bit 11-0 (12 bits): Segundo de la hora (00 a 3599)

7mo y 8vo bytes (2 bytes): Time Stamp2 (Fracción de Segundo)

Se interpreta como un número de 16 bits pero MOD 32768.

Esta fracción permite una granularidad de 1/32768 de segundo = **0,03 ms**

Comando 02h: Puesta en Hora (PeH).

Objeto: El CC (o el Concentrador) impone la hora a las UCs.

Argumentos: 7 bytes

Formato (Master):

02 20 0A 70 0F 17 12 05

Devolución: Este es un comando que se emite en Broadcast para todas las UC de la Subred por ende no existe devolución**.**

Formato (Slave):

No Existe.

Observaciones

En el ejemplo de formato del master puede observarse:

En primer lugar el byte de comando (02h) y luego (campo sombreado) la fecha y hora. La siguiente zona sin sombrear da la fracción de segundo. El último byte es el día de la semana (DoW).

Cada campo de 7 bytes de argumentos posee el siguiente significado:

1er al 4to byte (4 bytes): Time Stamp1: Año, Día del año, Hora, Segundo.

Se interpreta como un número Big-Endian de 32 bits que contiene:

Bit 31- 26 (6 bits): Año (00 a 63. Años 2000 al 2063).

Bit 25-17 (9 bits): Día del año (01 al 366. Incluye bisiestos)

Bit 16 -12 (5 bits): Hora del día (00 a 23)

Bit 11-0 (12 bits): Segundo de la hora (00 a 3599)

5to y 6to bytes (2 bytes): Time Stamp2 (Fracción de Segundo)

Se interpreta como un número de 16 bits pero MOD 32768.

Esta fracción permite una granularidad de 1/32768 de segundo = **0,03 ms**

7mo byte: DoW (Día de la semana)

Bit 15 : siempre 0.

Bit 14-0 : Fracción de segundo (n) = n/32768 Seg.

Análisis del Ejemplo: 02 20 0A 70 0F 17 12 05

20 02 70 0F = 001000-00 | 0000101-0 | 0111-0000 | 00001111

Para la interpretación de los 4 bytes de TimeStamp1, considérese que la barra vertical separa los bytes y el guión separa los campos. Luego, los campos arrojan los siguientes valores y su codificación:

16 : Año 2016
05: Día 5 (5 de enero)
07: Hora 7.
15: Segundo 15. Se interpreta como 00 min, 15 seg.

Para la interpretación de los 2 bytes de TimeStamp2, se toma el valor numérico de la siguiente forma:

1712h: 5906. Resultando 5906/ 32768 = 0,18023 s.

Para la interpretación del último campo (DoW)
05: día 5 = viernes.

Comando 03h: Control de puerto digital (1 bit).

Objeto: El CC pone en 1/0 un bit de un puerto de una UC.

Argumentos: 2 bytes

Formato (Master):

03 20 0A

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente.

Formato (Slave):

03 20 0A

Observaciones

En el ejemplo puede observarse:

En primer lugar el byte de comando (03h) y luego (campo sombreado) los argumentos.

Cada campo de 2 bytes de argumentos posee el siguiente significado:

1er Byte: Indirección / Puerto.

Bit 7: En 0 dirección directa. En 1: dirección indirecta (lógica a través de una tabla de indirección).

Bit 6-0: Dirección (0 a 127).

2do byte: Estado/ Nro de bit del puerto.

Bit 3: 0/1 Se pone a Bajo o Alto el bit del puerto.

Bit 2-0: Nro de bit del puerto de 8 bits (0-7).

Análisis del Ejemplo: 03 20 0A

1er byte:
bit 7: 0 dirección directa
bit 6-0: 010 0000 = Puerto 32

2do Byte:

Bit 3: 1. Poner a alto.
Bit 2-0: 010 = bit 2 del puerto directo indicado.

Comando 04h: Escritura de un registro de RAM.

Objeto: El CC escribe un valor de 8 bits en un registro de la RAM de una UC. Este es un comando de uso reservado.

Argumentos: 5 bytes

Formato (Master):

04 AA 55 01 32 1F

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente.

Formato (Slave):

04 AA 55 01 32 1F

Observaciones

En el ejemplo puede observarse:

En primer lugar el byte de comando (04h) y luego (campo sombreado) los argumentos.

Cada campo de 5 bytes de argumentos posee el siguiente significado:

1er y 2do Byte: AA 55 Fijos. A efectos de seguridad se deben introducir estos valores fijos.

3er byte: Número de página. Típicamente byte alto de una dirección de 16 bits.

4to byte: Dirección u offset en la página. Típicamente byte bajo de una dirección de 16 bits.

5to byte: valor a introducir.

Análisis del Ejemplo: 04 AA 55 01 32 1F

1ro y 2do bytes: AA 55 (constante).

3er Byte: 01 – Página 1.

4to byte: dirección 32h.

5to byte: Valor, 1Fh.

Comando 05h: Comando de reinicio de UCs (Reset).

Objeto: El CC ordena el reinicio de una o todas (broadcast) las UCs de un segmento de red.

Argumentos: 2 bytes

Formato (Master):

05 AA 55

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente. Se produce una demora permitiendo enviar la respuesta y luego se reinicia. Este es un comando de uso reservado.

Formato (Slave):

04 AA 55

Observaciones

En el ejemplo puede observarse:

En primer lugar el byte de comando (05h) y luego (campo sombreado) los argumentos.

Cada campo de 2 bytes de argumentos posee el siguiente significado:

1er y 2do Byte: AA 55 Fijos. A efectos de seguridad se deben introducir estos valores fijos.

Comando 06h: Lectura de registros de RAM.

Objeto: El CC lee valores de 8 bits de uno o varios registros consecutivos de la RAM de una UC.

Argumentos: 3 bytes

Formato (Master):

06 01 32 05

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente, sumando la cantidad de bytes de lectura.

Formato (Slave):

06 01 32 05 11 22 33 44 55

Observaciones

En el ejemplo de formato del slave puede observarse:

En primer lugar el byte de comando (06h) y luego (campo sombreado) los argumentos.

Cada campo de 3 bytes de argumentos posee el siguiente significado:

1er byte: Número de página. Típicamente byte alto de una dirección de 16 bits.

2do byte: Dirección u offset en la página. Típicamente byte bajo de una dirección de 16 bits.

3to byte: Cantidad de bytes a lectura.

Análisis del Ejemplo: 06 01 32 05 11 22 33 44 55

1er Byte: 01 – Página 1.

2do byte: dirección 32h.

3er byte: Cantidad a leer, 05.

4to hasta completar cinco bytes (en este ejemplo) con valores: 11h, 22h, 33h, 44h y 55h.

Comando 07h: Provoca eventos en UCs.

Objeto: El CC provoca una cantidad de eventos tipo 02. Util para fines de prueba y verificaciones de sincronismo.

Argumentos: 1 byte

Formato (Master):

07 02

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente y si la dirección es Unicast. No hay devolución si es comando es broadcast.

Formato (Slave):

07 02

Observaciones

En el ejemplo puede observarse:

En primer lugar el byte de comando (07h) y luego (campo sombreado) el argumento.

El campo de 1 byte de argumento posee el siguiente significado:

1er byte: Cantidad de eventos a provocar.

Análisis del Ejemplo: 06 02

1er Byte: 02. Se provocarán 2 eventos.

Comando 08h: Escritura de registros en EEPROM.

Objeto: El CC escribe valores de 8 bits en registros de la EEPROM de una varias UCs. Este es un comando para uso en configuración.

Argumentos: 5, mínimo hasta 242 bytes

Formato (Master):

08 02 32 04 11 22 33 44

Devolución: La UC devuelve el nro de comando, la cantidad escrita y un byte indicador de error que es cero si la acción es exitosa.

Formato (Slave):

08 04 00

Observaciones

En el ejemplo de formato del master puede observarse:

En primer lugar el byte de comando (08h) y luego (campo sombreado) los argumentos.

Cada campo de 5 bytes (mínimo) de argumentos posee el siguiente significado:

1er Byte: Zona. La escritura puede efectuarse en una especie de modo multicast (aunque no es exactamente así porque esto se está resolviendo en la capa de aplicación), dado que la misma información se puede emitir a algunas de las UCs de una red (las que contienen el mismo número de zona) sin afectar a resto.

2er byte: Dirección de EEPROM (máximo 256. Este comando deberá ampliarse en caso de unidades con mayor cantidad de EEPROM).

3er byte: Cantidad de bytes a introducir.

5to byte y sucesivos: valores a introducir en forma consecutiva.

Análisis del Ejemplo: 08 02 32 04 11 22 33 44

1er byte: 02 Zona 2.

2do byte: dirección 32h.

3er byte: 04 Cantidad.

4to hasta completar cuatro bytes (en este ejemplo) con valores: 11h, 22h, 33h y 44h.

En el ejemplo de formato del slave 08 04 00
puede observarse:

En primer lugar el byte de comando (08h) y luego (campo sombreado) los argumentos.

1er byte: 04. Cantidad de bytes escritos.

2do byte: 00. Indicador de error. 00h operación exitosa. Cualquier valor distinto de cero indica el Nro de error que se ha suscitado en la escritura.

Comando 09h: Lectura de registros de EEPROM.

Objeto: El CC lee valores de 8 bits de uno o varios registros consecutivos de la EEPROM de una UC.

Argumentos: 3 bytes

Formato (Master):

09 01 32 05

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente, sumando la cantidad de bytes de lectura.

Formato (Slave):

09 01 32 05 11 22 33 44 55

Observaciones

En el ejemplo de formato del slave puede observarse:

En primer lugar el byte de comando (09h) y luego (campo sombreado) los argumentos.

Cada campo de 3 bytes de argumentos posee el siguiente significado:

1er byte: Número de página. Típicamente byte alto de una dirección de 16 bits.

2do byte: Dirección u offset en la página. Típicamente byte bajo de una dirección de 16 bits.

3to byte: Cantidad de bytes a lectura.

Análisis del Ejemplo: 09 01 32 05 11 22 33 44 55

1er Byte: 01 – Página 1.

2do byte: dirección 32h.

3er byte: Cantidad a leer, 05.

4to hasta completar cinco bytes (en este ejemplo) con valores: 11h, 22h, 33h, 44h y 55h.

Comando 0Ah: Requerimiento de Fecha, Hora y sesgos a la UC.

Objeto: El CC lee el tiempo actual de la UC, conjuntamente con los sesgos de corrimiento de reloj que se registraron en la última puesta en hora. Este comando se utiliza principalmente para fines de diagnóstico y evaluación.

Argumentos: no posee

Formato (Master):

0A

Devolución: La UC devuelve un total de 11 bytes: 1 byte de comando y 10 de argumentos. Los primeros 7 bytes de argumentos son el tiempo actual y se corresponde con el formato que se efectuara en la Puesta en hora (PeH. Ver comando 02h).

Adicionalmente, la UC devuelve tres bytes que permiten evaluar el sesgo que poseía la UC respecto al Patrón de hora en la última puesta en hora.

Formato (Slave):

0A 20 0A 70 0F 17 12 05 04 01 03

Observaciones

En el ejemplo de formato del slave puede observarse:

En primer lugar el byte de comando (0Ah) y luego los argumentos.

Los primeros 7 bytes (sombreado) se corresponden con lo descrito para el comando de PeH (02h).

Los siguientes 3 bytes de argumentos (sin sombreado) poseen el siguiente significado:

1er byte: Número de orden. Sirve para correlacionar el momento que se efectúa la toma del sesgo. Como el comando de PeH se emite en modo broadcast, se supone que todas las UCs involucradas corrigen su hora en el mismo segundo, por lo tanto a efectos de que se puedan vincular sesgos pertenecientes al mismo instante (por ejemplo a fines de estudio), este número es igual en todas las UCs. (Típicamente es el nro de segundo en que ocurre la PeH).

2do y 3er bytes: Diferencia de la fracción de segundo (MOD32) existente entre la PeH y la actual de la UC. Si esta fracción es **positiva,** la UC **adelanta** respecto a la hora de la PeH.

Análisis del Ejemplo: 0A 20 0A 70 0F 17 12 05 04 01 03

1er a 7mo Byte: Idem a lo descrito en el comando 02h.

8vo byte: 04. Numero de orden.

9no y 10mo bytes: 0003. Sesgo. 0103h = 259. La UC adelanta.

Comando 0Ch: Comando Sync (de puesta en hora reducido).

Objeto: El CC (o el Concentrador) normalmente impone la hora a las UCs. El comando de PeH posee una largo de 15 bytes y esto provoca un retardo importante en la UC debido al tiempo de transmisión de la PeH. La UC debe corregir este valor por uno determinado acorde a la velocidad del canal. A efectos de contar con un comando con un número más reducido de bytes se otorga este comando que sólo sincroniza otorgando la frontera del segundo actual.

Argumentos: no posee

Formato (Master):

0C

Devolución: Este es un comando que se emite en Broadcast para todas las UC de la Subred por ende no existe devolución**.**

Formato (Slave): No existe.

Observaciones

Cuando llega este comando a la UC, esta corrige su propio timer adecuándolo al corrimiento de hora provocado por el retardo de la transmisión. El comando Sync se emite siempre por el CC o el Concentrador en el momento que se cumple el segundo, otorgado normalmente por el GPS. A llegada de este comando a la UC siempre se producirá en el instante 0+ del segundo (algunos milisegundos de su pasaje por cero) esto implica que al timer propio siempre se le aplicará un valor positivo y lejando a su valor tope con lo cual no existirán problemas de frontera en la actualización.

Comando 80h: Comando de usuario.

Objeto: Este es un comando de usuario definido específicamente para la aplicación que se trate. En el caso del prototipo que se presenta como ejemplo en esta tesis, se ejecuta un programa de usuario de un semáforo controlable a distancia.

El comando que se utiliza en este caso permite el pasaje de comando manual a automático de 1 de los tres programas con que se parametriza un semáforo. Por ejemplo, permite que el semáforo que en un momento está cumpliendo su rutina de tiempos aplicada a las luces: Verde, roja y amarilla, pueda ser llevado a amarillo parpadeante en señal de alerta. El mismo comando puede volver al semáforo a su proceso anterior.

Argumentos: no posee

Formato (Master):

80 01 01 00

Devolución: La UC devuelve exactamente el mismo comando si este es recibido correctamente y si la dirección es Unicast. No hay devolución si es comando es broadcast.

Formato (Slave):

80 01 01 00

Observaciones

En el ejemplo de formato del slave puede observarse:

En primer lugar el byte de comando (80h) y luego los argumentos.

Los siguientes 3 bytes de argumentos (sombreados) poseen el siguiente significado:

1er byte: Zona. Este comando se puede aplicar a un grupo de UCs de la red (ver lo explicitado en el comando 08h).

2do byte: Modo Automatico/Manual: 00: Modo automatico. 01: Modo Manual.

3er byte: PGM, programa al que se envía la aplicación si el modo es manual. No aplica si el modo es automático.

Análisis del Ejemplo: 80 01 01 00

1er Byte: 01. Se aplicará a las UCs que corresponden a la zona 1.

2do byte: 01. Se envía a la UC a modo manual.

3er byte: 00. Se envía a la UC a programa 0 (Luces amarillas intermitentes).

3.3 CONCLUSIONES

Conforme a lo descrito en el presente capítulo, se observa que se han confeccionado una importante cantidad de comandos a efectos de originar proceso en las UC de la red, quedando una cantidad importante de comandos a implementar en las ampliaciones futuras que puedan efectuarse del protocolo.

CAPÍTULO 4: CAPA DE TRANSPORTE DE COMUNICACIONES EN RS-485 Y ETHERNET

RESUMEN

Dados los requerimientos que se establecieron para la capa de aplicación en el capítulo anterior, se describen a continuación las capas inferiores del protocolo: En el presente capítulo la capa de Transporte y en el siguiente la capa física

Como ya se anticipara, se ha definido un protocolo Master – Slave que a nivel de capa de aplicación, se describió en términos de comandos y sus respuestas asociadas.

En función de ello, el Centro de Control en su rol de Master, debe mantener un sincronismo entre consultas y respuestas, ya que en una red de microcontroladores habría en condiciones normales, varios destinos posibles para una consulta.

La capa de Transporte tiene el rol crítico de proporcionar servicios de comunicación directamente a los procesos de aplicación que se están ejecutando sobre hosts diferentes (Centro de Control y UCs). Este rol permanece invariable para cualquier protocolo de transporte particular por el que se opte. En nuestro caso, los criterios y definiciones serán similares tanto para RS-485 donde se utiliza el denominado Mara-1 o para Ethernet donde se utiliza TCP/IP.

4.1. INTRODUCCIÓN.

Se describe aquí la política de comunicaciones seguida en la red 1, para el caso multipunto (Multi-Drop) bajo norma RS-485, como para el caso de TCP/IP sobre ethernet.

Para el primer caso, se definen pautas que debe cumplimentar el protocolo y en ciertas situaciones (debidamente aclaradas en recuadro), se describirán detalles de implementación sobre una UC desarrollada con microcontroladores, queriendo destacar que un aspecto es la definición abierta del protocolo, la cual es aplicable en forma general y otra cuestión son las adecuaciones instrumentales, que deban efectuarse en la implementación en un microcontrolador en general y más específicamente, sobre uno de una marca particular.

Para el segundo caso, soportado sobre TCP/IP, dado que la especificación de éste es estándar, sólo se describirán las consideraciones que merezca la instrumentación particular, ya que lógicamente, poco puede aportarse sobre su definición, ampliamente abordada por la bibliografía.

Como ya se expresara oportunamente, en la comunicación bajo norma RS485, se ha intentado seguir un modelo en capas de protocolo, tal cual lo recomendado en el modelo de Intercambio entre Sistemas Abiertos (Modelo OSI), en cuanto a lo definido como tareas a realizar en cada capa.

Como es conocido, en el caso de TCP/IP existe una aproximación bastante importante al modelo OSI que exime de mayores comentarios. Por conveniencia y generalidad, se ha querido mantener el mismo concepto en la construcción del protocolo Mara -1, sobre red multipunto.

4.2. CAPA DE TRANSPORTE SOBRE RS-485

Para mantener consistencia de terminología, se utilizará el término **Segmento de Transporte** para referirnos al conjunto de bytes que se emite o recibe en esta capa. Dado que esta capa de comunicaciones es única y embebe de alguna manera la capa de acceso al medio, también podría corresponder denominar como **Marco** al conjunto de bytes a comunicar. Considerándola como más apropiada, en toda la tesis se hará referencia al segmento como la entidad manejable en esta capa.

Dada la relativamente baja velocidad disponible (hasta 115200bps) en una red 485 y en modo half – duplex, es un objetivo vital obtener la más rápida respuesta por parte de la UC direccionada. Por ello, finalizada por una UC particular la recepción del último carácter del segmento entrante y verificada su integridad, se disparan en cascada, en Capa de Aplicación:

a) El análisis del mensaje recibido.

b) El proceso requerido por el comando.

c) La preparación de la respuesta.

d) El envío de la respuesta a la red, con la dirección del Master.

En función de esta estructura, la capa de transporte se compone de dos subcapas: Una **superior** y otra **inferior**.

Implementación: *Esta distinción en dos subcapas, tiene que ver con el hecho que el software constitutivo de la capa inferior, resuelve el tránsito de caracteres (tanto en ingreso como en egreso) mediante una técnica de interrupciones. Esto hace que estas rutinas de la capa inferior, sean totalmente asincrónicas con las de la subcapa superior, donde se resuelve el proceso del segmento dentro del ciclo principal del programa de la UC.*

Por otro lado, la subcapa inferior está orientada a la recepción de un carácter por vez, mientras que la subcapa superior está orientada al tratamiento de la secuencia ***conjunta*** *de caracteres que conforma un segmento.*

Subcapa Inferior

La subcapa inferior maneja el tránsito de bytes con el puerto de comunicaciones, desde o hacia un Buffer de caracteres destinado a contenerlos temporalmente (BufTras).

Este protocolo de la subcapa inferior tiene como misión recibir un segmento completo y depositarlo en el buffer. Dado que no se efectúa ninguna verificación de errores con tal acción, desde este punto de vista el protocolo **no es fiable**.

Segmento de Transporte

En la Figura 4.1 se presenta el formato del segmento. En la misma se destaca en sombreado, la función de los primeros tres bytes de la secuencia (la capa de protocolo de transporte adiciona un total de 7 bytes al mensaje de aplicación), específicamente los relacionados directamente con la subcapa inferior:

Denominación	**Sof**	**Qty**	**Dst**	Src	Data
Cant. de bytes	**1**	**1**	**1**	1	0 a 246

Figura 4.1: Formato del Segmento.

El primer byte, **Sof** (Start of frame) tiene como finalidad indicar el inicio del segmento. El valor de este byte se ha elegido como 1111 1110 b (FEh), por ser un valor de ocurrencia poco frecuente dentro de un segmento (ocasionalmente se lo podrá encontrar dentro del campo de datos o en el campo **BCC:** CheckSum) de 16 bits, trailer del segmento (que es manejado por la subcapa superior). Este carácter se utiliza como preámbulo y determina que todos los nodos de la red que estén en estado de recepción, (se supone que sólo uno de los nodos está en transmisión), sincronicen el comienzo de la lectura del segmento a partir de su detección.

El segundo byte, **Qty** (Quantity) es utilizado para que el receptor pueda determinar cuántos bytes posee el segmento y en consecuencia, dar por concluida la recepción cuando esta cantidad sea alcanzada.

La Qty máxima esperable está determinada por el largo del buffer de transporte, que puede variar acorde a la disponibilidad de memoria de una instalación, pero que no puede ser mayor a 256 bytes. La Qty mínima es igual a 8 bytes y se debe a que se requiere al menos un byte de aplicación y que la capa de transporte agrega 7 bytes de encabezado y terminación de segmento.

El tercer byte, **Dst** (destination) define la Dirección de Red del Nodo destino del segmento. En estas condiciones, se define implícitamente que la red no podrá poseer más de 254 nodos diferentes, puesto que las direcciones 00h y FFh se utilizarán para fines especiales. No obstante y debido a que es recomendación de la norma RS-485 la no existencia de más de 32 estaciones dentro de un mismo segmento de red, se efectúa una reducción de la aquella cantidad, permitiendo como numeración posible de estaciones valores **comprendidos entre 1 y 63.**

Hasta aquí se han efectuado algunas definiciones del protocolo de transporte y a efectos de claridad, se considera oportuno consignar algunos aspectos de implementación.

Implementación de la Subcapa Inferior

Como ya se ha expresado, en esta subcapa se resuelve el tránsito de cada byte por interrupciones y la única verificación que se efectúa, es que se emitan o reciban, todos los bytes que componen un Segmento.

En la recepción, cumplido el ingreso de todos los bytes componentes del segmento, esta subcapa inferior indica mediante la puesta a uno de una bandera (RxTrasFull) que el Buffer está lleno, para que la subcapa superior, haga el proceso correspondiente en cuanto detecte la condición.

En la transmisión, cumplida la preparación de un segmento por la subcapa superior e iniciada la transmisión del primer carácter, la subcapa inferior emite los mismos mediante interrupciones que dispara la capa física, hasta que se llega al último carácter. Desde el arranque de la emisión del segmento y hasta su fin, la subcapa inferior efectúa el manejo de la línea de control (TxControl), que hace el arbitrio del bus Half-Duplex

Recepción

En la Figura 4.2 se presenta el diagrama de flujo de la máquina de estados finitos, que cumple la rutina de interrupciones en la recepción de un segmento.

En el diagrama de flujo de recepción se observa primeramente que si un carácter presenta error - de encuadre o de sobreescritura -, el mismo es desechado y se reinician las condiciones de recepción. Luego de ello si, se trata del primer carácter recibido y es distinto de Sof, el mismo es ignorado. Si un Sof es recibido con éxito se habilita un timer de recepción cuyo funcionamiento se verá luego y se deposita en el buffer de recepción. Al ser el contador de chars (CBufTras) de este buffer distinto de cero, esta situación será detectada en la llegada de caracteres sucesivos

En el diagrama de flujo puede observarse que asumido un caracter como Qty, por ser el segundo byte recibido, se verifica a su vez si su valor está comprendido entre un máximo y un mínimo aceptable. Si esta condición no se cumple se reinician las condiciones a la espera de un nuevo Sof.

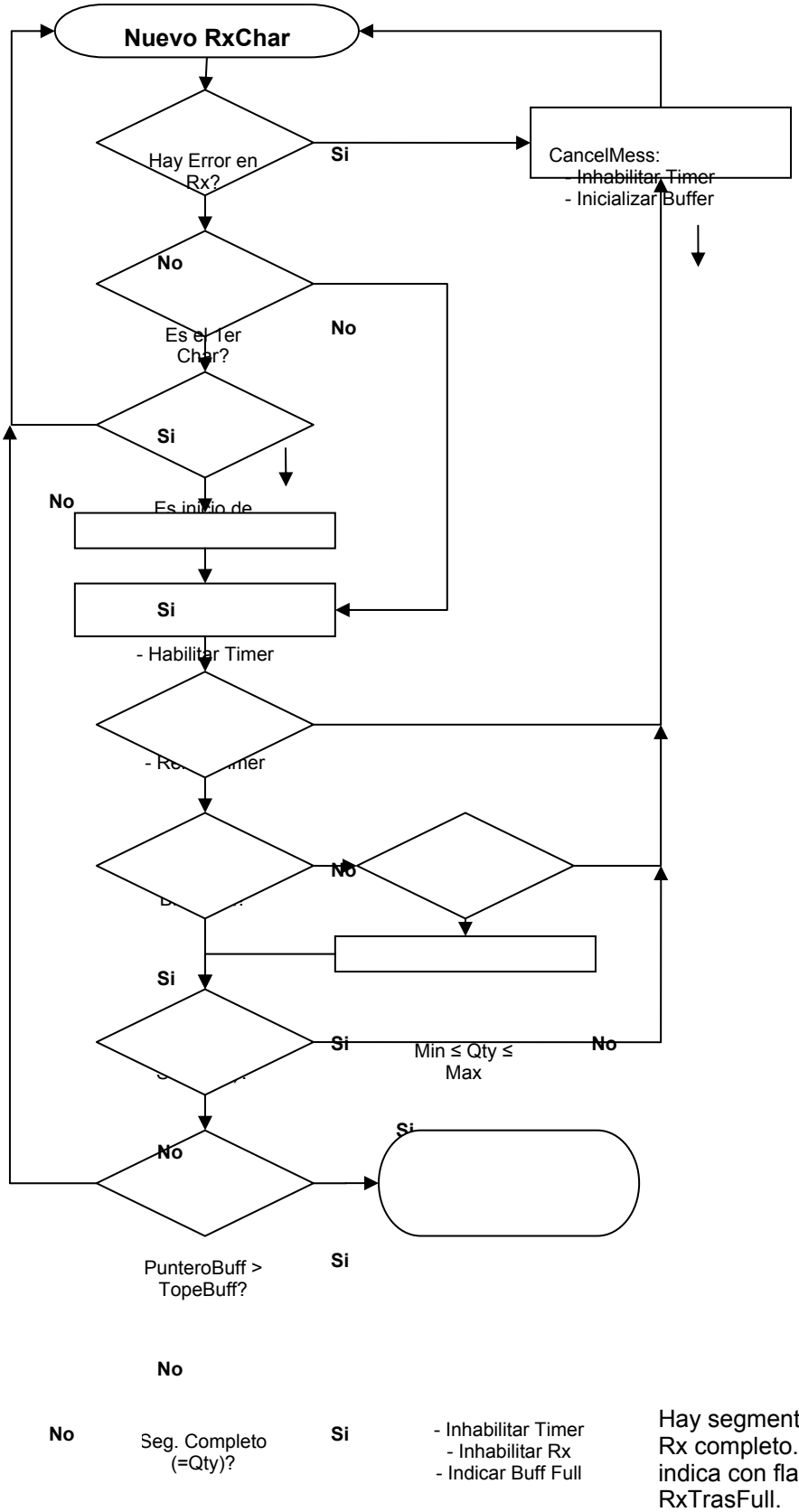


Figura 4.2: Diagrama de flujo de recepción de un caracter.

Cuando se alcanza la lectura del tercer byte (Dst), contado a partir del preámbulo, todos los receptores cuya dirección propia no coincida con aquel valor, abandonan el encolamiento del segmento actual por detección de que no son destinatarios del mismo (sólo uno de los receptores sería eventualmente el destino). Estos receptores reinicializan su condición, como puede apreciarse en el diagrama de flujo, quedando en análisis de los bytes entrantes, a la espera de un nuevo preámbulo que indique el inicio de un nuevo segmento, sobre el cual efectuarán nuevamente la verificación del Dst.

Sólo la unidad que posee una dirección coincidente con Dst, sigue encolando el segmento sin ninguna verificación, hasta su finalización cuando se haya alcanzado la cantidad Qty. Lógicamente podría suscitarse un error justamente en el campo de dirección Dst, ocasionando que una UC erróneamente encole el mensaje entrante, pero esta situación no puede detectarse a priori y solo será tratada en la subcapa superior de transporte como se verá luego.

Transmisión

En la Figura 4.3 se presenta el diagrama de flujo de la máquina de estados finitos, que cumple la rutina de interrupciones en la transmisión de un segmento, cuando éste es dispuesto por la subcapa superior de transporte, que a su vez habilita la transmisión.

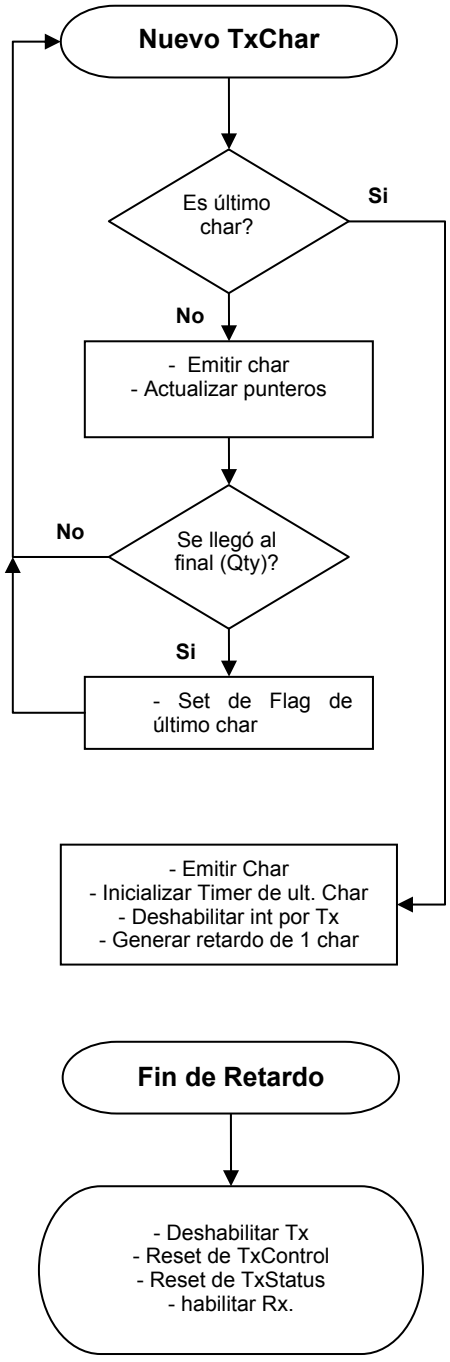
Es un estándar de las UARTs que una vez habilitada la transmisión, inmediatamente se produzca una interrupción (si ésta está habilitada) por registro de transmisión vacío. Esta situación hace que se entre en la rutina de servicio para transmisión de un carácter, lo cual se aprecia en el diagrama indicado.

Si no se está ante el último caracter a emitir, éste se emite actualizando los punteros del buffer de transmisión verificando luego si se han transmitido todos los caracteres existentes.

Si se alcanza la emisión del último caracter, se activa un flag que indica tal situación a efectos de esperar un tiempo determinado antes de inhibir la UART. Esto se hace porque después que la subcapa inferior de transporte escribe el último carácter del segmento, todavía es necesario que este sea serializado, lo cual culmina algún tiempo después. Esta situación tiene alguna complejidad y será descripta a detalle en el tratamiento de la capa física.

Al pie del diagrama de flujo, se simboliza la situación cuando ya ha culminado el retardo iniciado con la emisión del último carácter, produciéndose la interrupción respectiva. Dado que el último carácter ha sido emitido completamente al bus, es necesario deshabilitar la transmisión de la UART, bajar la línea TxControl y habilitar la recepción.

Nótese que la recepción de caracteres, inhibida en el momento de la recepción del caracter de preámbulo, a efectos de tomar mayores precauciones respecto a ruido entrante, se mantienen en esta condición de inhibición mientras hay un mensaje en el interior de la UC y hasta que este sea procesado. La habilitación de recepción se restituye al fin de la transmisión.



Hay nuevo char por interrupción de buffer de UART vacío. También puede ser que sea el último char

Se emitió un segmento completo. Se indica con reset de TxStatus internamente y con TxControl externamente.

Figura 4.3: Diagrama de flujo de transmisión de un caracter.

Subcapa Superior

La subcapa superior tiene como misión extraer los caracteres de encabezado y finalización de segmento y efectuar su verificación en modo recepción o, introducir estos caracteres al mensaje de aplicación, para formar un segmento en el modo transmisión.

Como ya se había anticipado, las rutinas que conforman esta subcapa se encuentran en el ciclo principal del programa de la UC y su ejecución es condicional a indicadores o banderas que proporcionan las capas adyacentes.

Se ha definido que el Buffer de transporte (BufTras) sea un área única de memoria tanto para transmisión como para recepción, dado que en condiciones normales nunca se puede estar en ambos estados simultáneamente por las características sincrónicas del protocolo Maestro – Esclavo. Esto es posible por el hecho que sólo un buffer es utilizado a la vez, haciendo innecesario el uso de memoria en exceso. Obviamente, esto obliga a hacer un uso especialmente controlado, de los punteros de cada buffer para la evolución de la comunicación.

Esta es una situación interesante especialmente en microcontroladores de rango medio y bajo donde el tamaño de RAM para implementación de variables es pequeño (por debajo de 256 bytes).

Continuación de la definición del Segmento de Transporte

En la Figura 4.4 se presenta nuevamente el formato del segmento, destacando ahora principalmente, la función de los bytes 4 y 5 y los dos últimos bytes que ofician de terminación (trailer) de un segmento. Los mencionados están relacionados directamente con la subcapa superior y su función es la siguiente:

El cuarto byte, **Src** (source) define la Dirección de Red del Nodo que actúa como emisor del segmento. Este byte no se utiliza en el modo recepción pero se conserva a efectos de ser intercambiado con el Dst al realizar la transmisión, acción que es efectuada por esta subcapa.

El quinto byte, **Sec** (secuencia) define un número correlativo de las emisiones por parte del CC. Este número permite relacionar un mensaje con su respuesta. El Número de secuencia siempre es generado por el Maestro (el esclavo siempre devuelve el mismo número a efectos de relacionar la respuesta con el requerimiento) y se va incrementando en uno hasta que lleva a un tope máximo de 127, volviendo luego a cero.

Cabe destacar que dado que se trata de un protocolo Maestro – Esclavo, sincrónico y **sin acumulación de mensajes**, sería suficiente con tener un estado binario para la secuencia [11]. Esto es, el emisor emitiría primero 0 como número de secuencia para el primer mensaje y el siguiente mensaje se emitiría con secuencia 1, luego 0 nuevamente y así sucesivamente. Si en algún momento se debe reiterar un requerimiento por no recibir respuesta de la UC que corresponda, **se repite** el número de secuencia anterior. Esto permitiría el ahorro de casi 1 byte en el protocolo. Cabe aclarar que cuando esto fue advertido en la etapa de implementación, ya se encontraba muy consolidado el formato del protocolo definido, por lo que se decidió mantener la situación, dejándola para una mejora futura o utilizar el campo con otros usos.

	Sof	Qty	Dst	**Src**	**Sec**	**Dato**	**BCC**
Cant. de bytes	1	1	1	**1**	**1**	**1 a 246 bytes**	**2**

Figura 4.4: Formato del Segmento.

El campo restante del segmento, exceptuando los dos últimos bytes, es **Dato** (Pay-Load) y conforma los campos de la capa de aplicación del protocolo.

El penúltimo y último byte, **BCC** (Block Check Character) conforman un valor de 16 bits, en formato Big Endian y su tratamiento lo efectúa esta subcapa superior mediante un algoritmo que se explica como sigue:

El campo BCC es conformado por el emisor efectuando el **complemento a uno** de la suma de cada par de los bytes componentes del segmento, incluyendo el Byte de Preámbulo y hasta el último byte del campo Dato, tomándolos en **módulo 64K**, de valores no signados de 16 bits (par de bytes a partir del Sof). De esta forma, el receptor efectuará la suma de todos los Words del segmento de transporte recibido (incluido BCC) y el resultado debiera ser FFFFh para un segmento libre de errores. En el caso que el tamaño del segmento no sea un número par, se agrega a los efectos de la suma de words, un byte en cero a continuación del campo dato. Este byte tiene el sólo objetivo de completar el último Word como byte bajo. Ese byte si bien se toma auxiliarmente a efectos de la suma, no pasará a ser parte integrante del mensaje.

Vale mencionar que esta forma de cálculo del BCC, es la utilizada por el estándar del protocolo TCP/IP, para conformación del campo checksum, en su capa de transporte.

Es una característica de esta capa de Transporte que, - tal cual lo determina el modelo OSI -, otorgue a las capas superiores un mensaje sin errores, listo para ser tratado por la aplicación. Este aspecto es lo que define al protocolo de transporte como un ente **fiable.**

Implementación de la Subcapa Superior

Recepción

En la Figura 4.5 se presenta el diagrama de flujo de la máquina de estados finitos, que cumple la rutina en la recepción de un segmento.

En el diagrama de flujo de la figura se observa que la entrada condicional al proceso está determinada por la bandera **RxTrasFull**. Esta bandera es puesta a uno por la subcapa inferior de transporte cuando se ha completado la recepción de un segmento. El ciclo principal del programa verifica periódicamente si esta bandera es puesta a uno.

Habiéndose verificado que el segmento está destinado a la UC local (debido a que su propia dirección coincide con el campo Dst), se pasa a verificar la integridad del segmento recibido, procesando en campo BCC.

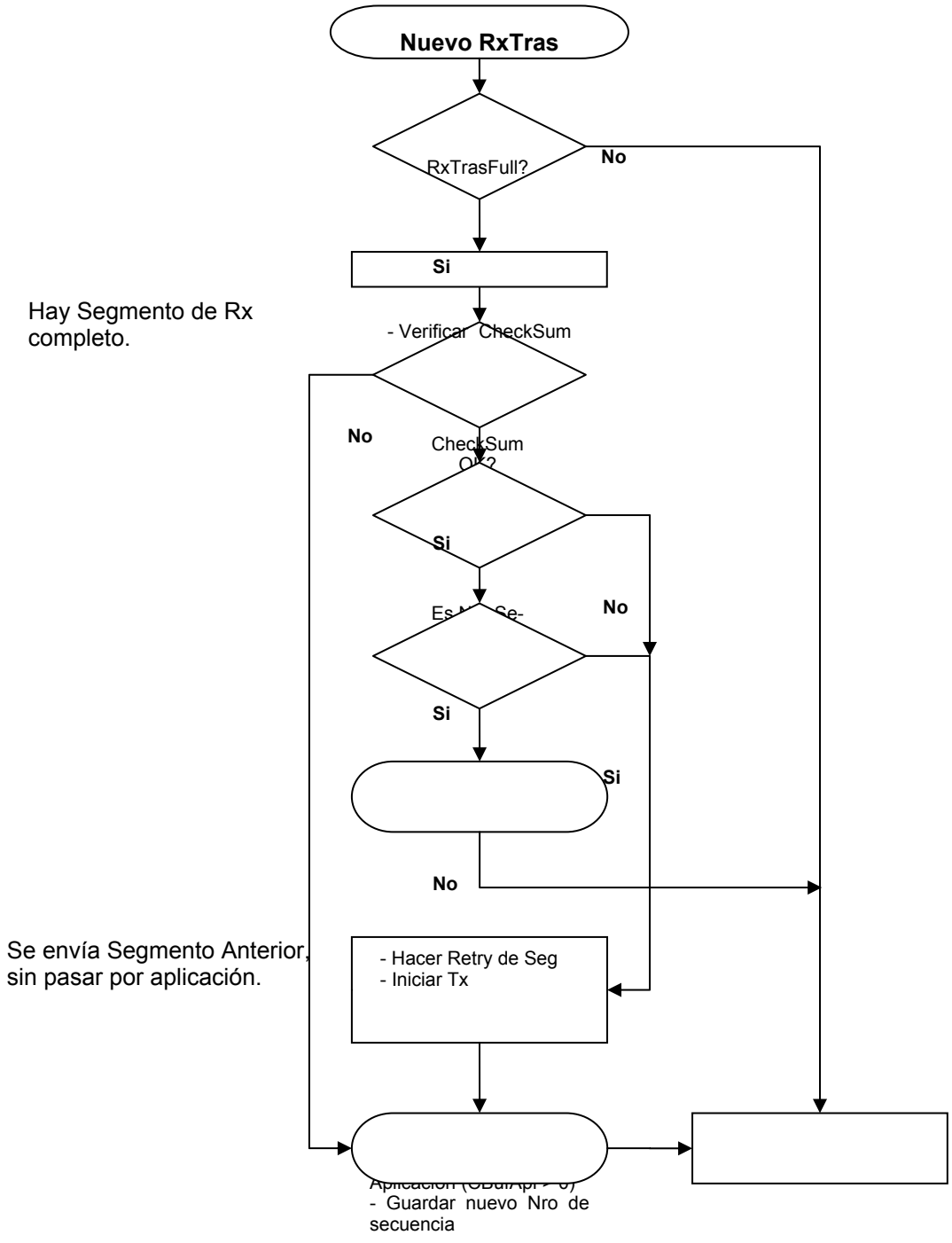


Figura 4.5: Diagrama de flujo de recepción de un segmento.

Si el receptor conformara una suma que resulte distinta al valor FFFFh, consideraría que uno o más bits del segmento se han modificado en el tránsito por la red, dando en consecuencia una recepción errónea del segmento, por lo que desechará el mismo, **sin emitir comunicación al emisor**.

En el diagrama de flujo, observamos que una vez verificada la integridad del mensaje y siendo ésta exitosa se procede a comparar el número de secuencia Sec. Cuando una UC conforma un segmento de respuesta a un comando para su emisión, reserva una copia del mismo en un buffer definido para tal fin. Si esta UC recibiera un requerimiento con el mismo número de secuencia Sec que el requerimiento anterior (numero de secuencia que fuera almacenado), asumirá que el CC receptor detectó un error en su recepción y en consecuencia, ha repetido el comando. La UC en lugar de elaborar una nueva respuesta, efectuará un re-envío (retry) del mensaje almacenado.

Esto es particularmente importante pues en el caso de los eventos ocurridos en la UC, estos se almacenan en un buffer de secuencia de eventos (SOE) y son emitidos al CC cuando este requiere el estado de la UC. Tales eventos se descargan del buffer y los punteros son adecuadamente actualizados no siendo posible repetir la acción. Por lo tanto si la respuesta a un comando de estado no hubiera sido recibida, un nuevo requerimiento consecutivo tiene respuesta favorable, en función de ese resguardo temporal que se efectúa de cada las respuesta emitida, hasta que llega un nuevo requerimiento.

Obsérvese que esta respuesta por parte de la UC es automática ante detección de la igualdad de Sec y se realiza sin intervención de la capa de aplicación.

Si en cambio el número de secuencia es distinto al del requerimiento anterior, se lo considera como nuevo y se pasa el mensaje a la capa de aplicación para su tratamiento. Esto se indica sacando de cero la variable **CBufApl**, en la que se coloca la cantidad de caracteres que contiene el mensaje recibido.

Habiéndose indicado a la capa de aplicación la recepción de caracteres mediante la variable indicada, se procede a guardar el nuevo número de secuencia y a inicializar variables del buffer de transporte. El sistema se encuentra ahora en tratamiento del mensaje por parte de la capa de aplicación, que efectuará la función requerida.

Transmisión

Como ya se indicara en el capítulo correspondiente, la Capa de Aplicación - que en modo de recepción recibe un mensaje consistente y libre de errores-, resuelve el requerimiento efectuado por el emisor, elaborando una respuesta ha ser transmitida en caso de corresponder.

Los mensajes de la Capa de Aplicación poseen un *primer byte* **(Com)** que indica el comando que se quiere ejercer sobre la UC, el cual identifica también su respuesta. Luego, existe un campo variable de datos que son los *argumentos o retornos* (en el requerimiento y la respuesta respectivamente) correspondientes con el comando efectuado.

El mensaje de aplicación elaborado por el emisor - ya sea como respuesta a un requerimiento, o como un requerimiento en si -, es introducido en el Buffer de Aplicación (BufApl) y acto seguido se lanza la rutina que efectúa la preparación del segmento para transmisión, ocupando el Buffer de transporte (BufTras).

En la Figura 4.6 se presenta el diagrama de flujo que cumple la rutina en la transmisión de un segmento.

En una UC la acción comienza con la elaboración de la cabecera. En este caso corresponde destacar que se usará el campo **Src** del requerimiento como destino de la respuesta.

Para la elaboración del segmento - dado que se usa la misma área de memoria para el buffer de transporte en transmisión y recepción y que no se trata de un buffer circular porque la ocupación no es simultánea -, algunos de los campos que se reciben en un requerimiento pueden reutilizarse en la respuesta y ya se encuentran en el lugar adecuado (caso de Sof y Sec). Otros campos (como Src y Dst) simplemente se conmutan. El campo Qty en cambio, se elabora con la cantidad de caracteres del mensaje obtenido de la aplicación sumando el valor 7, que es la longitud que se adiciona como cabecera y trailer, para conformar un segmento de transporte.

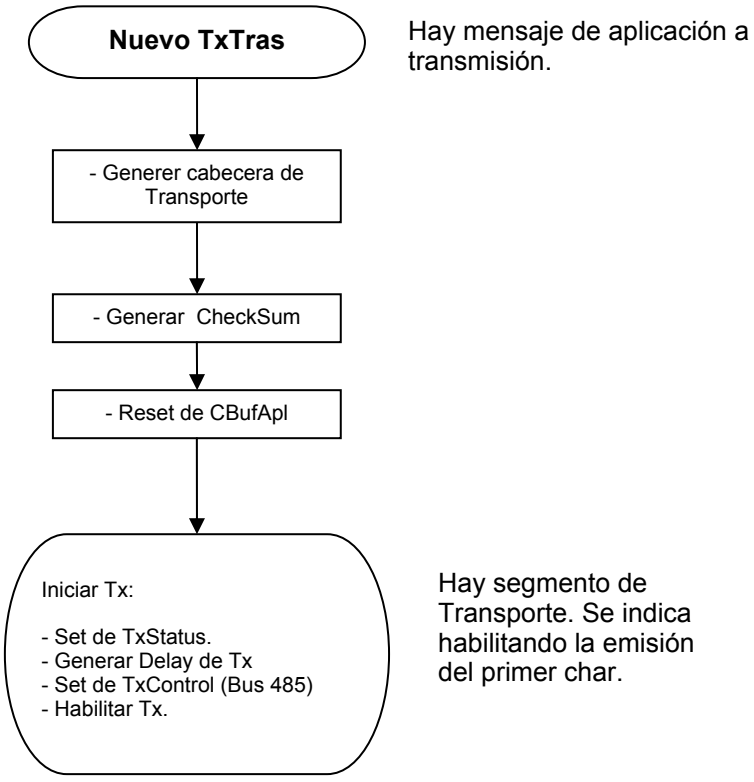


Figura 4.6.: Diagrama de flujo de transmisión de un segmento.

Una vez confeccionada la cabecera del segmento, se está en condiciones de elaborar el BCC que ocupará los dos últimos bytes del segmento.

Como ya se dijera, el campo BCC es conformado por el emisor efectuando el **complemento a uno** de la suma de cada par de los bytes componentes del segmento, incluyendo el Byte de Preámbulo y hasta el último byte del campo

Dato, tomándolos en **módulo 64K**, como valores no signados de 16 bits (par de bytes a partir del Sof).

A continuación se da inicio a la transmisión habilitando la UART para transmitir mediante la mecánica de interrupciones que ya se ha descrito, llevada a cabo por la subcapa inferior de transporte.

TimeOuts

La técnica de comunicaciones bajo norma RS485 es Half-Duplex y como consecuencia, el Master con sus consultas, fija el sincronismo del flujo de bytes en el canal.

El protocolo maestro – esclavo determina que un esclavo (la UC) sólo responde ante el requerimiento de un maestro. Por otra parte si un esclavo no interpreta un mensaje o si de interpretarlo constata que no tiene integridad, simplemente lo desecha sin emitir ninguna comunicación al maestro. Esto es así porque de otra forma podrían existir múltiples esclavos queriendo responder concurrentemente por el hecho potencial de no haber recibido un mensaje adecuadamente, lo cual es una situación inaceptable.

Esta escasa realimentación entre los integrantes de la red coincidentemente con la necesidad de aprovechar al máximo la velocidad de un canal no muy rápido, hace que se deban limitar al extremo los tiempos ociosos de espera de mensajes o incluso entre los bytes que conforman estos mensajes.

Para ello, se definen diferentes intervalos de espera que se deben instrumentar en el software de cada uno de los nodos, a saber:

1. **TO1 - En el Nodo Slave:** El Slave sólo responderá cuando se haya recibido un mensaje consistente. Por ende, no habrá requerimientos de re-envío por parte del Slave. A efectos de tener certeza en la incorporación de los bytes que componen un segmento entrante, el Slave lanzará un Time-Out a partir de la recepción del byte de preámbulo. Este Time-Out se reiniciará ante recepción de cada byte.

   Si entre byte y byte, hasta que se reciba la cantidad indicada por **Qty,** hubiere un tiempo T1, mayor que el TO1 definido (que en principio se puede establecer del orden de duración de la transferencia de 1 byte, conforme a la velocidad seleccionada), el mensaje completo se descartará, iniciando nuevamente la secuencia de espera de un SOF.

   Como se verá luego en el análisis de la capa física, es una característica de una UART (al menos en procesos con un ciclo muy corto) el envío consecutivo de caracteres en un mensaje, sin que medie casi ningún intervalo entre el stop bit de un byte y el start bit de su consecutivo. Debido a ello este Time-Out de espera entre byte y byte podía ser muy inferior a la duración de un byte de acuerdo a la velocidad. No obstante, típicamente se lo ajustará a un intervalo equivalente al tiempo de transmisión de un byte

2. **TO2 - En el Nodo Slave:** Antes de emitir una respuesta se efectuará un retardo, normalmente de una duración mayor al Time-Out TO1 definido en el apartado anterior, a efectos que nunca puedan concatenarse un requerimiento y una respuesta como un mensaje único. De esta forma el inicio de un segmento estará mejor definido ya que comienza con un SOF

(FEh). La ocurrencia del mismo carácter en el interior de alguno de los campos del segmento, en condiciones normales no podría confundirse con un SOF dado que tal ocurrencia será a continuación de otro carácter y antes del vencimiento del TO1.

Mas adelante se efectúan algunas consideraciones acerca de determinadas situaciones que pueden darse en la práctica y como ellas son salvadas.

3. **TO3 - En el Nodo Master:** un Time-Out que fijará la ventana temporal de respuesta del Slave. Si el Slave no responde dentro de la ventana prefijada (inicialmente, un tiempo mayor que el de duración de la transferencia de uno o dos caracteres conforme a la velocidad seleccionada), el maestro repetirá el mensaje con el mismo número de secuencia. Si no hay respuesta por ejemplo, a tres mensajes con la misma secuencia enviados, se considerará el nodo en falla o una falla de comunicaciones y se abortarán las consultas a ese nodo.

En la Figura 4.7 se observa un diagrama donde se representa en la zona superior la transmisión de un mensaje por parte de una unidad 1, actuando como master. El mensaje transmitido que simboliza un requerimiento emitido por la unidad 1 y con destino a la unidad 2, se ha simplificado a dos bytes de valor arbitrario 5Dh.

En consistencia con los dos bytes mencionados, se representa el diagrama de la línea TxControl que controla la transmisión al bus RS485 y que en este caso es controlado por el mismo emisor. Obsérvese que la mencionada línea se pone activa conjuntamente con el inicio del Start bit del primer carácter y se desactiva al término del stop bit del segundo carácter emitido.

A la conclusión del mensaje emitido, la unidad 1 se pone en recepción y la línea de transmisión se pone en reposo. En el instante que termina la emisión por parte de la unidad 1 que actúa como Master, ésta lanza su temporizador con un Time-Out de valor TO3.

En la zona inferior del diagrama se representa el estado de la unidad 2, que se encuentra a la recepción en tanto dura la transmisión de la unidad 1.

En el diagrama se ha representado el tiempo T1 aleatorio que media en la emisión entre byte y byte (el cual se ha magnificado por claridad). En relación con este se representa el Time-Out TO1 que lanza el receptor al término del stop bit de cada byte recibido.

En condiciones normales, para que sea posible una recepción exitosa por parte de la unidad 2, del segmento emitido por la unidad 1, se cumplirá:

$$\mathbf{TO1 >> T1}$$

Luego de recibido el mensaje y producido el retardo TO2, la unidad 2 emite su respuesta. No se han representado los retardos del proceso interno en esta última unidad y la respuesta se ha limitado a un byte por simplicidad en el diagrama.

En condiciones normales y a efectos que no se repita el requerimiento que emite la unidad 1 antes que llegue el primer byte de la respuesta emitida por la unidad 2, se cumplirá:

$$\mathbf{TO3 >> TO2}$$

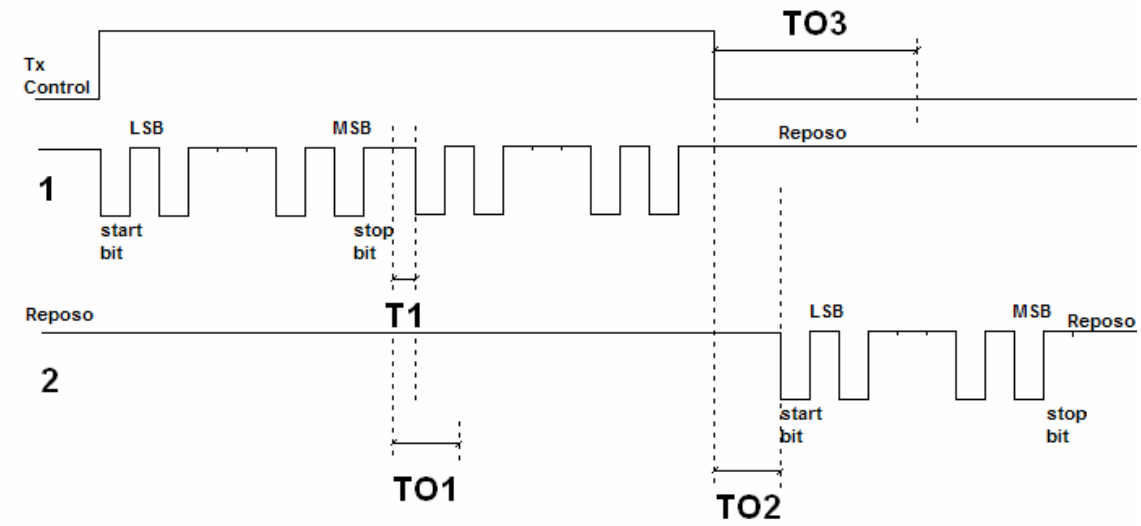


Figura 4.7.: Diagrama combinado de bits de Transmisión y Recepción en RS232.

Puede observarse entonces, - considerando el funcionamiento cooperativo de estos tres mecanismos enunciados -, que si el Slave aborta la recepción, ya sea porque no recibe el mensaje en tiempo (tiempo entre un byte y otro mayor que el tiempo de espera TO1 prefijado), o que calculado el Check-Sum el mismo no fue satisfactorio, en ambos casos, la porción del segmento hasta ese momento recibido se desecha, sin que medie indicación alguna por parte de la UC.

Estas situaciones conllevarán a la misma acción por parte del Master: Repetir el requerimiento luego de vencido su propio Time-Out TO3, pues asume que esta demora excesiva, surge de algún problema en la recepción de su segmento y que ya no habrá respuesta.

Tratamiento de errores

Como se ha descrito anteriormente, la capa de transporte del protocolo, cuenta con mecanismos que permiten convertir en fiable el mensaje recibido y dirigido a la capa de aplicación.

Estos mecanismos son:

a. Preámbulo indicador del comienzo de un segmento e indicación del largo del mismo.

b. Time-Out entre byte y byte en la subcapa inferior de transporte.

c. Verificación de Check-Sum en la subcapa superior de transporte.

d. Verificación de Nro. de secuencia en la subcapa superior de transporte.

Análisis de fallas

En base a lo expuesto sería razonable analizar como es el comportamiento ante determinadas situaciones para evaluar la robustez de estos mecanismos, funcionando ellos en forma independiente o combinada.

Como ya se anticipara los segmentos de transporte comienzan con un carácter de preámbulo (FEh) que indica el comienzo de un segmento y la secuencia de recepción. Detectado este carácter por todos los receptores asociados al mismo canal de multidifusión, los mismos toman el siguiente caracter **Qty** como largo del mensaje a aceptar y luego la dirección **Dst** de destino. Si esta dirección de destino no es la del receptor, el segmento hasta aquí introducido al buffer, se desecha.

Analizando más profundamente dentro del campo de intrincadas posibilidades, vemos que puede suceder que el valor FEh ocurra en el medio de un segmento (puesto que el caracter puede encontrarse dentro del campo de datos), por lo que algún receptor que esté a la espera de ese indicador podrá, erróneamente, considerar que se encuentra en el comienzo de un segmento genuino. En este caso, tomaría los dos bytes siguientes como cantidad y dirección de destino. Avanzando aún más con el análisis, un determinado receptor podría ver la coincidencia entre **Dst** y su propia dirección (el resto de las UC desecharía el incipiente segmento) y como consecuencia seguiría acumulando caracteres hasta la cantidad **Qty**. De haber suficientes caracteres (y no provocarse un Time-Out de recepción), puede ocurrir que la secuencia de caracteres del mensaje, alcance para completar un segmento completo en el receptor pero, el receptor evaluará luego el Check-Sum, que dará erróneo, desechando en consecuencia el mensaje.

En otro caso, los caracteres remanentes del mensaje original no alcanzarán para completar un segmento por lo que al no llegar más caracteres, funcionará el Time-Out de la UC, desechándose el segmento falso.

A efectos de tener controlado el arbitrio del bus, el Master lanzará un Time-Out al final de la emisión de su requerimiento. Cuando un receptor detecte que no es para si el segmento, - debido a la evaluación de la dirección de destino o, en otro caso, que existe un error en la recepción -, no existirá respuesta. Consecuentemente, esto llevará al Master (vencido su propio Time-Out de emisión), a la repetición de la consulta.

En el Master en cambio, un chequeo que detecte un error en el mensaje recibido del Slave, implicará la repetición del requerimiento por su parte, con el mismo número de secuencia y el reinicio de la cuenta de re-envíos conforme a lo que ya se indicara.

El receptor correspondiente, al ver un número de secuencia igual al almacenado por la consulta anterior, repetirá la emisión del mensaje dentro de su Capa de Transporte, sin intervención de las capas superiores, debido al uso de su Buffer de retransmisión.

Se entiende así, que con estos mecanismos, sumados a la calidad inherente que posee un canal de difusión con par trenzado en la capa física, con un protocolo que arbitra la ocupación del canal,- lo que redunda en una muy baja probabilidad de colisión-, se tiene un esquema de comunicación muy robusto y muy seguro.

4.3. CAPA DE TRANSPORTE SOBRE ETHERNET

Otra de las alternativas que va cobrando fuerza como mecanismo de comunicación en los sistemas embebidos, es la utilización de Ethernet como capa física [5][10] . Los Sistemas Embebidos en Red (Embedded Network Systems) son un paradigma con alcances insospechados, fundamentalmente debido a la ubicuidad de las redes e internet y por ello no podría ser dejado de lado el análisis

bajo esta tecnología, dado que Ethernet parece ser una de las formas óptimas de comunicación para una gran mayoría de casos, sustituyendo dispositivos y protocolos de comunicación, ahora obsoletos.

La implementación sobre Ethernet es prácticamente transparente en función que se ha definido también la utilización de TCP/IP en las capas superiores de comunicaciones. La mayoría de los fabricantes de microcontroladores [13] proveen hardware basado en tecnología 10BaseT o 100BaseT con los estándares de la norma IEEE 802.3 y a su vez proveen un “stack” de protocolos [14] fácilmente implementable como parte del software a ser instrumentado en las UCs.

En nuestro caso, si bien no se efectuó una implementación de ensayo sobre una UC (aunque si una muy parecida sobre el Concentrador), se define el hecho de mantener el protocolo de la capa de aplicación tal cual se lo ha definido en el capítulo anterior. Esto debe ser así, dado que la aplicación define comandos relacionados con funciones, que son invariables e independientes del método que se utilice para comunicar las unidades entre sí.

En función de lo expuesto, el mensaje de aplicación así definido se entrega a la capa de transporte en TCP/IP para su encapsulamiento y a partir de allí el paquete obtenido recibe el tratamiento que es estándar en la norma. Detalles de la implementación de esta tecnología se otorgan en el capítulo correspondiente a la aplicación del Concentrador de Comunicaciones.

CAPÍTULO 5: CAPA FÍSICA DE COMUNICACIONES EN RS-485 Y ETHERNET

RESUMEN

Dados los requerimientos planteados en la capa de aplicación, se han definido funciones que debe cumplimentar el software desarrollado para la capa de transporte. En está última capa se fija entonces, el funcionamiento que se exigirá a la capa física, ajustándose al funcionamiento de los componentes involucrados que poseen una fabricación normada por un estándar.

En el presente capítulo se describe a detalle la capa física, inferior de la estructura que se ha planteado, haciendo hincapié en algunas de sus características y particularidades.

Se verán a detalle los aspectos involucrados en una red funcionando bajo norma RS-485, sobre la que se efectúa una implementación y con un detalle menor los aspectos involucrados en una red Ethernet ya establecida por un estándar de amplia difusión.

5.1.1 INTRODUCCIÓN

La red de interconexión RS-485, es la primera que se ha tenido en cuenta en esta tesis para una red de interconexión de microcontroladores, por varias razones a saber:

- Satisface los dos requerimientos dados para la red física de interconexión ya mencionados anteriormente: capacidad de multidifusión (*broadcast*) y distancia de varias decenas de metros. De hecho, dependiendo de la velocidad de comunicación se pueden interconectar dispositivos a varios cientos de metros de distancia (900 mts por norma). Si en cambio se involucran en los segmentos de red otros dispositivos menos susceptibles a ruido eléctrico – fibra óptica, por ejemplo -, los segmentos se pueden extender prácticamente en forma ilimitada con repetidores transparentes.
- Está ampliamente probada en entornos industriales de producción y reconocida como una de las más inmunes a ruido (básicamente por ser *balanceada*) [8] [9].
- En el caso de dispositivos interconectados, la capacidad de comunicación es muy satisfactoria, al menos en términos relativos de rendimiento de los microcontroladores. Se debe tener en cuenta, que los microcontroladores que están en contracto directo con el campo de aplicación no tienen, en general, gran capacidad de procesamiento sino que están orientados a tener muy buena respuesta en tiempo real, manejando directamente sensores y actuadores.
- Puede ser controlada/manejada de manera directa desde la inmensa mayoría de los microcontroladores, por puertos de entrada/salida de los mismos. Efectivamente, la mayoría de los microcontroladores (sino todos), normalmente incluyen **embebida** al menos una UART (Universal Synchronous Receiver / Transmitter), o incluso una USART (Universal Synchronous / Asynchronous Receiver / Transmitter), que adicionan la capacidad de comunicación serie sincrónica.

La red (o bus) RS-485, en la actualidad norma EIA-485, es una especificación eléctrica de capa 1 (física) del modelo OSI, para comunicaciones serie half-duplex multipunto. Esto significa que en una comunicación entre dispositivos, donde sólo uno de los integrantes de la red puede transmitir (Tx) y el resto debe estar en recepción (Rx). Para que esto sea posible, los instantes de transmisión de cada dispositivo deberían estar controlados por la propia evolución de las comunicaciones de los dispositivos interconectados, que hace que las emisiones simultáneas sean muy poco probables, por ejemplo siguiendo un protocolo Master-Slave. La definición del estándar RS-485 otorga las condiciones para que exista éxito de comunicación en una configuración half-duplex (un dispositivo envía y el o los demás dispositivos deberían recibir) y no se establecen condiciones para un tipo de error muy posible, que se suscitaría en caso que los dispositivos efectuaran contienda o puja por la utilización del bus. Este error se denomina ***Line Contention*** y se debería evitar por alguna forma de administración/control de las comunicaciones.

A partir de este punto, red 485, bus 485 y RS-485 se usarán como sinónimos de la especificación EIA-485, a menos que se aclare específicamente otra cosa.

5.2. REQUERIMIENTOS Y CARACTERÍSTICAS

En principio, los dos requerimientos más importantes para la interconexión (red física) de los microcontroladores son:

1. *Capacidad de multidifusión de la red*. Esto implica que la red física debe tener la capacidad de que - como mínimo para algunas comunicaciones -, uno de los dispositivos envíe un dato que pueda ser recibido de manera simultánea por todos los demás dispositivos conectados a la red. La capacidad de multidifusión, está directamente orientada al mantenimiento del sincronismo sin algoritmos muy complejos, utilizando comunicaciones de tipo *broadcast* (o *colectivas*) y además, podría ser utilizado para comunicar eventos muy importantes y/o críticos en el tiempo, para los cuales las conexiones *punto a punto* pueden resultar ineficientes o en peor caso, inútiles.
2. *Distancia máxima de al menos centenas de metros*. Dado que en principio no está determinado el tipo de sistema a monitorizar y/o controlar con la red de microcontroladores, se tiene en cuenta una distribución física relativamente amplia. No se espera que con esta distribución (del orden de las centenas de metros) se cubran todas las alternativas posibles, sino que se tenga una red versátil en cuanto a que cubra al menos, los entornos de máquinas herramientas (robots, por ejemplo) en una planta o ambiente industrial o de producción relativamente reducidos (del contexto de una red local, por ejemplo).

Otros requerimientos, que deben ser tenidos en cuenta aunque que no se establecen requerimientos a priori sobre ellos, serían:

3. *Inmunidad al ruido*. Está claro que mientras se tenga mayor inmunidad a ruido tanto la red como los mecanismos de comunicación sobre la misma serán más robustos (con baja frecuencias de fallas). Se debe tener en cuenta que pueden haber aplicaciones con entornos de comunicación con muy poco ruido (redes a campo abierto, como en el caso de sistemas de control de riego o compuertas en sistemas hidráulicos) y otras en ambientes industriales de mucho ruido para las comunicaciones digitales (Control de alimentación eléctrica, rectificadores para procesos de electrólisis).
4. *Capacidad de comunicación/rendimiento en ancho de banda y/o latencia*. Sin lugar a dudas, siempre es mejor contar con una red de alto rendimiento en cuanto a la capacidad de comunicar datos por unidad de tiempo o el tiempo mínimo para una transferencia de datos. Sin embargo, se debe recordar que los sistemas a monitorear y/o controlar pueden tener requerimientos de tiempo muy dispares, dependiendo de la aplicación. Es por esta razón que no se podría establecer a priori un determinado rango de requerimientos. Por otro lado, siempre se debe tener en cuenta que el rendimiento de la red de comunicaciones se puede evaluar *relativo* a la capacidad de procesamiento de los microcontroladores que se interconectan. No tiene mucho sentido en realidad, tener más capacidad de comunicaciones de datos, si los procesadores no pueden ni siquiera enviar o recibir por tener baja frecuencias de reloj, por ejemplo.

Asimismo, se tienen varias características a tener en cuenta para la red física de interconexión de microcontroladores, ahora con la decisión de utilizar RS-485 para la capa física de comunicaciones y también utilizar la capacidad de los microcontroladores de transmisiones serie asincrónicas:

5. *La red de microcontroladores debe tener la capacidad de interconectar varios microcontroladores con capacidad de multidifusión*. Esto implica que normalmente habrá más de dos dispositivos interconectados y que las comunicaciones pueden ser desde uno cualquiera de ellos hacia otro/s.
6. *La especificación de Rs-485 no incluye la forma en que se controla (o se asegura) la comunicación half-duplex.* Se debe establecer una forma de comunicaciones que evite Line Contention sobre el bus 485. La consecuencia directa, es que no se define ninguna parte de la red física para evitar Line Contention, las colisiones se deben evitar en alguna de las capas superiores.
7. *Se tiende a utilizar la capacidad de comunicaciones serie incorporada (integrada) en los microcontroladores.* Ello con puertos de comunicaciones basados en UART o USART. Esto implica la utilización de algún mecanismo de conversión de un puerto diseñado para comunicaciones serie, full-duplex, que posee una línea de Rx y otra de Tx, a un esquema como el propuesto por la norma RS-485, donde hay un único par de conductores, balanceado, por donde se debe transmitir y recibir. Concretamente, se debe convertir un puerto serie asincrónico full-duplex al manejo de señales eléctricas half-duplex que propone RS-485.

5.3. DEFINICIÓN DE COMPONENTES EN RS-485

Interfase con la red

Afortunadamente, uno de los dispositivos físicos que cumple con tales requerimientos es el circuito integrado SN75176B de Texas Instruments [9] y una serie similar de otros fabricantes que cumplen el mismo estándar. En función de esta multiplicidad de fuentes de dispositivos similares, haremos mención del dispositivo nombrado.

Al definir la utilización del SN75176B a nivel físico, se debe establecer el tipo de señal/es de control y de datos a utilizar desde los microcontroladores, que el propio dispositivo se encargará de *traducir a* RS-485. Esta/s señal/es tienen relación directa con la forma de utilización de los puertos de los microcontroladores para entrada/salida que incluyen al menos una UART.

Puesto de otra forma, se debe administrar la red RS-485 desde el SN75176B utilizando lo que sea necesario de un microcontrolador, que incluye dos puertos de entrada/salida para transmisión y recepción vía UART. El SN75176B posee los pines de R y D que corresponderían a Rx y Tx respectivos de una UART y además, dos pines para indicar el *sentido* de la comunicación half-duplex:

- DE, que lo habilita para envío de datos hacia la red RS-485.
- $\overline{\text{RE}}$ que lo habilita para recepción desde la red RS-485.

La Figura 5.1 muestra esquemáticamente el SN75176B, en lo que respecta a su interfase con un microcontrolador o en general, con cualquier dispositivo que lo utilice para enviar o recibir información vía una red RS-485.

Se aprecia que dado que el bus es un par de conductores, debe efectuarse conmutación en el dispositivo que intente recibir o transmitir. La transmisión o escritura sobre el bus es una acción que amerita baja impedancia eléctrica de salida para imponer el valor. Esto se ve evidenciado por el buffer saliente que controla el pin D. En la recepción en cambio el dispositivo posee alta impedancia eléctrica y esto

permite que haya varios dispositivos que se encuentren simultáneamente leyendo del bus.

Tanto la entrada como la salida en las líneas A y B, sobre el bus 485 son balanceadas. Y los buffers electrónicos tienen como misión adicional, adaptar la entrada / salida balanceada sobre el bus 485 a una entrada / salida desbalanceada, con hilo de masa común, característicos de una UART.

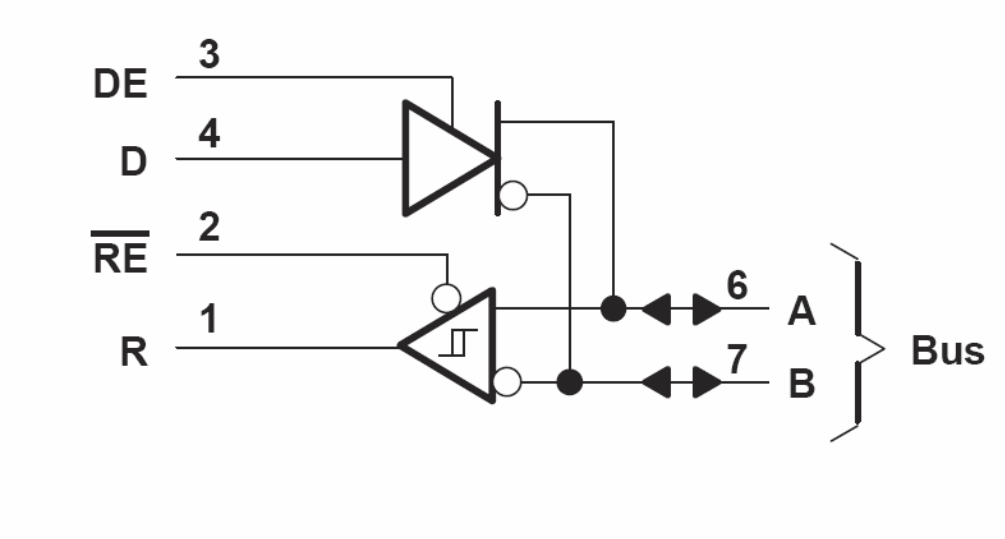


Figura 5.1.: Diagrama en bloques del SN75176B.

La Figura 5.2 nos otorga la tabla de verdad del dispositivo y es importante tenerla en cuenta porque en función de ella deben agregarse algunos componentes de hardware menores y deben efectuarse diferentes acciones sobre el software controlador de la capa inmediata superior (inferior de transporte).

En la parte superior de la figura observamos la situación que se presenta en la transmisión. Se presenta el estado de tensión relativo de las líneas A y B del bus en función de de el estado de la variable D de entrada.

En particular, se observa que cuando el buffer está deshabilitado (linea DE en estado bajo), la salida está en alta impedancia, virtualmente desconectada del bus. Como se verá luego, éste es el estado “por defecto” en que debe mantenerse este driver, a efectos que no provoque inconvenientes a los otros dispositivos cuando no se quiere transmitir, dado que si se lo mantiene en baja impedancia de salida, para los otros drivers presentes en la red es un cortocircuito que absorbe la señal.

En la parte inferior de la figura observamos la situación que se presenta en la recepción. Vemos cual es el estado lógico de la línea de salida R en relación con la tensión diferencial vista desde A hacia B (VA – VB = V_{AB}). En primer lugar se observa que debe existir a la entrada una tensión que supere un umbral que ronda en los 0.2 volts, a afectos no exista indeterminación del valor entrante.

Por otro lado se observa que la salida es indeterminada si la entrada está abierta o que queda en alta impedancia cuando el buffer está deshabilitado (RE = High). Estas

condiciones de indeterminación debe ser tenidas en cuenta, para actuar como entrada a un microcontrolador.

DRIVER

INPUT D	ENABLE DE	OUTPUTS	
		A	B
H	H	H	L
L	H	L	H
X	L	Z	Z

RECEIVER

DIFFERENTIAL INPUTS A–B	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0.2$ V	L	H
-0.2 V $< V_{ID} < 0.2$ V	L	?
$V_{ID} \leq -0.2$ V	L	L
X	H	Z
Open	L	?

H = high level, L = low level, ? = indeterminate,
X = irrelevant, Z = high impedance (off)

Figura 5.2.: Tablas de verdad de los bloques del SN75176B.

Red física

En la Figura 5.3, se presenta un esquema de conexión de varias interfases a un bus 485. Según establece la norma RS-485, es de 32 la cantidad máxima de dispositivos que pueden asociarse a un segmento de hasta 900 mts de longitud.

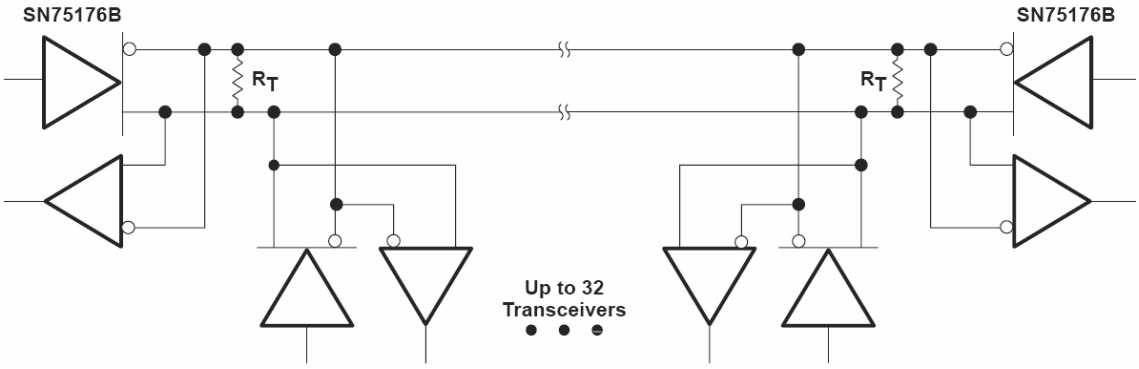


Figura 5.3.: Detalle de conexión de unidades a una red bajo norma RS 485.

La red bajo norma RS-485 es una red en bus. Esto significa que cada unidad se conecta a un único bus de dos cables en un modo pasante. En la práctica el cable debe abrirse para que una unidad pueda ser conectada, pero la conexión se resuelve en un modo “guirnalda” o “daisy Chain” de dispositivos interconectados.

En los dispositivos que se encuentran ubicados en los dos extremos del bus deben instalarse unas resistencias de terminación, adaptadoras de impedancia, que en la figura de las denomina como $\mathbf{R_T}$.

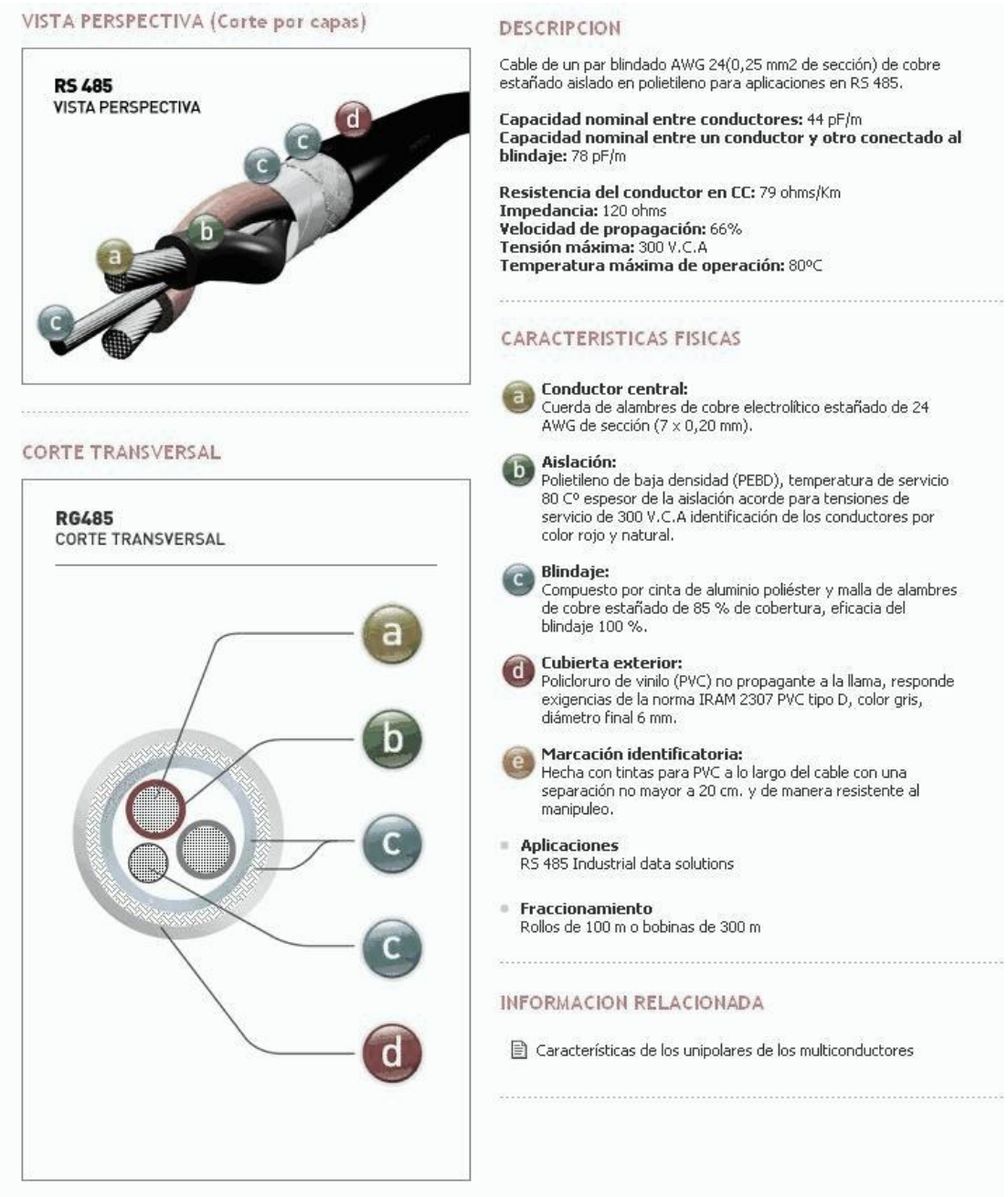


Figura 5.4.: Cable apantallado para red bajo norma RS 485.

En la Figura 5.4, se presenta una especificación técnica de un cable estándar para red 485, que responde a la denominación AWG 24. Como puede apreciarse en la figura, el mismo es un cable apantallado con un par trenzado en su interior. Se trata de un cable tipo STP (Shielded twisted pair) con una impedancia característica de 120 Ohms.

5.4. IMPLEMENTACIÓN DE LA RED

Interfase con el microcontrolador

En la Figura 5.5 se presenta un esquemático que además de incluir el CI SN75176B, incorpora unas resistencias eléctricas que poseen la siguiente función:

1. La resistencia R1 (típicamente de valor 10Kohm), es la que mantiene en estado 1 (reposo) a la línea Rx de entrada a la UART tanto si el bloque está deshabilitado ($\overline{RE}$ = High), como si la entrada A y B está desconectada, caso que en la práctica puede presentarse cuando la línea está cortada o si el circuito integrado conversor no está presente o está averiado. No sería deseable que la UART estuviera fuera de reposo ante estas anomalías.
2. La resistencia R2 (típicamente de valor 10Kohm), es la que mantiene al driver de transmisión en estado de alta impedancia de salida cuando no hay orden de entrada o cuando el microcontrolador está en algún estado transitorio o incluso ausente. No es deseable que el bus esté en baja impedancia pues esto provoca que **ninguna** de las unidades de la red pueda transmitir.

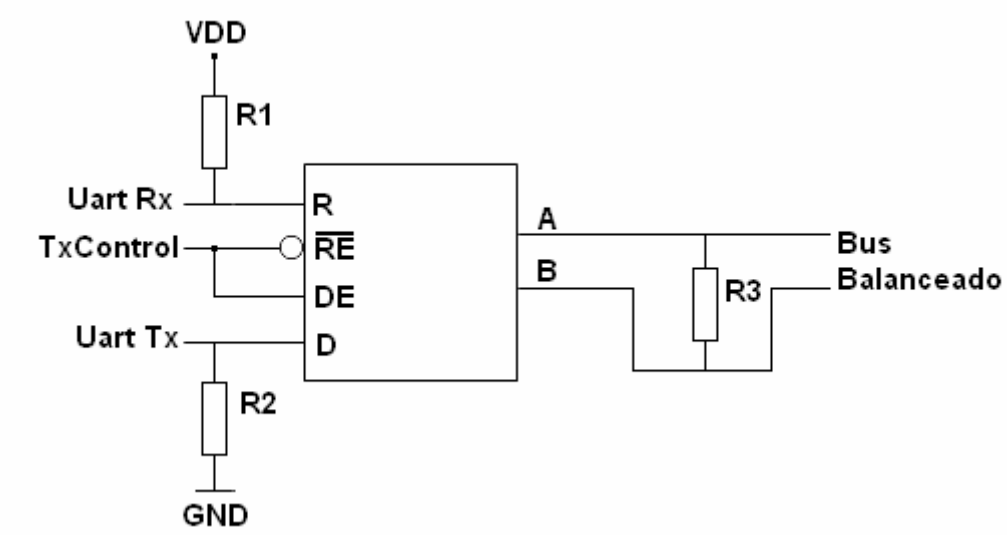


Figura 5.5.: Detalle de elementos periféricos para la conexión del SN75176B.

3. La resistencia R3 (típicamente de valor 120 ohm), es la que adapta la impedancia en el extremo de la red a la impedancia característica del cable que se trate. Esta resistencia solo se coloca para las unidades que están como nodo terminal en cada uno de los dos extremos del bus.

Handshake

Debido a la necesidad de arbitrar el bus de comunicaciones por parte del dispositivo que en determinado momento cumpla el papel de Emisor, el control de flujo se ejercerá a través de una línea TxControl que proviene del microcontrolador. La línea mencionada estará activa, - poniendo en baja impedancia la salida hacia el canal de multidifusión de comunicaciones - durante el tiempo de escritura del mensaje emitido. Al término de la salida del bit de stop del último byte de un segmento, esta línea deberá pasar a estado pasivo a escucha de recepción.

En los microcontroladores se utilizará para ello un Pin de salida para comando de las líneas ENABLE de los Chips adaptadores (por ejemplo SN65176b), que ofician de interface con el canal Multi-Drop.

Por ello, una configuración que podría considerarse *natural* de cada dispositivo conectado a la red RS485 vía un SN75176B y que incluye un *pin* de control (**TxControl**) que debe otorgar el microcontrolador, para determinar el acceso al canal en el momento en que transmite haría que, durante el tiempo en el cual un dispositivo transmite, se debe poner activa (en alto) la línea TxControl, a efectos de tener acceso al canal. El resto del tiempo, el dispositivo está en estado de recepción, con la línea TxControl desactivada (en bajo).

Téngase en cuenta que dado que solo uno de los drivers (de transmisión o de recepción) del SN75176B está activo a la vez, la línea TxControl puede ser conectada a ambas puertas de control de los mencionados drivers, permitiendo con el cambio binario, habilitar uno de ellos cuando simultáneamente se deshabilita el otro.

La Figura 5.6 muestra esquemáticamente la red física de microcontroladores, desde el punto de vista de un microcontrolador con un puerto serie asincrónico integrado, utilizando un SN75176B como dispositivo de conversión de señales. En la figura, solamente se muestra lo que corresponde a las señales de habilitación y datos (no aparecen las líneas de alimentación).

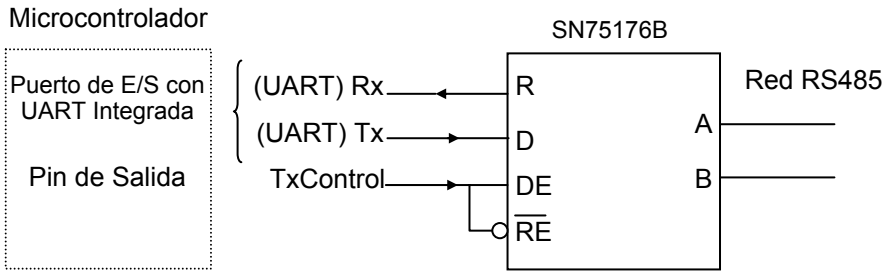


Figura 5.6: Microcontrolador asociado con el SN75176B.

Para reforzar la funcionalidad expuesta en forma gráfica, en la Figura 5.7, se muestra una secuencia de transmisión de 3 caracteres iguales (5Bh) consecutivos (por parte de un mismo emisor).

Se muestra en trazo más grueso, al principio, la transmisión del primer carácter y se muestra en el mismo trazo, al final, la transmisión del último carácter. La comunicación asincrónica está definida con un *Start Bit*, ocho bits de dato (carácter) y un *Stop Bit*.

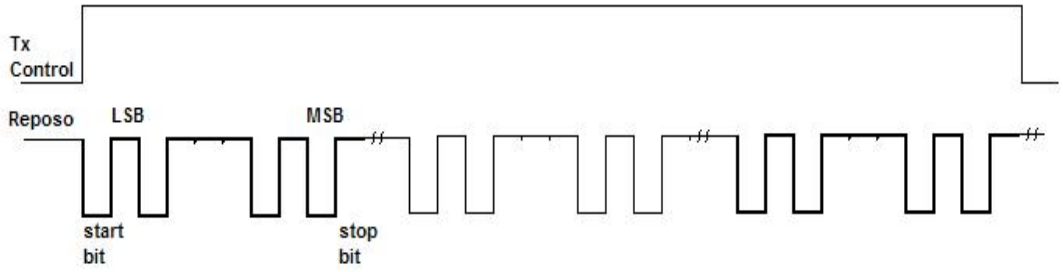


Figura 5.7: Transmisión de Tres Caracteres y Señal TxControl.

En la parte superior de la figura, se destaca como la línea TxControl permanece activa (en alto), desde el momento de comienzo del *Start Bit* del primer carácter a transmitir, hasta el momento de finalización del *Stop Bit* del último carácter, en la comunicación asincrónica de varios caracteres consecutivos.

Hasta este punto, se tiene una forma general de interconexión con red RS-485, a partir de un dispositivo que se puede utilizar de manera sencilla desde un microcontrolador con una UART integrada. De hecho, se podría utilizar desde cualquier microcontrolador con tres pines de E/S: uno de entrada y dos de salida, aunque se deberían manejar de manera precisa los tiempos de comunicaciones, tarea que se simplifica teniendo una UART integrada.

A modo de resumen, la interfase de la red física de microcontroladores que se propone en la Figura 5.6 tiene tres *pines* y un *comportamiento* bien definido, tal como se muestra en la Tabla 5.1.

TxControl	Operación	Rx	Tx
Alto (1)	Escritura	No usado	Datos de Salida
Bajo (0)	Lectura	Datos de Entrada	No usado

Tabla 5.1.: Resumen de Funcionamiento de la Red de Interconexión de Microcontroladores.

Tanto las operaciones que se muestran en dicha tabla, como el uso de los pines, se dan desde la perspectiva del microcontrolador, no de la red de interconexión propiamente dicha. La transmisión representada en la Figura 5.7 sigue estos lineamientos, es decir que tiene en cuenta la relación de la línea TxControl con la comunicación de datos a través del SN75176B. Más específicamente, aquella figura muestra las comunicaciones desde el microcontrolador que envía los datos, usando las líneas TxControl y Tx tal como se muestra en la Tabla.

Utilizando la Red de Interconexión desde un Microcontrolador

Una vez que se ha definido la red de interconexión de microcontroladores tal como la que se muestra en la Figura 5.3, estos deben se programados de manera tal que se manejen las señales TxControl, Rx y Tx de la manera especificada. Si bien la mayoría de los microcontroladores tiene al menos un puerto serie incorporado (*integrado*), no necesariamente todos manejan estos puertos de la misma manera. La programación de un microcontrolador para utilizar la red física RS-485 que se propone es específica de ese microcontrolador.

En este contexto, la programación de un microcontrolador para manejar las transferencias de datos a través de la red propuesta, sería equivalente a la creación de un *driver* de comunicaciones, de un sistema operativo de propósito general como Linux, por ejemplo.

En el caso de los microcontroladores de Microchip – utilizada para la implementación de referencia en esta tesis-, las unidades de rango intermedio (mid-range family) poseen dos registros para el modo transmisión de la UART: Uno de ellos es un Registro Buffer (RB) que hace de interfase con el Bus interno y el otro, íntimamente conectado a este es un registro (RS) que se encarga de la "serialización" de los caracteres haciéndolos salir en formato serie por la línea Tx. En recepción la situación es análoga con otros dos registros.

Este esquema lo sigue también la UART 8250 de Intel y la SIA 6805 de Motorota. Las unidades más modernas poseen un buffer de mayor tamaño, por ejemplo 16 bytes en la Intel 16550.

La transmisión normalmente se produce porque la UART interrumpe por registro de transmisión vacío. O sea que esa interrupción se produce inmediatamente en la habilitación. Esto, normalmente hace que la CPU escriba en el registro RB de la UART el primer carácter a transmitir. El registro RB inmediatamente transfiere al registro RS, provocando inmediatamente (mucho antes que se complete la Tx del primer caracter) una segunda interrupción consecutiva. Por ende puede pensarse que la UART cuenta con dos recipientes RB y RS que conforman una cola de dos caracteres.

Ahora bien, como ya se dijo anteriormente, cuando se produce la finalización del último caracter del segmento emitido por el Master, (fin del Stop bit) es necesario bajar inmediatamente la línea TxControl para ponerse en escucha. Esto es necesario, porque al ya estar completo el segmento transmitido al canal, es esperable que alguno de los Host de la red produzca su propia transmisión, como respuesta a la encuesta recibida (en el orden de 100 microsegundos o menos), por ende, el bus debe estar liberado por el Master a efectos de dar lugar a la transmisión por parte de alguno de los Slaves.

Algunos microcontroladores poseen una interrupción más una bandera que se activa y que acusan el término de emisión de un caracter, por parte del registro que efectúa la serialización. Un ejemplo de esto, es la USART 8250 de Intel (o sus upgrades como la USART 16550, del mismo fabricante). En otras líneas de fabricación, por ejemplo Microchip, no existe una interrupción al término de una emisión, aunque lo que si persiste es la existencia de una bandera (TMRT), que se activa en el momento que ya no existe caracter en RS (es una bandera que indica RS vacío), al término del Stop Bit.

Nótese que en tanto exista un caracter en RB a ser transferido a RS, la bandera TMRT no se activa. Esta sólo lo hace cuando ya no existe caracter en RS y simultáneamente no existe caracter a ser transferido desde RB.

O sea que el problema, se remite a estar consciente de la emisión del último caracter y en esa situación, actuar por la interrupción o como alternativa, efectuar un polling de la activación de TMRT a efectos de desactivar en ese instante, la línea TxControl y además, deshabilitar la transmisión de la UART, dado que de no ser así, la etapa de transmisión de la UART seguiría interrumpiendo en demanda de un nuevo carácter para Tx.

Cabe mencionar aquí, que tampoco es posible desactivar la UART cuando se envía el último caracter a RB, porque hacerlo implica que en ese instante la línea Tx pase a estado de reposo, suspendiendo la transmisión del caracter que en ese momento se pudiera estar realizando. Como puede verse, es un problema crítico el de actuación y desactivación, de la UART y la línea TxControl, un instante antes que otro integrante de la red necesita comenzar su propia transmisión.

En nuestro esquema, esta misión evidentemente tiene que desprenderse desde la subcapa inferior de transporte, que es la que conoce el final de segmento de transporte, controlando una rutina software que puede pensarse estaría incluida en un *driver*, por encima de la capa 1 (física).

O sea que para conservar el rigor del modelo OSI, la capa física - en este tipo de Host – la compondrían, la UART, el chip de conversión RS-232 a RS-485 y la rutina de software que como integrante de un driver, recibe el pasaje de argumentos y lanzamiento desde la subcapa inferior de transporte.

Conformación del Driver de manejo de la UART

Como se ha dicho en párrafos anteriores, el driver mencionado debe tener en cuenta la interrupción provocada por la UART, cuando se completa la emisión del carácter en formato serie, o en su defecto el polling de una bandera que se activa cuando la acción antedicha es completada.

Ahora bien, si la técnica fuera por interrupción, este driver sería una unidad independiente de software. En cambio, si la lógica fuera por polling de la bandera, la situación es más compleja puesto que como el Microcontrolador no posee sistema operativo que pueda dedicar un hilo a ese proceso, la encuesta quedaría necesariamente embebida dentro del Loop principal del software y en este caso tendríamos un driver que, estando situado por debajo de la capa de menor nivel del software de comunicaciones, depende de una rutina que se encuentra en el loop principal, conformando un tratamiento inadecuado del problema.

Hay varias soluciones que se han evaluado, cada una con sus ventajas y desventajas.

Una solución que se estudió en el marco teórico, es la utilización de un último caracter “dummy” que genera la subcapa inferior pero que no es parte integrante del segmento. Este caracter se envía a continuación del último del segmento y al igual que cualquier otro caracter, provocará una interrupción cuando se transfiera desde el registro RB hacia el RS, dado que el primero queda vacío. Dado que este caracter es quien ocupa ahora al RS, se desprende que en ese mismo instante el último caracter del segmento habrá salido desde el registro RS hacia la red y el driver puede cancelar la transmisión, bajando la línea TxControl.

La ventaja de este método es que se conoce con mucha precisión el momento en que culmina el segmento pero existe el problema que para cuando se ejecutan las instrucciones (dentro del cuerpo de la interrupción) que inhiben la UART es posible que esta ya haya emitido el flanco de bajada del bit de Start del carácter dummy. Esto pone en sincronismo a todas las UARTs que se encuentran leyendo el canal, a la espera de un nuevo caracter y al ser este abortado, puede haber un error de framming que detecten estas UARTS, lo que las indispone al menos por el tiempo de un caracter, haciendo probable la pérdida del SOF de un mensaje real, que en ese momento estuviera siendo producido por uno de los integrantes de la red.

Otra alternativa de solución que es la que se ha seguido en el prototipo implementado, ha sido la de utilización de un timer que, con un tiempo equivalente al determinado por el largo de un caracter, se lanza en el instante que el último carácter del segmento ingresa al buffer RS para su serialización, valiéndose de la interrupción que ocurre. Al vencimiento del timer se produce la desactivación de la UART.

Esta solución parece adecuada y fue puesta en práctica con buenos resultados, puesto que la transmisión del segmento queda encuadrada en límites conocidos, existiendo un tratamiento independiente y de bajo nivel del procedimiento.

5.5. DEFINICIÓN DE COMPONENTES EN ETHERNET

Algunos fabricantes de microcontroladores proveen un chip ethernet que embebe la capa física y la subcapa de acceso al medio (MAC) bajo norma IEEE 802.3 [5]. La mecánica de comunicación de este chip con el microcontrolador es variada. En algún caso se lo hace a través de un puerto serie sincrónico (SPI) que poseen como interfaca tanto el microcontrolador como el chip ethernet.

Otros fabricantes embeben en un microcontrolador con un solo chip la interface ethernet, quedando fuera de la inclusión las inductancias de adaptación, transformadores de pulsos y resto de componentes para obtener una línea de salida balanceada como lo requiere la norma ethernet. Estos componentes son pasivos y no admiten ser integrados conjuntamente con el chip.

Figura 5.8: Conector RJ45 que incluye componentes de adaptación a la línea.

En la Figura 5.8 se tiene una vista de un conector RJ45 (denominado comercialmente MagJack) que integra dentro de una carcasa metálica a efectos de minimizar el ruido, los componentes pasivos de adaptación de las líneas desbalanceadas Tx y Rx del chip ethernet a la red balanceada. El cable y conector a utilizar también están normalizados. Un ejemplo es el cable UTP categoría 5.

En el caso del Concentrador que sirve como prototipo en esta tesis, se utilizó el chip embebido en un microcontrolador 18F97J60. En este caso se implementa una conexión de un microcontrolador con ethernet a una red. Esta situación puede ser fácilmente extrapolada, para interconectar microcontroladores entre sí, configurando la Red 1.

CAPÍTULO 6: DEFINICIÓN Y DESARROLLO DEL SOFTWARE DE LAS UCS

RESUMEN

En el presente capítulo se otorga una descripción general y luego de detalle de las aplicaciones con que debe dotarse a las UCs.

Los requerimientos que se establecieron a las aplicaciones poseen varias fuentes de definición otorgadas en los capítulos precedentes. Veremos en este capítulo que tales definiciones impactan en el diseño de los módulos correspondientes.

Al final del capítulo se presenta un diagrama de flujo de la aplicación base donde se ha embebido una aplicación de usuario tomada como ejemplo.

6.1. INTRODUCCIÓN.

Los dispositivos Microcontroladores tienen la misión de interactuar con el campo. Normalmente, la cantidad de individuos y configuración particular de cada uno se corresponderá con la complejidad del sistema a tratar. La aplicación a instalar en cada unidad podrá tener particularidades, si fuera necesario contemplar algún proceso específico de control sobre el campo, pero en general contempla las siguientes funciones:

- Adquisición de estados binarios. (Digital Input).
- Adquisición de valores analógicos. (Analog Input).
- Adquisición de Secuencia de Eventos (SOE), con precisión del orden de10^{-4} segundo.
- Acción de Control sobre campo. (Digital Output).
- Utilización del Puerto Sincrónico (SSP) como interfase al controlador Ethernet, para comunicación con el Concentrador de Comunicaciones.
- Utilización del Puerto Asincrónico (UART), con conversión a red Half Duplex RS485, para comunicación con el Concentrador de Comunicaciones.
- Aceptación de Sincronización horaria por parte del Concentrador.

6.2. DEFINICIÓN DEL SOFTWARE

Funciones de la Aplicación

Dado el listado de funciones que se exhibe en el apartado anterior, realizaremos una descripción de las definiciones que en consecuencia han debido adoptarse para cumplimentar los objetivos. La aplicación se estructura en dos áreas o conjuntos de rutinas, con diferente jerarquía:

a) **El área del Sistema:** Lo constituye una serie de rutinas que ofician como sistema operativo de la unidad. Estas rutinas son invocadas por el programa principal o por interrupciones disparadas por el hardware, de forma tal que cumplen el rol de servicio para el invocante.

b) **El área de Usuario:** Lo constituyen las rutinas que conforman la aplicación específica de usuario, objeto principal de la implementación. Estas rutinas de usuario – que en algún caso son invocadoras de las rutinas del sistema, tomándolas como servicio para determinados fines -, contienen los procedimientos particulares directamente relacionados con la acción global que tendrá el microcontrolador en el campo a controlar.

Con el desglose apuntado es más simple la interpretación del funcionamiento de una aplicación. Esta abstracción en distintas capas, permite efectuar mejoras en las rutinas del sistema o nuevas funcionalidades en el área de usuario, sin prácticamente alteración de una capa cuando se modifica otra.

Las rutinas del sistema son las que toman control de la unidad luego del arranque y predisponen la configuración de inicio, situación para el normal desempeño de la aplicación de usuario. Estas rutinas de configuración del sistema se encuentran

distribuidas secuencialmente en el programa principal y así son llamadas hasta que el sistema en su conjunto entra en la fase de operación.

En la descripción del software que se hace a continuación, se relacionará la descripción con las etiquetas de las rutinas. Las mismas, se detallan entre paréntesis y en negrita. Asimismo, esta descripción tiene correlación con el Mapa en Pseudo-código (diagrama de flujo) dispuesto al final de este capítulo. El siguiente nivel de profundidad de desarrollo, ya lo otorga directamente el archivo fuente de la rutina respectiva, donde el código es usualmente comentado para su entendimiento y ulterior mantenimiento o modificación. El código está escrito en assembler de los microcontroladores de rango medio (16F) y alto (18F) de la línea Microchip. En el primer caso se posee un Set de **35 instrucciones**. En el segundo caso se posee un Set de **75 instrucciones**. Esto implicó la escritura de dos códigos ligeramente distintos al desarrollar los prototipos, pero conservando la estructura funcional en cada uno de los módulos, en ambas versiones de software.

Aplicación de usuario

Era necesario definir una aplicación de usuario para orientar el uso del microcontrolador a un ejemplo específico. Se efectuó un desarrollo para manejo de semáforos como un ejemplo trivial.

Es normal que los semáforos en las esquinas de calles, se conformen en varias unidades contenedoras de luces (vehiculares y peatonales). Esto especialmente en aquellas esquinas donde convergen más de dos calles (caso de existencia de diagonales) y es normal que estos contenedores estén controlados por una única Unidad de Control. Ésta es la que controla los tiempos de encendido de las luces sincronizándolas entre sí y con las otras unidades de control de las esquinas aledañas, especialmente para ocasionar lo que se ha dado en denominar “onda verde”.

En ciertos casos se requiere que estas unidades sean configurables desde el exterior (al pié de la unidad o a distancia), a través de un puerto de comunicaciones y mediante protocolo. La configuración debe incluir al menos posibilidades de agregado y quita de semáforos y cambios en los periodos de encendido de luces entre otras funciones menores.

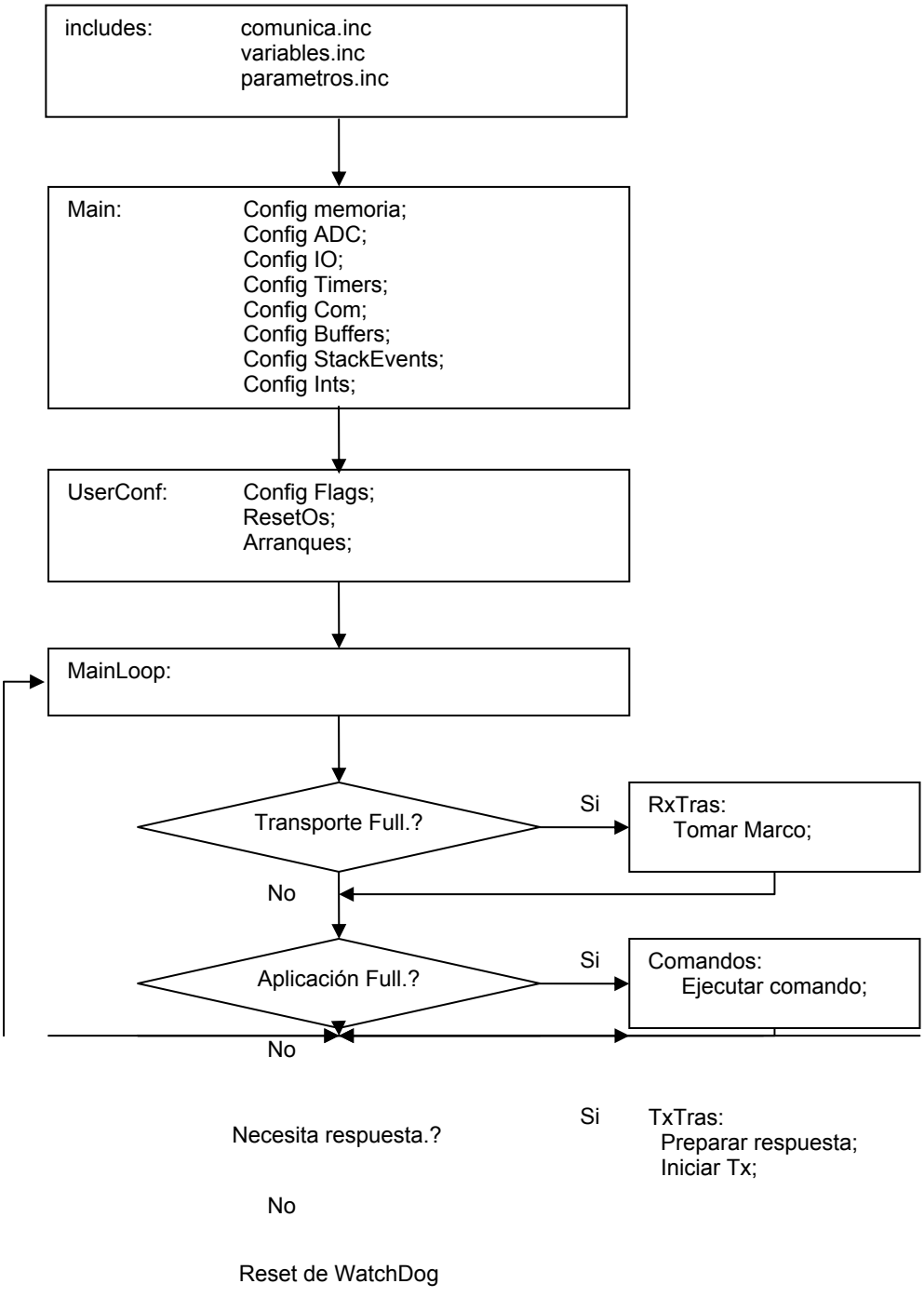
En desarrollos más sofisticados, los tiempos de encendido son determinados por alguna forma de muestreo del tránsito que fluye por cada una de las arterias, otorgando un periodo mayor a aquella dirección donde el flujo es mayor. Normalmente esta situación se detecta mediante trasductores que ingresan a la UC como una magnitud analógica.

Lo expuesto hasta aquí, también se tomará en cuenta en el desarrollo de la aplicación.

Programa principal

El programa principal, inicia con la definición de variables y constantes que se utilizan en el proceso **(include)**. En la Figura 6.1, se otorga un diagrama de flujo del mismo. Por una cuestión de organización, tales definiciones se han desplegado en distintos fuentes o “includes” a saber:

1. **Variables.inc:** Incluye nombres y direcciones de variables y de ciertos bits “flags” en algunas de ellas, a efectos de ser utilizadas en la aplicación. Estas variables pueden ser tanto del Sistema como de Usuario.
2. **Parametros.inc:** Incluye valores constantes de configuración, que utiliza la aplicación. Mayoritariamente se trata de definiciones para uso del programa de Usuario.
3. **Comunica.inc:** Incluye valores constantes para la lógica de comunicaciones en formato serie (RS232 o RS485).Flujo del programa principal



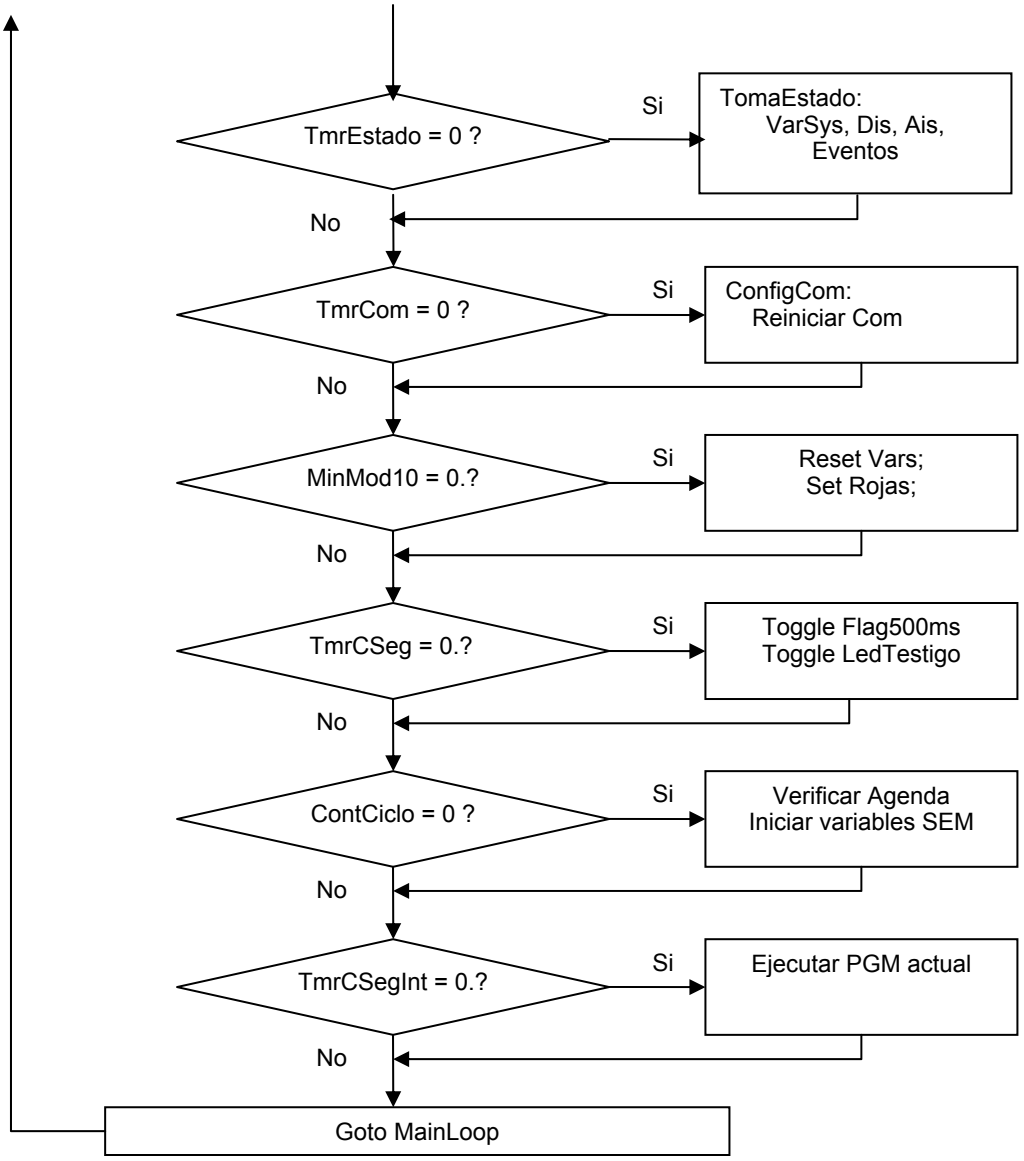


Figura 6.1: Diagrama de Flujo del Programa Principal.

A continuación se salta a la rutina principal **(main)**. En el desarrollo de programa principal se establece:

- Inicialización de memoria
- Definiciones de puertos binarios de I/O.
- Definiciones de los registros de control de dispositivos: ADC, Ports I/O, Timers, UART

- Definiciones de los “buffers” de tiempo real, capa de transporte y aplicación con sus punteros asociados, que intervienen en las comunicaciones con el Concentrador.
- Stack de eventos binarios.
- Habilitación de Interrupciones

Se establece luego la configuración de Flags iniciales de usuario **(UserConf)** y reset de los movimientos configurados en la instalación específica. A continuación se consigna como variable de importancia los arranques de la unidad **(Arranques)**. Esta variable permite seguir anomalías que provoquen el reset de la CPU sin una explícita orden de ello, por parte de un operario o el operador del CC.

Luego de la inicialización descrita, se entra en el bucle repetitivo **(MainLoop)** de la unidad. El bucle repetitivo contiene una agenda de procesos secuenciales, que el procesador efectúa en orden evitando los saltos indiscriminados a distintas zonas del código, efectuando en lo posible procedimientos estructurados – llamadas a subrutinas- y con un punto único de retorno. Esta estructura atómica de los procesos a realizar, sólo se ve alterada por las interrupciones -según se explica después-, que producen afectaciones en la entrada y salida en forma muy rápida y lógicamente, sin interferencia con la secuencia seguida en el bucle principal. Los resultados de esas interrupciones, siempre se exteriorizan como cambios en banderas, que el programa principal verifica y procesa en el orden previsto en su agenda de procesos.

Bucle repetitivo

El proceso repetitivo se inicia con la verificación de la bandera del Buffer de Transporte en Recepción **(RxBufTras)**. Si esta bandera está activa se realiza la lectura de un conjunto de bytes, que representan un **Segmento de Transporte entrante.** Este segmento se verifica en integridad y de ser pertinente su admisión, se constituye en un mensaje en la Capa de Aplicación. Tal situación se detecta por el cambio desde cero, de la variable **CBufApl** que indica el largo del mensaje a procesar.

La existencia de un mensaje implica un proceso **(Comando)** a cumplimentar por la Unidad de Control, luego, el proceso se discrimina, se ejecuta y se devuelve su respuesta (**TxTras)** de ser pertinente.

Luego se efectúa un reset del Watchdog del microprocesador, fijado en un tiempo de 576 ms de excursión máxima. A continuación se efectúa una verificación de otro timer del sistema **(TmrEstado)**, que de haber llegado a cero, indica la ejecución de la rutina de Toma de Estado de la unidad. Esta rutina efectúa un recorrido sobre todas las variables de entrada, analógicas y digitales que se hayan definido en su tabla correspondiente. Asimismo, en la misma rutina se recogen otras variables del sistema **(VarSys)**. Este proceso, tiene como finalidad efectuar la recolección de las variables definidas, dejándolas en un buffer de tiempo real **(BufAD)** que se suma a otro Buffer del sistema **(Eventos)**, para ser enviado como respuesta cuando el CC, en su “scan” repetitivo lo requiera. En estas condiciones, el Buffer de tiempo real, se refresca con el periodo fijado en un parámetro del sistema **(PeriodoToma).**

Seguidamente, se efectúa una verificación del estado de comunicaciones. Si no se recibió un mensaje desde el CC en un tiempo **(TmrCom)** prefijado y variable –típicamente 2 minutos-, las comunicaciones se reinician. A continuación se verifica si finalizó el timer que lleva el conteo de medio segundo **(TmrCSeg).** Este timer es programable y en esta aplicación se lo ha definido en 50 centisegundos, con la finalidad que exista una bandera que marca los flancos de cambio y sincroniza todos los eventos repetitivos que tienen un periodo de medio segundo (Luces intermitentes, Leds testigo, etc.).

En este punto se cumplimenta un aspecto importante para la aplicación de usuario: Se verifica si se cumple algún múltiplo de 10 minutos dentro de la hora **(MinMod10)**. Esta es una verificación que se efectúa por seguridad y tiene por objetivo, que si determinada unidad pudiere haber quedado asincrónica respecto al tiempo global, por un reset o algún otro causal, esta tenga oportunidad de sincronizar el proceso de semaforización con la hora global en un tiempo relativamente corto.

Para ello, ante el pasaje por cero de la variable anteriormente apuntada, se reinicia el contador de ciclo de la semaforización y todas sus variables asociadas. El hecho de permitir solamente ciclos cuyos valores sean divisores exactos de un tiempo de 10 minutos (600 seg.) tiene como objetivo que todos los ciclos de cualquier unidad del sistema de semaforización, comiencen en la frontera de los múltiplos de 10 minutos del tiempo global. Esto determina que ante re-arranque de cualquier unidad (por corte de luz o tareas de mantenimiento), la misma podrá estar asincrónica con el resto a lo sumo 10 minutos pues se conoce que para cualquier periodo programable, su comienzo siempre se produce en el inicio del minuto 0,10, 20, 30, 40 ó 50.

Luego de estos pasos, la agenda continúa por el análisis de los procesos de Usuario, en este caso de la semaforización.

Se verifica si el ciclo ha concluido y en función de ello, si está prefijado un cambio de programa en la semaforización. El sistema tiene previsto la utilización de 3 (tres) programas posibles: **PGM0**, Amarillo Intermitente. **PGM1 y PGM2**, programas normales, con un periodo fijado por la variable **ContCiclo.** Estos periodos pueden tomar los siguientes valores, en segundos:

1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 25, 30, 40, 50, 60, 75, 100, 120,150,200, 300, 600.

Como puede observarse, estos valores tiene como propiedad común, el hecho de ser divisores exactos de un periodo base de 600 segundos como ya se aclarara anteriormente.

Si se programa en la Agenda de Semaforización un cambio de programa a una hora determinada (con formato HH:MM (Horas.Minutos)), tal cambio ocurrirá a la hora prefijada luego que se cumple el ciclo que está en curso.

Por último, se verifica el pasaje por cero de la variable que marca el tiempo básico de la semaforización **(TmrCSegInt),** de 500 milisegundos **-** Ninguna acción de semaforización tiene lugar en un intervalo menor -, y llegado a cero, se cursan las acciones del programa en curso.

Concluida esta acción se regresa al inicio del bucle principal **(MainLoop)**.

Interrupciones

El programa de tratamiento de las interrupciones, es una acumulación de rutinas de ejecución condicional relacionada con alguna de las banderas identificatorias de un suceso. En la Figura 6.2, se otorga un diagrama de las consultas que deben efectuarse ante la ocurrencia de una interrupción y cada rombo de decisión está relacionado con los títulos en los párrafos que siguen a continuación del mencionado diagrama:

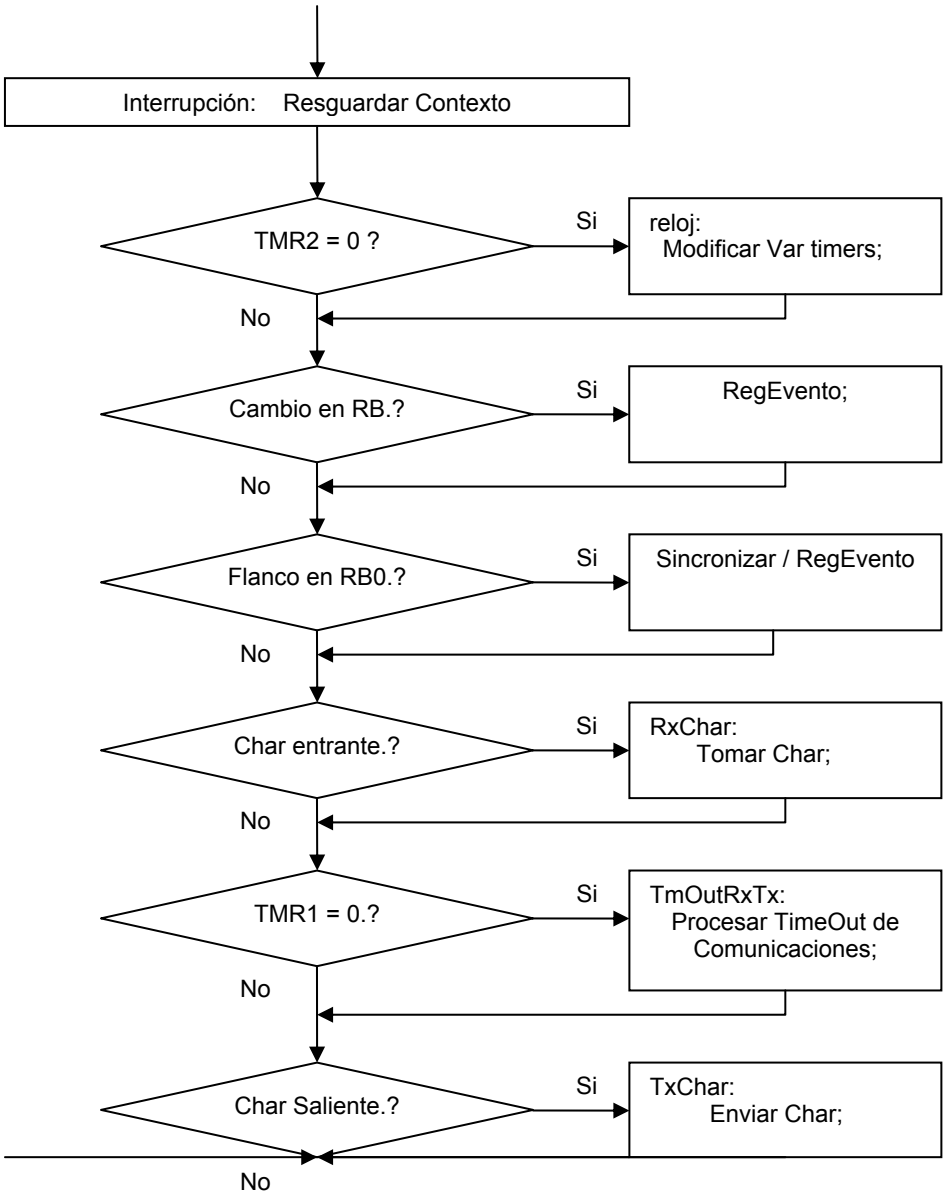


Figura 6.2: Diagrama de Flujo de las Interrupciones.

Reloj: Definida por expiración del timer TMR2, con un periodo de 1 segundo con cristal de 32768 HZ, de modo tal que en un arreglo de variables se lleva la hora real del sistema, con granularidad de 10^{-4} segundo y alcance hasta 64 años (años 2000 a 2063). Este valor de la hora es sincronizado periódicamente con una función de protocolo, que le permite llevar la sincronización interna entre unidades y además servirá para imponer el Time-Stamp, ante la ocurrencia de un evento.

RegEvento: Se define el evento como el cambio de estado de alguna de las entradas digitales definidas como tal. Ciertos pins del MicroC, se establecen como DIs con capacidad de **interrumpir** por cambio de estado. Otros pins se muestrean periódicamente **(TomaEstado)** y se detecta un cambio por comparación con su estado anterior.

Independientemente de su técnica para detectarlo, un cambio de estado se consigna como un evento. Ante ocurrencia de un evento se ejecuta la rutina RegEvento, donde se registra el tipo de evento, el puerto, el bit y su estado actual conjuntamente con el Time-Stamp de ocurrencia, extraído desde el Reloj del sistema en el momento de llamada a la rutina. Esta estampa de tiempo tendrá el formato **YY:DD:HH:SS.mmmm**. Esto es, se almacenará el año, el día del año, hora, segundo de la hora y fracción de segundo con 4 dígitos significativos del momento de ocurrencia del evento.

Cada uno de los eventos que ocurran se registra en un banco de memoria de la UC. La capacidad de ese buffer es de 12 eventos en los microcontroladores de rango medio y 32 en los de rango alto (podría ser superior pero no se lo consideró necesario), que quedan a la espera de la consulta por parte del CC para ser evacuados. El Buffer tiene formato de cola cíclica, por lo que completada la cola, de no existir la evacuación de eventos, no se registrarán nuevos.

Evento por flanco: La línea Rbo del microprocesador posee una bandera específica de detección para un único flanco de la señal entrante (flanco ascendente o descendente). Esta propiedad puede ser utilizada especialmente para sincronismo o para detección de algún evento singular. En el caso que nos ocupa, la línea en cuestión puede utilizarse para registrar el flanco ascendente de una señal centralizada de sincronismo, que permitiría ante ausencia de un procesador dedicado a la sincronización de un área – rol que en el caso normal cumple el Concentrador-, sincronizar con una fuente global externa para todas las unidades.

RxChar: Ingreso de Char en la UART. Cuando ingresa un nuevo carácter en la UART se provoca una interrupción que llama a la rutina **RxChar**. Dentro de esta rutina está escrito el código de la **Subcapa Inferior de Transporte** del protocolo de comunicaciones. De acuerdo a dicho protocolo, si se recibe el carácter **SOF** (start of frame), se da inicio a la recepción secuencial de caracteres, introduciendo cada caracter entrante en el Buffer de Transporte **BufTras**, estableciéndose el timer **TMR1,** -como tiempo máximo de separación entre caracteres (Time-out) -, con un intervalo equivalente al tiempo de transmisión de 1 ó 2 caracteres (variable).

TmOutRxTx: Agotamiento del TMR1**.** De ocurrir el Time-out en recepción aludido en el párrafo anterior, por no recibir un nuevo carácter en el intervalo mencionado, el Buffer de recepción en Transporte se re-inicializa, quedando a la espera de un nuevo inicio de trama (nuevo SOF).

En condiciones normales, se reciben todos los caracteres determinados por el largo del mensaje (**Qty**), con lo cual se arriba a la finalización del mensaje entrante, activando la bandera **RxTrasFull**, que recogerá luego el programa principal.

TxChar: Egreso de Char de la UART**.** Cuando el registro de transmisión de a UART se encuentra libre para admitir un nuevo carácter a transmitir, se emite un interrupción que llama a la rutina **TxChar** que coloca un nuevo carácter proveniente del Buffer de Transporte en Transmisión. Cuando se llega al tope de caracteres a transmitir (Qty), se inhibe la transmisión de la UART.

Rutinas del Sistema y de Usuario

Estas rutinas son de servicio tanto para el sistema como para la aplicación de usuario. Son bloques o procedimientos de código compacto, adecuados para ser invocados, que devuelven alguna acción concreta y permiten su reutilización en distintos puntos del programa. En el Apéndice A se otorga el esquema jerárquico de dependencia de estas rutinas, relacionándolas en un una estructura de llamante-llamado, como otro enfoque complementario de interpretación.

CAPÍTULO 7: DESARROLLO / ADECUACIÓN DEL SOFTWARE DEL CONCENTRADOR DE COMUNICACIONES

RESUMEN

En este capítulo se describe el funcionamiento de un dispositivo esencial de la red: El Concentrador. Éste sirve de Puente o Gateway, entre la Red 1y la Red 2.

La concentración de UCs mediante un enlace multipunto, determina que deba utilizarse un protocolo maestro – esclavo para arbitrio de las comunicaciones en el bus. Esta situación, permite una interesante propiedad del bus controlado, que puede aprovecharse para la sincronización interna de las Unidades de Control que se encuentran en la Red 1 debajo de cada concentrador, mediante una técnica de Broadcast Embebido. Esta acción de sincronización que ejerce cada concentrador sobre su red dependiente, permite una elevada precisión de los relojes individuales de cada UC.

Además de las dos funciones, la inclusión del Concentrador se encuentra justificada por la necesidad de disponer de un hardware, que establezca una relación peer-to-peer con las UCs integrantes de una subred

En otra instalación de Red, tal Concentrador podría ser un simple Hub o Switch Ethernet

7.1. INTRODUCCIÓN.

Aplicación del Concentrador de Comunicaciones

La elección del nombre: **Concentrador,** está motivada porque es un dispositivo que funciona en varias capas. Es de capa física, ya que sólo transmite bits sin analizar. Un **Bridge** en cambio, es un dispositivo de Capa 2 (enlace) que transmite segmentos, analizando las direcciones de emisión y destino de esos segmentos, para encauzarlos a la puerta de enlace correspondiente.

Desde el punto de vista apuntado, nuestro Concentrador cumple el mismo papel que cumple un Hub en ethernet, que es un dispositivo de capa Física que solo replica Bits a todos los segmentos. Estas conclusiones están de acuerdo con la bibliografía de referencia [2] [3] [12]. Pero en nuestro caso, el Concentrador es también el responsable de mantener el sincronismo de la Red 1, lo que lo convierte en un dispositivo **híbrido,** que podría ubicárselo como un elemento de capa de transporte o superior.

El Concentrador es un elemento que ha sido diseñado inicialmente para operar entre dos redes, una red Ethernet y otra red serie RS485. Además tiene capacidad de comunicarse mediante otro canal serie bajo norma RS232. En el contexto de esta tesis, el Concentrador está ubicado entre la **Red 1** y la **Red 2** y gestiona las comunicaciones entre el Centro de Control (CC) y las Unidades de Control (UCs). Por otro lado, el canal serie alternativo ha sido utilizado para ingresar información desde un GPS a efectos de sincronización externa. El hardware del Concentrador está basado en el microcontrolador PIC 18F97J60 que incluye un módulo Ethernet en firmware.

Funciones del Concentrador

Las dos funciones principales del Concentrador son:

a) Actuar de puente o gateway, entre la **Red 1** bajo norma RS485 que une a las UCs entre sí, y la **Red 2** bajo norma Ethernet, que vincula los concentradores entre sí y al CC.

b) Proveer sincronismo a las unidades de control vinculadas a la Red RS485.

Existen otras funciones adicionales complementarias como son las de llevar un reloj de tiempo real de buena precisión, con cristal de 32768 Hz que le otorga una granularidad de 1/32768 s. Además el Concentrador efectúa la lectura de un GPS asociado a su puerta serie secundaria. Estas dos funciones son muy importantes a efectos de efectuar la **sincronización externa** de la red, a la que nos referiremos en el Capítulo 9.

Para cumplir la funcionalidad apuntada en a), el Concentrador efectúa un programa cíclico en el que permanentemente verifica la existencia de caracteres en alguna de las dos colas sobre las que efectúa el puente: La cola asociada a la conexión ethernet y la cola asociada a la conexión RS-485.

En ocasión de una consulta, el CC elabora el paquete completo del protocolo definido para la red RS-485. Este paquete se elabora desde el SOF hasta los bytes que ofician de BCC. Este arreglo se encapsula luego en TCP/IP y es enviado al Concentrador que posee una dirección IP fija. Cada concentrador de la red

mantiene abierto un Server Socket y el CC abre tantas conexiones cliente **persistentes,** como concentradores existan en la red.

Cuando el CC envía un paquete a un concentrador determinado, el mismo tiene prevista la lectura de su Server Socket y la acumulación de los caracteres recibidos en la capa de aplicación. La lectura se efectúa hasta el largo efectivo (lo hace leyendo Qty) del paquete en RS-485. Lo importante de esta acumulación de caracteres en un modo Store and Forward, es que el segmento de transporte que se enviará luego a la red RS-485 no sufra interrupciones en su envío, en un modo tal que pudiere disparar el timeout de recepción de la UC destino.

En sentido inverso sobre la Red 1, cuando el Concentrador recibe un paquete desde una UC, lo acumula en la cola respectiva hasta que este se completa (se observa Qty) y logrado esto, se lo encapsula en TCP/IP y se lo envía con dirección al CC con quien está establecida la respectiva conexión persistente.

Como puede observarse el tratamiento sobre el lado TCP/IP desde el punto de vista de la **fragmentación** de paquetes, queda librado al funcionamiento que establece el fabricante para el Stack TCP, sin mayores previsiones que las que se establecen por defecto. Esta situación no es la misma sobre el lado de la red multipunto, pues allí es crucial el envío “atómico” del paquete, situación que se refuerza teniendo la necesidad que cada paquete tenga un tiempo de envío constante y predecible en la emisión de la PeH.

Para cumplir la funcionalidad apuntada en b), el Concentrador periódicamente (típicamente cada 5 segundos) debe emitir un paquete de PeH. Ese paquete es emitido en broadcast y para asegurar éxito de llegada a todas las UCs de la subred, el canal RS-485 debe silenciarse. Para ello, el Concentrador asume un **tiempo de silenciamiento** del canal que en las pruebas se lo fijó en **50 ms.** Al inicio de ese periodo se lanza un timer para su medición, suspendiendo simultáneamente la lectura del socket TCP. El hecho que se suspenda la lectura del socket, hace que no se emitan consultas a la red multipunto y que en consecuencia, no haya respuestas quedando silenciado el canal de la Red 1, aunque sí se reciban en los buffers de la red ethernet los paquetes que pueda estar emitiendo el CC.

Ahora bien, al suspender con este mecanismo las consultas a la red multipunto, se debe garantizar que la eventual consulta que estuviere en tránsito, se resuelva en un tiempo menor a 50 ms. El tiempo máximo aproximado se obtiene como las suma de los tiempos de transmisión de los bytes de la consulta y su respuesta respectiva en el caso peor. Por ejemplo en 9600 bps, para un total de 250 bytes como peor condición, el periodo de silenciamiento debiera ser de alrededor de 250 ms. Como puede observarse, existe una relación de compromiso para el tiempo asignado a la PeH, que le quita eficiencia al canal por encima de la ya comprometida situación que se tiene con un canal Half-Duplex y de baja velocidad. Se debe entonces tener en cuenta la velocidad, la frecuencia con que se emite la PeH y el largo máximo de los paquetes esperables en la red, sumando la consulta y su respectiva respuesta para evaluar la eficiencia.

Emitida la PeH, luego del periodo de silenciamiento, el Concentrador reanuda la lectura del socket TCP hasta que llegue el momento de hacer una nueva emisión de PeH, repitiendo el mecanismo. En lo que sigue se otorga una descripción de detalle de la aplicación del Concentrador.

Software del Concentrador

El código fuente del Concentrador es un programa con una etapa preliminar de inicialización de variables y dispositivos y un bucle principal infinito, característica de los sistemas dedicados. Para las comunicaciones sobre Ethernet se utiliza la pila TCP/IP libre y de código abierto desarrollada por Microchip. Esta pila brinda una API de alto nivel que cubre las funciones básicas de comunicación, como la creación y uso de sockets.

Dado que el Concentrador tiene que cumplir con más de una función principal, en particular con las dos funciones mencionadas, y que para cumplir con estas funciones debe atender a varias tareas por lo general independientes entre sí, es evidente que se requiere de una estructura adecuada para el software. Tal estructura podría ser la que brinda un Sistema Operativo, por lo tanto una opción es adecuar un Sistema Operativo de código abierto desarrollado para microcontroladores a las necesidades del Concentrador. Otra opción es desarrollar desde cero el Sistema Operativo. El Concentrador utiliza una tercera opción que es la de hacer uso de la técnica de programación denominada multitareas colaborativas.

Por lo tanto, la estructura del Concentrador es la de un gran programa con un único hilo de ejecución, que invoca secuencialmente a diferentes tareas (colaborativas) en un ciclo infinito. Las tareas colaboran con el fin común resolviendo su trabajo en un tiempo apropiado (no demasiado largo), o resolviendo parte de su trabajo y entregando el control al invocador. De esta forma se asegura que todas las tareas tendrán oportunidad de ejecutarse.

En lo que sigue se otorga una descripción de detalle del software del concentrador, introduciendo en recuadros aspectos más finos de la descripción y que pueden pasarse por alto a efectos de no perder el hilo del desarrollo general.

El Concentrador además utiliza interrupciones en dos niveles de prioridad, como se verá más adelante. El programa del Concentrador está compuesto por varios módulos y varios archivos de cabecera. Los principales módulos y archivos de cabecera son los siguientes:

AppLayer.c: Realiza tareas de alto nivel como sincronizar la red RS485 y setear el reloj de tiempo real (RTC – Real Time Clock) con la fecha y hora ya leídos del GPS.

AppLayer.h: Definición de constantes y publicación de funciones de AppLayer.c.

Concentrador.h: Definición de constantes que deben ser conocidas globalmente.

HardwareProfile.h: Definición de constantes que definen el hardware utilizado.

Main.c: Módulo principal. Contiene la función main() que es donde comienza la ejecución del microcontrolador. Esta función puede considerarse el programa principal, dado que en primer lugar inicializa los dispositivos y las variables a utilizar, y luego entra en un ciclo repetitivo infinito invocando a las diferentes tareas colaborativas.

RTClock.c: Contiene el código que implementa un reloj de tiempo real manejado por interrupciones y con posibilidad de sincronización desde un GPS.

RTClock.h: Definición de constantes y tipos de datos necesarios para el RTC. Además, publicación de funciones para ser utilizadas desde otros módulos.

UART2TCPBridge.c: Contiene el código que transfiere los datos recibidos desde la red Ethernet a la red RS485 y viceversa. Estos datos vienen en paquetes correspondientes a un protocolo en capas utilizado tanto por el CC como por las UCs. Por esto el Concentrador debe ser consciente de este protocolo, si bien no requiere conocer todos los detalles del mismo.

UART2TCPBridge.h: Publicación de funciones para ser invocadas desde otros módulos.

Existen además varios módulos y archivos de cabecera pertenecientes a la pila TCP/IP utilizada, de los cuales se puede encontrar documentación en el sitio oficial de Microchip.

Programa principal

El programa principal es la función main() contenida en el módulo Main.c. Un diagrama de flujo de la estructura del programa puede apreciarse en la Figura 7.1.

Entrando en el módulo main.c se observa la inclusión de los archivos .h de declaración de funciones que se utilizan en los distintos módulos del programa. La función main() en su comienzo, invoca a varias funciones de inicialización, de las cuales se otorga un detalle en el cuadro que sigue.

InitialiceBoard() efectúa la inicialización de distintos periféricos de la placa (Leds, pulsadores y otros).

LCDInit() yLCDUpdate() hace lo propio fijando las condiciones iniciales del display LCD que permitirá visualizar la hora real de la placa.

TickInit() y RTC TickInit() inicializa el timer utilizado en el RTC.

StackInit() fija condiciones de inicio para el Stack TCP/IP incluido en las librerías del fabricante de la placa.

UART2TCPBridgeInit() inicializa la UART utilizada en el puente entre la Red1 (RS-485) y la Red 2 (Ethernet).

SyncGPSInit() inicializa la UART utilizada para la comunicación con el GPS.

GPSInit() inicializa la línea de interrupción utilizada para la recepción del pulso PPS (Pulse Per Second) que emite un GPS que contenga esa prestación.

stTime() inicializa la fecha y hora a un valor arbitrario, miércoles 31 de diciembre de 2008, a las 23:59:15, es decir, 45 segundos antes del comienzo del año 2009. Esta por supuesto no es la fecha y hora actual. Para lograr que el Concentrador tenga la fecha y hora actual se debe conectar un receptor GPS en el enlace serie alternativo. En caso de no existencia de este, es posible enviar un comando de PeH con destino al Concentrador desde el CC y a partir de allí, aquel es el patrón de hora de su subred. Para lograr la máxima precisión posible para Sincronización Externa, sería necesario además conectar la línea PPS del receptor GPS al pin correspondiente en la placa del Concentrador.

Particularmente, para el correcto funcionamiento del Concentrador, no es necesaria una inicialización explícita del RTC. En este caso, al iniciarse, este toma como fecha y hora el 1 de enero de 1970 a las 00:00:00. Ocurre que esta no es una fecha válida en el protocolo de aplicación definido, dado que el rango de fechas soportado por este protocolo (año 2000 a 2063), es inferior al rango

de fechas disponible en el Concentrador. Es por esto que se hace necesaria la invocación a setTime(timeTest) a modo de inicialización.

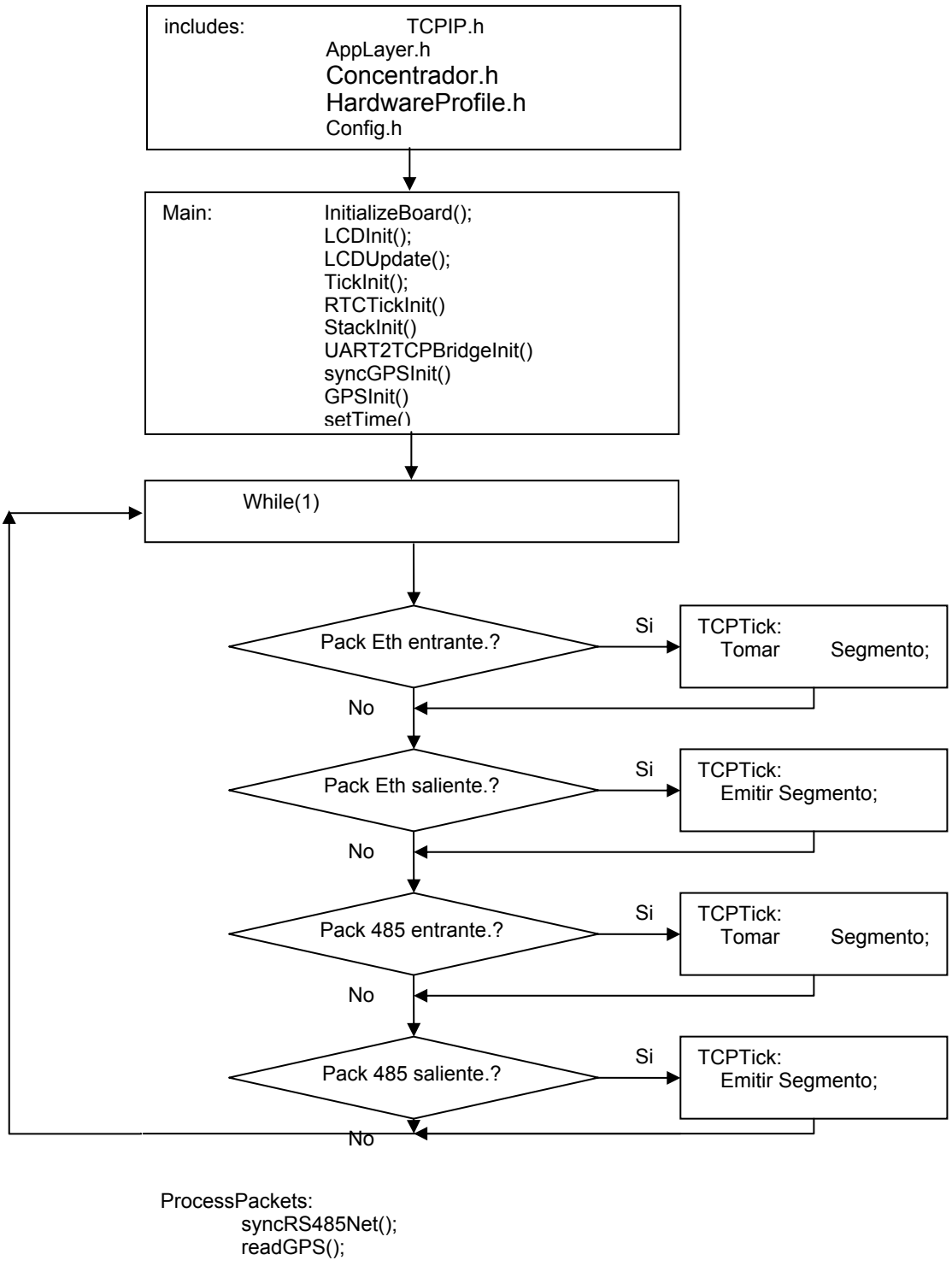


Figura 7.1: Diagrama de flujo del Software del Concentrador.

Luego de inicializar la fecha y hora para el Concentrador, se inicializan variables que indican cambio de segundo (Timers2zero.T0 y Timers2one.T1) y un contador regresivo de segundos downCount[DC_0] que se carga con el valor PeH_INTERVAL definido en AppLayer.h. El mismo indica el intervalo de tiempo entre dos paquetes de Puesta en Hora enviados a la red RS485. La implementación de estos marcadores y contadores de segundos se verá más adelante cuando se analice el módulo RTClock.c.

Ciclo repetitivo

Una vez finalizadas todas las inicializaciones se ingresa en el ciclo repetitivo infinito del programa principal. Este ciclo es el while(1) cuyo fin coincide con el final de la función main().

Al principio del while se hace un toggle del led 0 de la placa del Concentrador (el de más a la derecha) cada 2 segundos. A continuación se muestra la fecha y hora en el LCD cada un segundo. Se incluye el día de la semana correspondiente a la fecha, en el final de la segunda línea del LCD, justo debajo de la unidad del segundo. El día de la semana se indica con un número entre 1 y 7, donde 1 corresponde al lunes y 7 al domingo.

Luego se muestra una variable de debug en la línea inferior del LCD antes del día de la semana. Esta variable indica cuántos ticks del timer del RTC pasaron en la última vuelta del ciclo completo del programa. Un tick es una unidad en el registro contador del timer. Esto puede traducirse en una medición de tiempo sabiendo cómo está configurado el timer. Nótese que este valor no se muestra en cada iteración del while(1) sino solo cada un segundo.

Más adelante hay una invocación a StackTask(). Esta invocación es esencial para que funcione la pila TCP/IP. En esta función se comprueba si existe un paquete entrante y de ser así se invoca la entidad apropiada para procesarlo. En modo saliente se opera en modo similar, emitiendo un segmento saliente si el mismo se encuentra en el buffer de aplicación para ser procesado. Es de destacar que esta visión del diagrama de flujo se sitúa en una abstracción al tope de la capa de transporte, dado que las capas inferiores están embebidas dentro de la librería del Stack TCP/IP provisto por el fabricante.

A continuación se invoca a UART2TCPBridgeTask() que procesa de un modo similar, cualquier tarea pendiente relacionada con la función de recibir o emitir paquetes desde y hacia la red RS-485, completando la función puente del Concentrador.

Por último, se encuentra una invocación a processPackets() que realiza dos tareas independientes entre sí. La primera es gestionar el servicio de sincronismo para la red RS485 mediante la invocación a syncRS485Net(). La segunda es hacer efectiva la última fecha y hora válida recibida del receptor GPS en el reloj local (RTC) invocando a readGPS() cuando este está conectado. Si el GPS no existe la hora es la que lleva el RTC del Concentrador.

Funciones syncRS485Net() y readGPS()

Habiendo descripto la función main() y el módulo Main.c, pasemos a ver los detalles de syncRS485Net() en el módulo AppLayer.c. En esta función puede notarse la estructura que tienen todas las tareas colaborativas que componen el programa.

Se utiliza una variable de estado (que puede tomar diferentes estados), en este caso SyncState y se determina la acción a realizar según el estado actual en un bloque switch. A este algoritmo y otros equivalentes que se encuentran en muchas funciones del programa también se le puede denominar máquina de estados finitos (FSM – Finite State Machine), dado que cumple con el concepto de una máquina de estados finitos básica. En particular, la lógica de esta FSM realiza lo siguiente:

En primer lugar espera PeH_INTERVAL segundos. El cambio de estado se produce en la frontera del cambio real de segundo en el RTC. Esto se debe a cómo está implementado downCount[DC_0], como se verá más adelante.

El segundo estado es otra espera, en este caso de 45 centésimas de segundo. Justo antes de cambiar al próximo estado se detiene la recepción de datos desde la red Ethernet mediante la invocación a stopBridging().

En el siguiente estado se esperan 5 centésimas de segundo y se construye un paquete de Puesta en Hora (PeH), el que además se comienza a enviar a la red RS485.

En el último estado se van enviando, a medida que la UART está disponible, todos los bytes que componen el paquete PeH. Cuando se envía el último byte se produce la transición al estado inicial y todo el ciclo se vuelve a repetir indefinidamente.

Antes de volver al estado inicial, se restituye la recepción desde la red Ethernet mediante la invocación a resumeBridging().

En el mismo módulo, la función readGPS() es otra máquina de estados finitos, que se encarga de ajustar la fecha y hora del RTC hasta nivel de segundo con la última información leída del receptor GPS. Este ajuste solo se lleva a cabo si ya existe en memoria una copia de la información leída del GPS. La disponibilidad de esta información se comprueba cada vez que transcurre el 40 por ciento de cada segundo del RTC.

Puente entre la red Ethernet y la red RS485

Como se describió en la sección anterior, la función main() invoca a UART2TCPBridgeInit() en su etapa de inicialización. Esta función se encuentra en el módulo UART2TCPBridge.c e inicializa una de las dos UARTs con que cuenta el microcontrolador PIC18F97J60 para ser utilizada en la función de puente del Concentrador. Las variables y constantes que hacen referencia a esta UART terminan con ‘x’, mientras que la UART utilizada para la conexión con el receptor GPS se denomina UART ‘y’. Como parte de la inicialización se determina que el nivel de interrupción que generará esta UART será bajo (low priority).

Además se inicializa el estado de la línea TxControl en alto para preparar las comunicaciones en la red RS485. Esta condición determina que solo el

Concentrador inicia las comunicaciones en el canal, ya que este se encuentra siempre en modo Tx, determinado por su línea TxControl. Esta forma de trabajo con el Concentrador monopolizando el canal si no se espera escuchar, es conveniente ya que pone el bus en un modo de baja impedancia (determinada por el driver de salida del Concentrador) lo que lo hace más altamente inmune al ruido.

Al principio del módulo UART2TCPBridge.c pueden verse algunas constantes útiles para el puente y debajo de estas las variables más importantes utilizadas en este módulo, los buffers de envío y recepción y los punteros a la cabeza y a la cola de cada buffer. Estas variables son las siguientes:

vUARTRXFIFO[MAX_PACKET_SIZE + 8] Buffer de recepción desde la red RS485 (a través de la UART), a su vez este es el buffer de envío hacia la red Ethernet.

vUARTTXFIFO[MAX_PACKET_SIZE + 8] Buffer de envío hacia la red RS485 (a través de la UART), a su vez este es el buffer de recepción desde la red Ethernet.

RXHeadPtr Puntero a la cabeza del buffer vUARTRXFIFO.

RXTailPtr Puntero a la cola del buffer vUARTRXFIFO.

TXHeadPtr Puntero a la cabeza del buffer vUARTTXFIFO.

TXTailPtr Puntero a la cola del buffer vUARTTXFIFO.

Estos buffers se utilizan como colas circulares en modo FIFO (el primer dato que entra es el primero que sale).

La función Timer2Start() inicializa y da comienzo al timer 2 para que genere una interrupción en el tiempo indicado por el parámetro period que recibe. Este timer se utiliza para bajar la línea TxContol después de un tiempo predeterminado posterior al envío del último byte a la red RS485, a fin de permitir que la unidad de control que corresponda pueda enviar un paquete de respuesta.

La función UART2TCPBridgeTask() controla si existen datos disponibles para enviar por TCP/IP y si hay datos recibidos desde el mismo vínculo TCP/IP. Para determinar la disponibilidad de datos que esperan ser enviados hacia la red Ethernet se controlan los punteros del buffer vUARTRXFIFO y también se controla el espacio disponible en el buffer de la pila TCP/IP por medio de TCPIsPutReady(). De esta manera se envía todo lo que sea posible enviar y se reajustan los punteros. Lo que no se pueda enviar se deja para la próxima iteración del programa principal.

Para comprobar si hay datos recibidos por TCP/IP se utiliza la función TCPIsGetReady(). Es importante notar que esta comprobación solo se realiza si la variable noBridging está en 0. Esta variable se pondrá en 1 un tiempo antes del envío de un paquete de sincronización (PeH) a la red RS485, y se restituirá al valor normal (cero) luego de ese envío. Si se encuentran datos recibidos por TCP/IP, estos deben ser transferidos al buffer vUARTTXFIFO. También se utilizan en este caso los punteros del buffer para prevenir un rebasamiento del mismo.

La función UART2TCPBridgeISR() es la rutina de atención para las interrupciones de la UART x. En primer lugar la función comprueba si la interrupción fue por recepción, y si fue así recibe el byte y lo deposita en el buffer vUARTRXFIFO.

Luego se comprueba si la interrupción fue por envío y en este caso toma el byte correspondiente del buffer vUARTTXFIFO y lo envía hacia la red RS485.

Interrupciones

En el módulo main.c, antes de la definición de main(), pueden verse las dos funciones que se ejecutan ante una interrupción. La primera es LowISR() que se ejecuta cuando se produce una interrupción de bajo nivel. La segunda es HighISR() que se ejecuta cuando se produce una interrupción de alto nivel. Debe recordarse que cada nivel de prioridad posee sus respectivos vectores de interrupción en las direcciones 18h y 08h de la memoria de programa, respectivamente. El diagrama de flujo de ambos niveles de interrupción se presenta en las Figuras 7.2. y 7.3.

En el PIC18F97J60 las fuentes de interrupción son múltiples, prácticamente todos los dispositivos incluidos en el microcontrolador tienen capacidad de generar una interrupción (timers, UARTs, ADC, etc.). Es por esto que una vez producida la interrupción, cuando el hilo de ejecución se bifurca a una de las dos funciones apuntadas, debe buscarse la fuente concreta que interrumpió.

Por otro lado, cuando se configura un dispositivo, normalmente se hace una elección acerca del nivel de interrupción que utilizará, se asigna este nivel durante la inicialización del dispositivo, y se mantiene este nivel (por lo general) mientras el programa esté en ejecución.

Bajo nivel de prioridad

En el Concentrador el Timer0 que es la base de tiempo para distintos timers del sistema genera interrupciones de bajo nivel. Lo mismo ocurre con la UARTx utilizada para la conexión con la red RS485. El Timer2 utilizado también en la función de bridge y la UARTy que está conectada al receptor GPS.

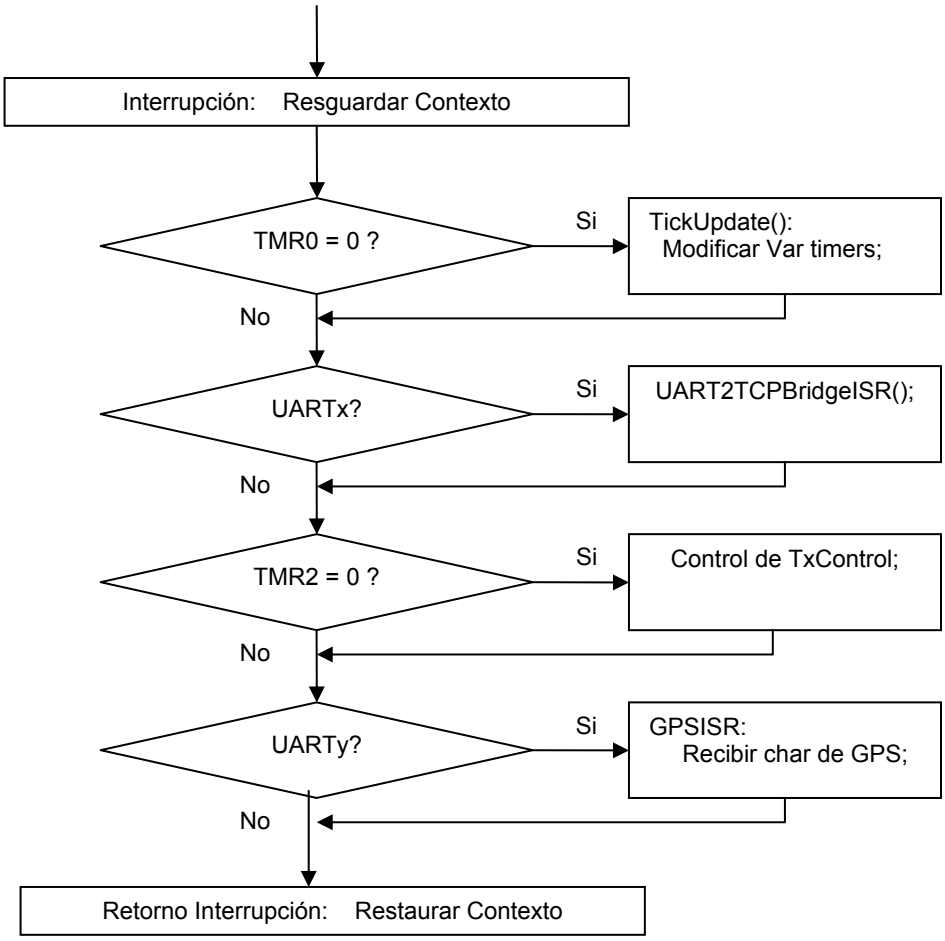


Figura 7.2: Diagrama de flujo de las interrupciones de bajo nivel de prioridad.

Alto nivel de prioridad

Existen solo dos posibles interrupciones de alto nivel. La primera es el pulso PPS (pulse per second), proveniente del receptor GPS y la segunda es el Timer1 utilizado para implementar el reloj de tiempo real.

En las dos funciones de atención de interrupciones (LowISR() y HighISR()) pueden verse las invocaciones a funciones que atienden cada uno de los casos particulares mencionados. Internamente cada función comprobará como primera medida si realmente se produjo la interrupción que le corresponde.

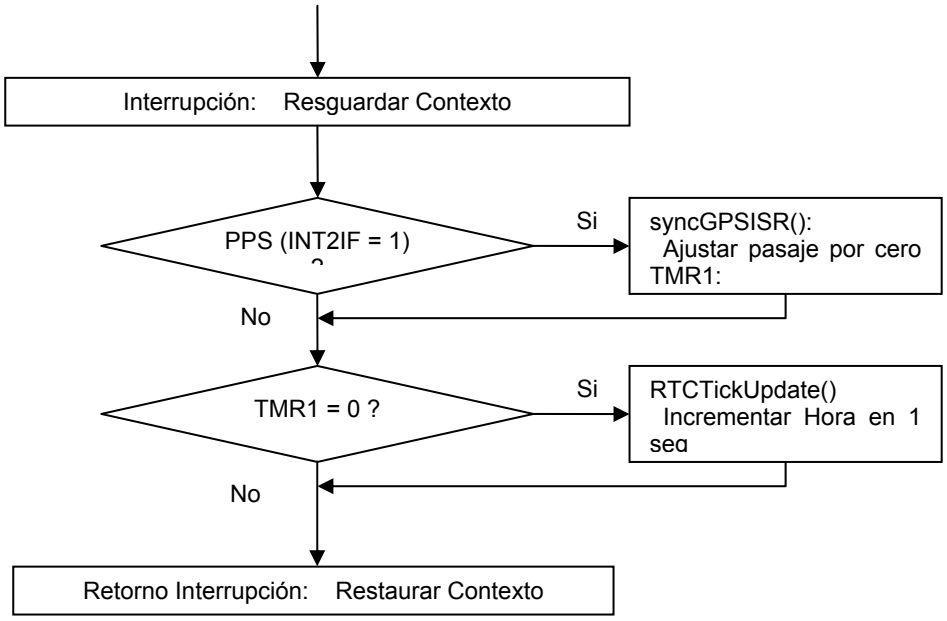


Figura 7.3: Diagrama de flujo de las interrupciones de alto nivel de prioridad.

Reloj de Tiempo Real

El RTC está implementado en los archivos RTClock.c y RTClock.h. El timer utilizado para este fin es el timer 1 con un cristal de 32768 Hz.

Dado que no se asigna prescaler y que el registro contador es de 16 bits, este timer generaría una interrupción cada 2 segundos. Sin embargo, en la rutina de atención del timer 1 se setea (siempre) el bit más significativo del registro TMR1H, con lo que se logra finalmente, una interrupción por segundo.

La función RTCTickInit() inicializa el timer y los registros TMR1H y TMR1 acorde a lo descripto anteriormente. El tiempo se registra en 47 bits en total. De estos 47 bits, 32 corresponden a la variable RTCHigh y 15 al registro TMR1 (dado que el bit más significativo se descarta por estar siempre en 1 como se explicó más arriba).

A cada incremento en el registro TMR1 se denomina tick. De esta forma el tiempo se registra como cantidad de ticks en 47 bits. Si solo se miran los 32 bits de mayor peso, es decir el contenido de la variable RTCHigh, se tiene el registro de segundos. De esta manera el rango que soporta este RTC es de 136 años.

El pulso de PPS del receptor GPS se utiliza para ajustar el valor de TMR1 que registra la subdivisión del segundo, con lo cual se logra una sincronización real con el tiempo del GPS. La función RTCGetTickCopy() toma una copia segura de los 47 bits del registro del tiempo.

La función RTCTickUpdate() es la rutina de atención para la interrupción de timer1. Esta se produce exactamente cada un segundo y simplemente incrementa el registro de segundos. También se actualizan variables que sirven para marcar cambios de segundo. Estas variables son timers2zero, timers2one y timers2toggle. Todas tienen el tamaño de un byte, si bien desde otras partes del

programa se utilizan como bits independientes. timers2zero son bits que pasan a 0 en el comienzo de cada segundo, timers2one son bits que pasan a 1 y timers2toggle son bits cambian en cada segundo.

Tomando una convención utilizada en muchos Sistemas Operativos tipo UNIX se considera que el valor cero (0 ticks) corresponde con el jueves 01 de enero de 1970 a las 00:00hs con 00 segundos. Dado que RTCHigh registra segundos, estas son algunas equivalencias:

0x00015180 = 1 día

0x00000E10 = 1 hora

0x00000708 = 30 minutos

0x0000003C = 1 minuto

0x0000001E = 30 segundos

0x0000000F = 15 segundos

0x00000001 = 1 segundo

Y este es un caso especial por su proximidad al límite del rango soportado por el RTC.

RTCHigh = 0xFFFFFFF0

Este valor corresponde al domingo 7 de febrero de 2106 a las 06:28:00. En esta situación luego de 15 segundos el Concentrador informará que la fecha es 01 de enero de 1970. Cuando el Concentrador recibe la fecha y hora del receptor GPS en formato día-mes-año, hora-minuto-segundo, convierte esta información teniendo en cuenta también la zona horaria, a la cantidad de segundos transcurridos desde el 01 de enero de 1970.

Desde cualquier tarea colaborativa puede modificarse o consultarse el estado de un bit en particular para detectar el cambio de segundo. Solo se debe tener precaución en que dos tareas colaborativas no utilicen el mismo bit. Para facilitar el uso de estos bits se utilizan estructuras de campos de bits que brindan acceso a los bits de una manera más clara. Por ejemplo, la sentencia

```
Timers2zero.T3 = 1;
```

Que puede invocarse desde una tarea colaborativa, setea el bit 3 de timers2zero en 1. Este bit volverá a 0 (cero) en el próximo cambio de segundo.

Por último la rutina de atención actualiza contadores regresivos de segundos que se encuentran en downCount[ ]. Estos contadores siempre se descuentan sin controlar siquiera rebasamiento. Pueden ser utilizados por las tareas colaborativas de manera similar a los bits, teniendo en cuenta también que dos tareas diferentes no deberían usar el mismo contador.

La función setTime() ajusta el RTC hasta nivel de segundo (RTCHigh) con el valor de tiempo recibido por parámetro en una estructura struct Time con los siguientes campos:

```
typedef struct Time {
        WORD YY;
        BYTE MM;
        BYTE DD;
        BYTE HH;
        BYTE Min;
        BYTE Sec;
        WORD mSec;
        WORD SecDiv10K;   // Second/10000.
        BYTE DoW;
        short int TZ;     // Time Zone (Argentina = -3).
};
```

Esta estructura está definida en RTClock.h. La función calcula la cantidad de segundos pasados desde el 01 de enero de 1970 contemplando años bisiestos. La información contenida en la estructura puede provenir por ejemplo de lo recibido desde el GPS.

getTime() hace el proceso inverso entregando el tiempo actual del RTC en una estructura struct Time. Esta función recibe como parámetro la zona horaria con que se quiere ver la información de tiempo, ya que el RTC siempre almacena el tiempo referido al meridiano de Greenwich (GMT).

Existen dos funciones equivalentes a las dos últimas descriptas, que son setTimeProtoFormat() y getTimeProtoFormat(). La estructura utilizada por estas dos funciones tiene la siguiente forma:

```
typedef struct TimeProtoFormat {
        BYTE YY;          // Year. 6 bits on the left.
        BYTE DD;          // Day of year. 9 bits.
        BYTE HH;          // Hour. 5 bits.
        BYTE Sec;         // Seconds of hour. 12 bits.
        BYTE CSec;        // Hundredths of second. 8 bits.
        BYTE DmSec;               // CSec/100. 8 bits.
        BYTE DoW;         // Day of Week. 8 bits.
};
```

Estas funciones sirven para traducir los valores de tiempo del RTC al formato empleado en el protocolo de aplicación utilizado entre el CC y las unidades de control.

Después de estas funciones en el archivo RTClock.c y hasta el final del mismo se encuentra todo el código relacionado con la sincronización con GPS. La función GPSInit() inicializa la UART utilizada para la conexión con el receptor GPS y le asigna bajo nivel de interrupción. GPSISR() es la rutina de atención de la UART e implementa una máquina de estados finitos que va recibiendo byte por byte las sentencias NMEA del receptor GPS.

NMEA es un protocolo de texto que utilizan los receptores GPS para recibir y entregar información. De las distintas sentencias NMEA existentes, el Concentrador solo utiliza la sentencia $GPRMC y lo que hace GPSISR() es reconocer esta sentencia, comprobar checksum y almacenarla en memoria dando aviso del arribo de información de tiempo válida.

syncGPSInit() inicializa la línea externa por la cual se recibirá el pulso PPS proveniente del receptor GPS, el cual generará una interrupción. Por último, syncGPSISR() es la rutina de atención a la interrupción generada por la llegada de un pulso PPS. Esta función ajusta el valor de TMR1 para sincronizarlo con el GPS. Además hace un cálculo del ajuste que fue necesario, que se puede consultar desde otras partes del programa.

CAPÍTULO 8: IMPLEMENTACIÓN DE LA APLICACIÓN DE RED Y OBSERVACIÓN DE RESULTADOS

RESUMEN

EL presente capítulo describe el hardware y el software implementado, así como el software que se desarrolló para la verificación de su funcionamiento. El mismo se ha instrumentado sobre la base de tres placas: Dos de provisión comercial y una de un proyecto preexistente y adecuada ad hoc.

Sobre el hardware así definido, se instaló la programación y se efectuaron los ajustes a ésta para el logro del cometido.

Se describe un escenario de pruebas a efectuar y se consignan los resultados que arrojó el prototipo para tales pruebas que a su vez sirven como consigna para la repetibilidad de la evidencia.

8.1. INTRODUCCIÓN

Para la implementación de la aplicación de prueba, se constituyó un ambiente con 4 UCs en red, denominadas UC2 a UC5, un concentrador y una terminal que oficia de CC. Para ello, se utilizó un Hardware de Desarrollo de fabricación artesanal, conteniendo 3 microcontroladores Microchip de Rango medio: 16F877/76. Los microcontroladores poseen cristales de 20MHz y se efectuaron modificaciones a la placa para adicionar cristales de 32768 HZ y vincularlos mediante una red RS-485. Este arreglo constituye un esquema de 3 UCs vinculada en red. En la placa existen otras dos unidades de rango medio (PIC 16F628) que no tienen función asignada en el ambiente de esta tesis. Adicionalmente, se ha integrado a la red otra placa comercial, conteniendo un microcontrolador 18F97J60 conformando la cuarta UC del sistema (UC5). Esta placa se asocia a la red mediante la inclusión de un conversor RS232 a RS-485.

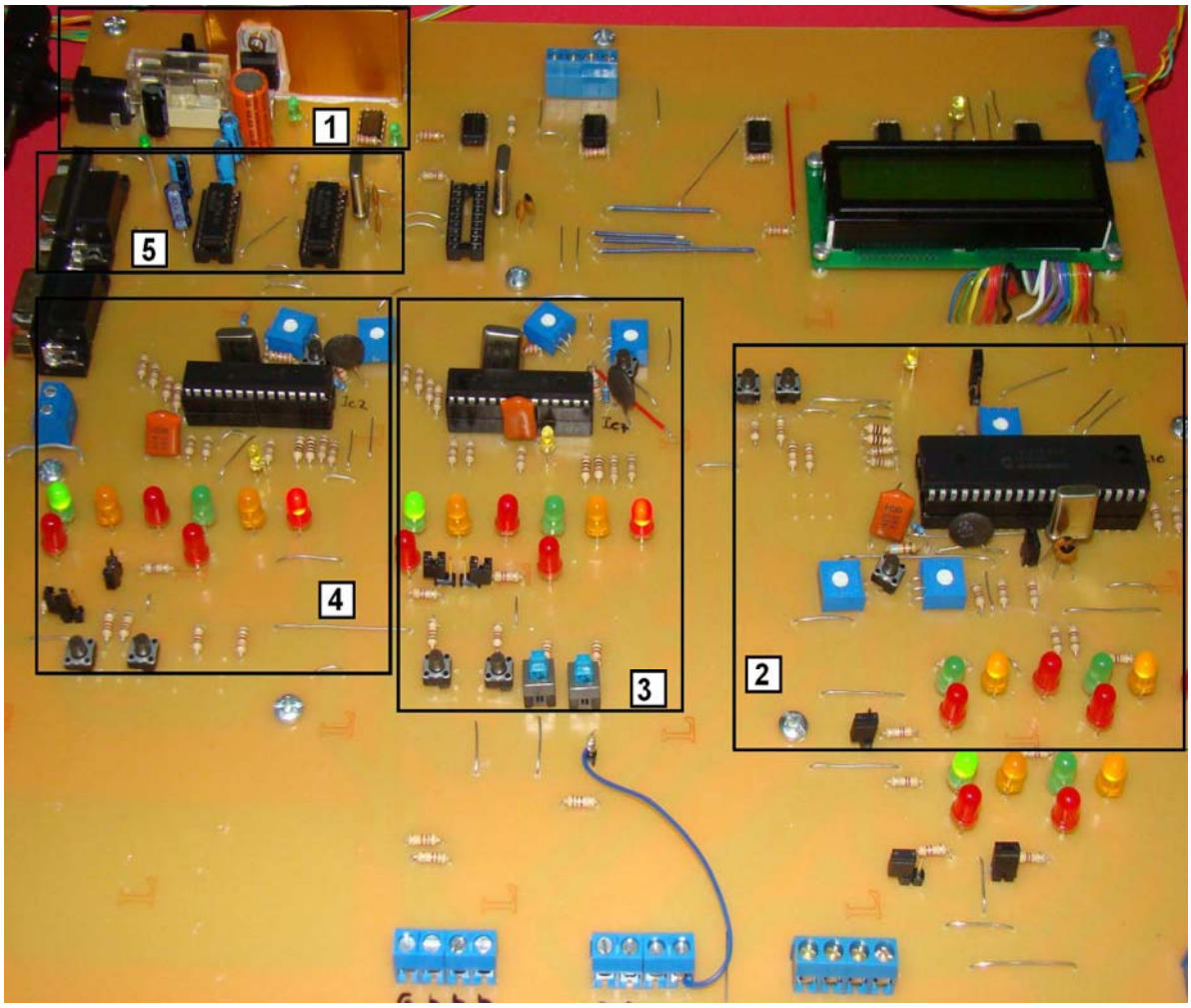


Figura 8.1: Vista de la placa prototipo de microcontroladores de rango medio en red.

La Figura 8.1 otorga una vista de la placa de hardware. En la zona central pueden apreciarse las unidades principales:

UC2 (recuadro 2, a la derecha),

UC3 (recuadro 3, al centro) y

UC4 (recuadro 4, a la izquierda.

Estas UCs contienen CPU, capacidad de memoria y dispositivos de E/S suficientes como para las pruebas de software del sistema. En la figura, dentro de los recuadros respectivos, se destaca el detalle de la periferia que poseen las unidades principales:

a) Entradas analógicas. Estas están conectadas a 2 potenciómetros para cada unidad (cuadrados azules en la figura).

b) Entradas digitales. Las mismas están asociadas a 2 pulsadores en cada unidad e ingresan por el puerto B, nibble High, que posee capacidad interruptiva por cambio de estado por flanco (botones en la zona inferior de los recuadros 3 y 4 o superior izquierda en el recuadro 2).

c) Leds como salidas digitales. La unidad UC2 posee asociados 14 leds (aunque sólo 8 son utilizados en este desarrollo) mientras que las unidades UC3 y UC4 poseen 8 cada una.

También se destaca el cristal principal de 20MHZ de cada unidad. La UC2 posee además, un display LCD de 2 x 16 caracteres (zona superior derecha de la figura), aunque este dispositivo no cumplió ninguna función en las pruebas.

En la parte superior izquierda de la figura (recuadro 1), se observa la fuente regulada.

Más abajo (recuadro 5), un microcontrolador 16F628, que actúa como puente entre la red RS-485 y una salida RS-232, útil para observar el tráfico del canal con una terminal en un modo totalmente independiente al prototipo. Esta facilidad contribuyó en forma importante en algunas cuestiones de diagnóstico conjuntamente con el Osciloscopio de dos canales con memoria.

En la figura 8.2. se otorga un diagrama circuital que responde a cualquiera de las 4 UCs que se dispusieron para la prueba. El circuito corresponde exactamente al que tienen las Unidades UC3 y UC4, aunque para las otras dos unidades la disposición es similar.

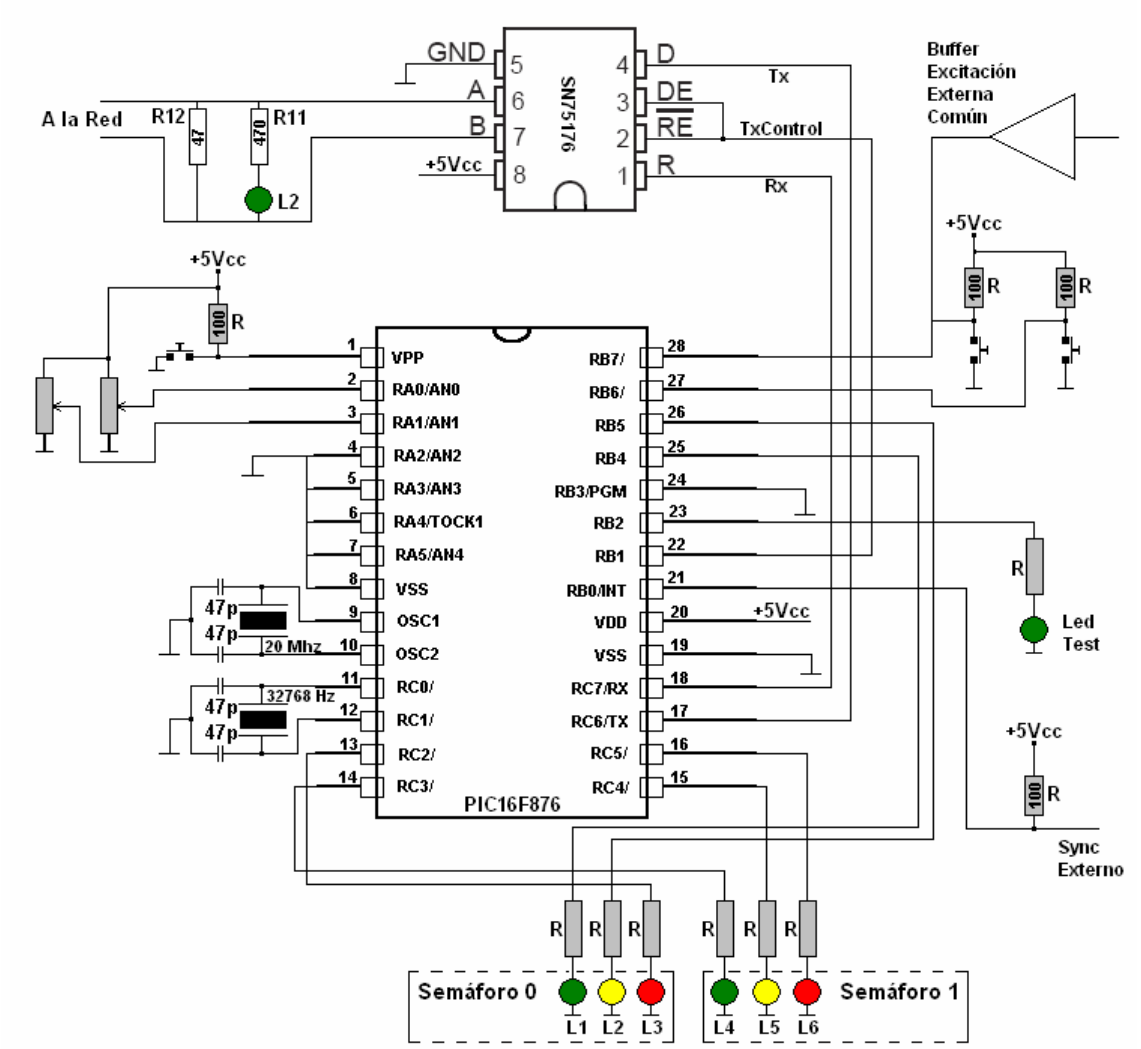


Figura 8.2: Circuito de cada unidad del escenario de prueba.

En el circuito mencionado destacaremos algunos componentes constitutivos y su relación con el escenario de prueba:

En la parte central se observa el microcontrolador de la línea 16F876 de Microchip. Asociado al PIN 1 se tiene el circuito de reset de la unidad.

Los PINs 2 y 3 se asocian a dos potenciómetros lineales de 10Kohm, para las entradas analógicas. Estas entradas permitirán simular la densidad de vehículos detectada en una calle.

Los PINS 4 a 7 son puertas que no se han utilizado y al igual que el PIN 8 (VSS) se las vinculó a tierra para evitar ruido externo. Los PINs 9 a 12 son las conexiones del cristal principal de 20MHZ y el cristal de 32768 asociado al TMR1 del reloj.

Los PINS 13 a 16, como asimismo los PINS 25 y 26 son salidas para 6 LEDs de colores (L1 a L6), que sirven para la simulación de la aplicación de usuario, esto es, dos semáforos con sus respectivas luces: Verde, Amarilla y Roja.

Los PINs 17 y 18 son las líneas de Transmisión y Recepción serie que se vinculan al conversor SN75176 que está situado en la posición superior del dibujo. Los PINs 19 y 20 son la alimentación principal del microcontrolador de 5Vcc que vienen de la fuente regulada que posee la placa. El PIN 21 es una entrada que sirve ingresar un flanco de sincronismo externo (comúnmente un GPS) que finalmente no fue utilizado.

El PIN 22 es la señal de salida necesaria para control del flujo hacia el canal RS485 en multipunto. El PIN 23 es un LED que en las aplicaciones tienen el rol de testigo de funcionamiento de la unidad. Normalmente tiene un encendido intermitente con frecuenta de 1HZ. El PIN 24 es un puerto que se conectó a tierra.

Los PINs 27 y 28 son entradas con pulsador y resistencia de Pull-Up a efectos que por defecto ingrese un valor HIGH y mantener una baja impedancia (100 ohm) por razones de ruido. En particular el PIN 28 posee asociado el ingreso de un buffer separador que permite una entrada de un evento provocado simultáneamente con las otras dos UCs de la placa, como se explicará en detalle en el capítulo 9.

En la parte superior de la figura se encuentra el circuito conversor de norma RS-232 a RS-485 cuyo funcionamiento se describiera en detalle en el capítulo 5.

En la figura 8.3, se otorga una vista de la placa comercial que ha servido para la instrumentación de la UC5, con periferia similar a la descrita y utilizando el puerto de comunicaciones RS232 que se observa a la izquierda de la figura, con el agregado de un conversor externo de norma RS232 a RS-485. El mencionado conversor es automático y por ello no ha sido necesaria la conexión de línea de control de flujo (TxControl).

Figura 8.3: Vista de la placa del Concentrador y de la UC con microcontrolador de rango alto.

Con otro ejemplar del mismo hardware, se conformó el Concentrador del sistema en este caso adicionando el uso de la etapa Ethernet que dispone la placa y de la cual observamos su conector integrado RJ485, en la parte superior derecha de la figura 8.3. Para ingresar esta placa a la red RS-485 se instrumentó una etapa de conversión dentro de la placa artesanal anteriormente descrita.

Las unidades de control UC2 a UC5 fueron dotadas de un software que responde a la estructura y funcionalidad descrita en los capítulos correspondientes.

8.2. TERMINAL DE DIAGNÓSTICO

A efectos de concreción de pruebas y diagnóstico de los microcontroladores en red, se adaptó un software que oficiaba de terminal para uso en las cátedras. El mencionado software, originalmente denominado **ArquiCom**, fue dotado en sus últimas versiones las capas de aplicación y transporte del protocolo Mara-1. Este terminal es un producto stand-alone y corre sobre cualquier PC con Windows 98, 2000 o XT.

El terminal posee conectividad en RS-232 y sobre esta modalidad instrumenta el protocolo estudiado. En este modo debe ser conectado a la placa Alfa31 mediante un cable MODEM null, a su conector DB9 (recuadro 5 de la Figura 8.1).

El terminal también permite la utilización de Ethernet mediante la utilización de Sockets TCP, encapsulando el protocolo de estudio bajo TCP/IP. Esta conexión es la adecuada para conectarse con el Concentrador.

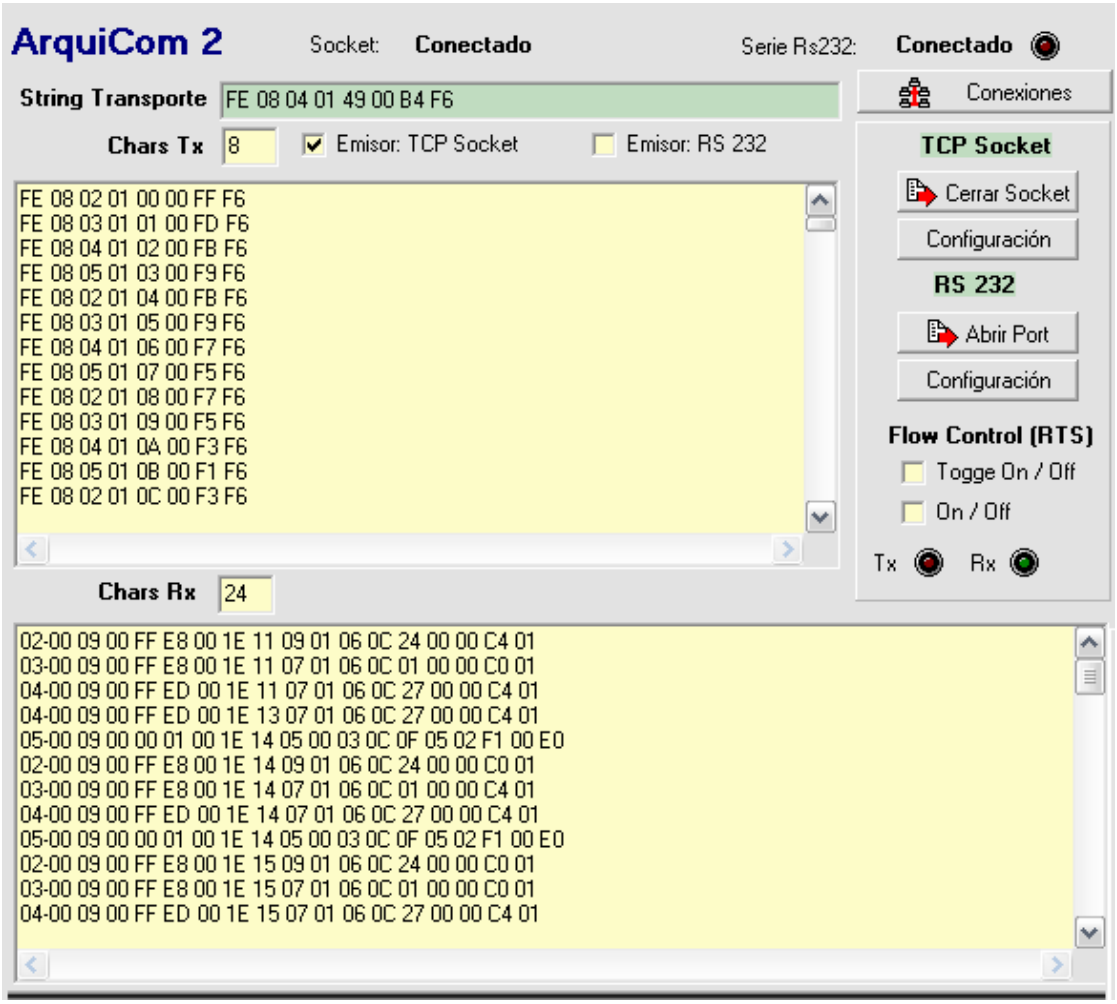


Figura 8.4: Pantalla de configuración de conexiones y Terminal de caracteres.

En su interfase con el usuario el terminal dispone pantallas que permiten la observación de las tramas de protocolo, tanto en la capa de transporte como en la de aplicación. Los valores son presentados en Hexadecimal lo que conforma un ambiente

apto para la evaluación del protocolo. La figura 8.4 presenta una vista donde puede apreciarse el panel de control de conexiones (arriba – derecha) y la terminal de caracteres. En la figura se está exhibiendo la representación hexadecimal de una trama.

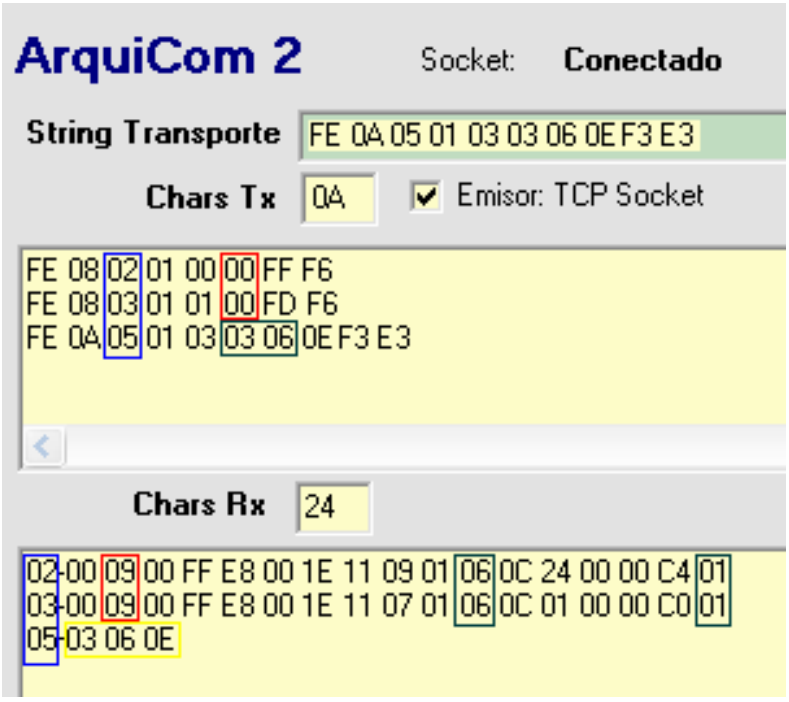


Figura 8.5: Vista de la emisión de comandos y su respuesta.

En la Figura 8.5, se observa en la parte superior correspondiente a la ventana de transmisión, las tramas correspondientes a tres comandos para las UCs 02,03 y 05 (denotados con recuadro azul). De estos tres comandos los dos primeros son un requerimiento de Estado (00h: denotado con recuadro rojo) y el tercero un requerimiento de control (03h) del puerto 06h (denotados con recuadro verde). Estas cadenas de bytes son las que se han descrito en los capítulos 3 y 4 correspondientes a las capas de aplicación y transporte del protocolo.

En la parte inferior de la misma figura, se observa la zona correspondiente a la ventana de recepción del terminal. En la misma, se presenta la respuesta correspondiente a cada uno de los tres comandos emitidos en secuencia. La presentación elegida en este caso, retorna sólo los bytes de aplicación del protocolo, precedido por la dirección de la UC correspondiente (denotados con recuadro azul).

En las dos primeras líneas se observa la respuesta a un requerimiento de estado, que en este caso retorna 9 bytes de variables de estado de la CPU (denotado en rojo). Se observa también el retorno de 6 bytes de estados de los puertos digitales y 1 byte de variables analógicas (ambos denotados en verde).

La tercera línea devuelve la respuesta exitosa del comando requerido (denotado en amarillo).

Interfase Hombre-Máquina

En sus últimas versiones, se ha dotado a la terminal de una interfase visual, que permite una apreciación de más alto nivel de la aplicación de usuario. Esto es, mediante la existencia de íconos que modifican su estado en función de la información

que surge de las UCs en campo y mediante la existencia de botones, que permiten la emisión de los comandos de aplicación, que se han descrito en el capítulo 3 referente a la capa de aplicación. La figura 8.6 presenta una vista de esta interfaz Hombre – Máquina, donde pueden apreciarse los símbolos de los semáforos vinculados a cada una de las UCs y los valores de estado de las variables analógicas (Como barras de desplazamiento y valores numéricos en texto).

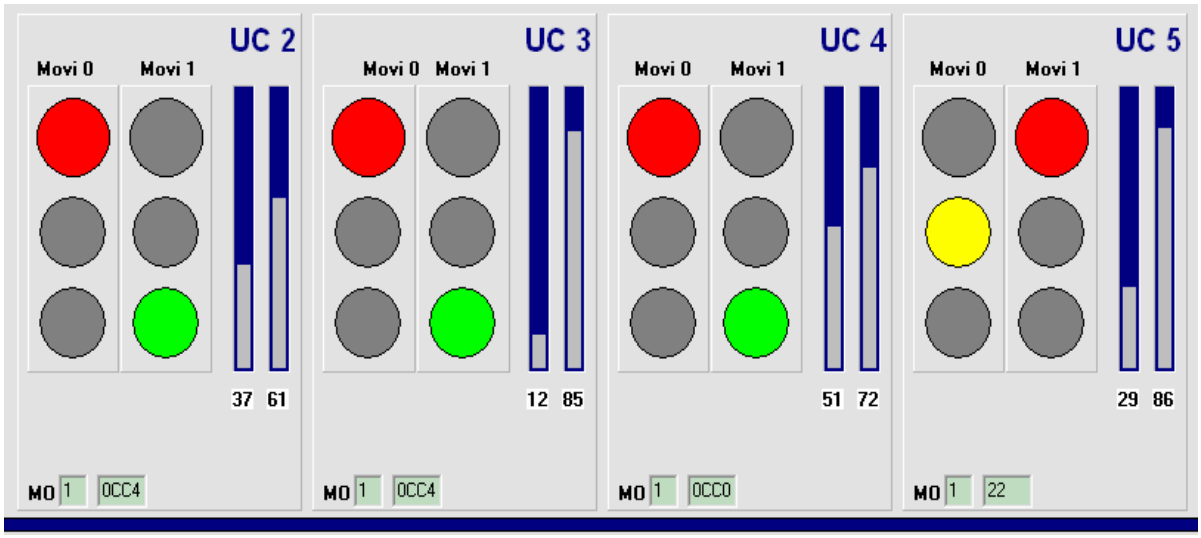


Figura 8.6: Pantalla de la Interfaz Hombre - Máquina.

Ésta sería una versión reducida, meramente representativa, de una MMI (Man-Machine interface) de un SCADA.

8.3. CONFORMACIÓN DEL ESCENARIO DE PRUEBA Y DIAGNÓSTICO

Para el Test del sistema en su conjunto, se tuvo en cuenta un Protocolo de Prueba que de alguna manera permitiera ameritar cada una de las partes integrantes del desarrollo. En función de ello, se enunciaron una serie de ítems que implican algún aspecto de prueba y que se lo vincula con alguna evidencia reproducible sobre el ambiente de prueba. El Protocolo de Prueba tentativo se esboza en la Tabla 8.1 al final de este capítulo y es el que se reproduciría en la presentación de esta tesis. Obviamente el mencionado Protocolo no es exhaustivo y sólo incluye aquellos aspectos más relevantes sugeridos. Ha sido desglosado en los distintos aspectos que posee la tesis y en relación con los bloques funcionales y sus capas.

Las columnas de la tabla Tabla 8.1 tienen el siguiente significado:

Módulo: Etapa se Hardware / Software que se somete a prueba.

Acción: Destaca la acción concreta que debe aplicarse al módulo a efectos de provocar la respuesta.

Verificación / Resultado: Describe la respuesta que se espera conforme al estímulo otorgado al módulo. En algún caso esa respuesta es evidenciable mediante los dispositivos de salida (Display, Leds, HMi de Arquicom y otros) y en otro caso mediante análisis de la trama de protocolo que se presenta en el terminal.

Observaciones: Otorga algún detalle de la prueba.

Tabla 8.1: Protocolo de Pruebas de las unidades Funcionales del Sistema.

Unidad de Control				
#	**Módulo**	**Acción**	**Verificación / Resultado**	**Observaciones**
1	Etapas de salida de la UC.	Arranque en intermitente de los LEDS amarillos y de Test.	Dentro de los primeros 10 segundos se debe evidenciar sincronismo: Led Test permanente.	Correlacionar con el resto de las UCs. Observar sincronismo.
2	Etapas de salida de la UC.	Arranque de los tiempos individuales de luces del semáforo. Intervalos de 15 segundos para luz verde y roja.	Dentro del minuto de post arranque se debe iniciar la función normal de semaforización.	Debe comenzar la intermitencia del Led Test, indicador de fin del estado transitorio.
3	Sincronización automática de una unidad respecto al resto, ante re-arranque.	Apagar y Encender una unidad. Verificar Sincronismo.	Verificar Pasos del punto 1 y verificar enganche de luces con resto del sistema.	**Emitir Com 05 unicast desde la terminal**
4				

Concentrador				
#	**Módulo**	**Acción**	**Verificación / Resultado**	**Observaciones**
1	Arranque y Puesta en Hora del Concentrador.	Concentrador: Arranca con Hora semilla. Emitir Comando de PeH desde la Terminal.	Concentrador: Adopta hora de la terminal.	Se obtiene la Hora actual en el Display
2	Función de Sincronismo:	Enviar Eventos en Broadcast a las	Comparar el Time Stamp de	**Emitir comando 07 broadcast desde la**

	Sincronización de UCs.	UCs y verificar hora del suceso.	eventos correspondientes. **Deben converger en el orden de +/-0,3 ms.**	**terminal**
3	Función de puente Ethernet – RS 485.	Emisión de comandos desde el CC a las UC. Emisión del comando 00h en forma continuada.	Recepción de respuesta a comando 00h. y la recepción continua de las respuestas sin retrys.	**Emitir comando 00 unicast en Round-Robin desde la terminal**

Interacción UC y Centro de Control				
#	**Módulo**	**Acción**	**Verificación / Resultado**	**Observaciones**
	Comunicaciones	Abrir conexiones Ethernet y RS-232 con el Concentrador.	Establecimiento de la conexión.	
3	Función de puente Ethernet – RS 485 del Concentrador	Emisión de comandos desde el CC a las UC.	Recepción de respuesta. Verificar la emisión del comando 00 y la recepción contínua de las respuestas sin retrys (Errores de checkSum detectados por la terminal o Retrys).	**Emitir comando 00 unicast en Round-Robin desde la terminal.**
	Acciones de Control del CC	Emitir un comando para apagado y encendido de Leds.	Verificar Apagado/Encendido	**Emitir com 03 en Unicast.**
1	Escritura y Lectura de Parámetros en EEPROM	Emitir un comando de escritura.	Verificar con comando de lectura.	**Emitir com 08 y 09 en Unicast y broadcast.**
2	Provocación de eventos individuales en UC.	Operar pulsadores individuales de eventos.	Registro de eventos con horario secuencial (SOE)	

3	Provocación de eventos simultáneos en UC.	Operar pulsador común de eventos simultáneos.	Comparar el Time Stamp de eventos correspondientes. **Deben converger en el orden de +/-0,3 ms.**	
	Entradas Analógicas	Cambiar posición potenciómetros	Verificar cambiso consonantes en el mímico de la UC respectiva	
	Comando de usuario	Cambiar de PGM (programa de semaforización a una o varias UCs)	Verificar el Cambio a emergencia (luces amarillas en intermitente) y el cambio a normal (ciclo de 30 segundos de semaforización).	**Emitir Com 10 en Unicast y broadcast.**
	Requerir Hora y sesgos de diagnóstico a UCs	Emitir comando de requerimiento de hora.	Verificar que los segos oscilen en el orden de +/- 10 unidades (+/- 0,3 ms)	**Emitir com 0A en Unicast.**
	Sincronismo	Cambiar intervalos de la PeH a 10 y 20 segundos.	Verificar el empeoramiento de la exactitud en el registro de eventos simultáneos.	**Emitir Com 3B al Concentrador. Manejar pulsador de eventos simultáneos en UCs.**

CAPÍTULO 9: SINCRONIZACIÓN DE MICROCONTROLADORES

RESUMEN

En el presente capítulo se enfoca en el problema de la sincronización de relojes de los microcontroladores en la red, teniendo en cuenta su utilización para adquisición de datos y control.

Es importante remarcar que el problema de sincronización de relojes sigue siendo un área activa de investigación en el contexto de los sistemas distribuidos. Esta temática, en el contexto de microcontroladores interconectados, puede aprovechar lo ya estudiado considerando que existen restricciones, aplicaciones y características específicas.

El tema representa un volumen importante y de un potencial para ser por sí mismo un tema de tesis. No obstante, como el tema no puede ser dejado de lado por la relación estrecha que tiene con el protocolo que se ha definido, se pretenden otorgar aquí los aspectos vinculados con éste, aportando algunos valores de mérito que justifiquen la técnica utilizada.

Los problemas como la evaluación de la sincronización, tanto analítica como experimental, son de vital importancia en lo relacionado con los sistemas SCADA (Supervisory Control And Data Acquisition).

9.1. INTRODUCCIÓN.

La problemática de sincronización tiene como escenario la Red 1 implementada con Microcontroladores bajo norma RS-485 como la que se presenta en la Figura 9.1, aunque el concepto se puede extender a cualquier red que admita transmisión en **Broadcast**, tal el caso de Ethernet.

Los dispositivos Microcontroladores consignados en la figura, tienen la misión de interactuar con el campo para el control y la adquisición, en la modalidad descripta en el Capítulo 2 de esta tesis.

A su vez varias de estas redes pueden ser agrupadas mediante el uso de concentradores que entre sí y con el motor del SCADA configuran la Red 2. Por ello, en adelante denominaremos a las primeras como subredes.

La cantidad de nodos responde a variaciones, configuración particular y complejidad del sistema a tratar. Todos los dispositivos de las subredes se encuentran interconectados mediante una esquema multipunto, con velocidades de comunicación relativamente bajas: 9600, 28800, 57600 bps. El Concentrador tiene una misión primordial que es la de ser puente entre cada una de las subredes multipunto - a la que cada Concentrador está asociado - y la Red 2 que sería de mayor velocidad, a efectos de multiplicar la cantidad de dispositivos a interconectar.

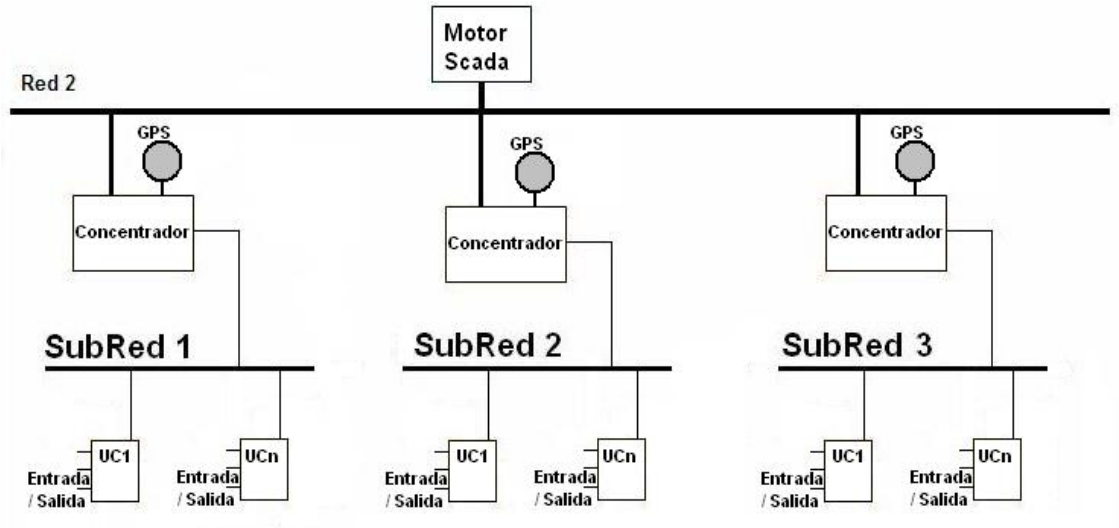


Figura 9.1: Contexto de Sincronización de Microcontroladores Interconectados.

La adquisición de datos, es una de las tareas definidas para el sistema. Por ello, cierto tipo de datos (analógicos o digitales) que sean definidos como eventos deben tener en su registro, una marca de tiempo (**timestamp**) asociada, para que pueda ser posible crear una secuencia de ellos y establecer una relación causa y efecto. Un evento se define como un cambio de estado de su variable asociada. En particular, un evento digital sería el pasaje de 0 a 1 ó de 1 a 0. Para posibilitar el registro, el estado anterior del evento debe estar previamente almacenado.

Los microcontroladores pueden provocar una interrupción ante el cambio de estado de un puerto. La interrupción ejecuta la rutina de servicio asociada en unos pocos

microsegundos y en ese tiempo, se lee el valor del reloj de tiempo real (**RTC**: Real Time Clock) contenido en la unidad, se determina el nuevo estado de la variable y todo este conjunto se almacena en una pila secuencial de eventos destinada a tal fin.

En la sincronización de sistemas de una única CPU (o microcontrolador) no se requiere ninguna consideración especial en el diseño del software, ya que existe un reloj único que proporciona en forma regular el tiempo en cada momento y consecuentemente, todos los eventos que se registren tendrán una secuencia natural y un registro que utilizan una **única** base de tiempo como patrón de hora. Los sistemas distribuidos en cambio, tienen un reloj por cada CPU del sistema, por lo que es fundamental una coordinación entre todos los relojes para tener una hora única, o con más propiedad, una hora lo más aproximada posible a la que se defina como patrón.

Por esto, se debe definir alguna representación de tiempo y además, los dispositivos UC1 a UCn de cualquiera de las subredes de la figura deben aceptar sincronización horaria por un dispositivo externo. Un factor de mérito deseable entonces, es la convergencia de hora que presentan cada una de estas unidades respecto al reloj patrón. Este factor se denomina exactitud, debe ser acotado y con la tecnología actual, es normal obtener un valor de éste que se encuentre por debajo de 1 milisegundo (1ms).

La sincronización así definida haría posible presentar con consistencia, una secuencia de eventos ocurridos en un ambiente (eléctrico, industrial u otro similar), vinculado a microcontroladores del SCADA, independientemente de la unidad y la subred en las cuales se hubiere registrado cada evento. La exactitud de esta secuencia, sería determinada por la convergencia que guardaren los RTC de cada una de las unidades a través del tiempo. Es por ello que la sincronización, es elemento vital en la presentación de los resultados o en la evaluación del estado del sistema que es presentado al usuario.

Los osciladores de cada microcontrolador son ligeramente diferentes. Típicamente las mismas frecuencias nominales de oscilación, difieren entre sí en alrededor de 30 partes por millón (ppm) en cristales estándar y como consecuencia todos los relojes asociados a este tipo de cristales, sufren un desfase y deben ser sincronizados periódicamente.

En una instalación como la representada en la Figura 9.1, la sincronización no es trivial, porque se debe realizar mediante mensajes por la red, cuyo tiempo de transmisión puede ser variable debido principalmente variabilidad del reloj en el dispositivo que sincroniza y por otro lado las propias derivas de los relojes de cada unidad, efectos tales que conjugadamente, llevan a sesgos apreciables en un momento dado. Dado que todos los dispositivos de la red tienen la deriva de hora aludida, debe elegirse uno de ellos como “patrón” de hora y el resto de los individuos de la red debe ser sincronizado mediante algún procedimiento con la hora que indica ese patrón. De esta forma, se llega a lo que se da en llamar **Sincronización Interna** [2] [15].

En base al alcance de la distribución física, similar tipo de tecnología y necesidad de contar con una cantidad acotada (alrededor de 30) de dispositivos en una zona o área de trabajo, se definió que el candidato ideal para la tarea de patrón o master de la sincronización sea el Concentrador para cada subred.

En efecto, ya se observó oportunamente que este dispositivo ocupa un lugar preponderante, debido a su actuación como puente o gateway entre dos redes de

distinta velocidad y alcance. No obstante para esta función de puente, el Concentrador se comporta como un elemento pasivo y no necesita efectuar análisis de los paquetes que intercambia con ambas redes. Como veremos luego, la sincronización trae otra complejidad para este dispositivo, debido a que debe decidir el momento en que efectúa la sincronización.

Como queda esbozado en la Figura 9.1, se podrán tener "islas" o subredes, cada una sincronizada por su Concentrador respectivo y con alguna exactitud a determinar. Dado que cada Concentrador puede tener asociado un dispositivo que otorgue el **UTC** (Tiempo Universal Coordinado), como ser un GPS, podría lograrse la llamada **Sincronización Externa** de todo el sistema.

9.2. DESARROLLO

Sin lugar a dudas, una de las primeras líneas de estudio se enfocó al análisis de las estrategias de sincronización existentes en el contexto de sistemas distribuidos [15]. Dicha estrategia es esencial para resolver problemas tales como ordenamiento de eventos (envío y recepción de mensajes, lanzamiento de eventos, etc). Era esperable llegar a una propuesta de sincronización que, por un lado tenga en cuenta las limitaciones de capacidad de los microcontroladores, pero por el otro que aproveche las ventajas de un sistema que podría considerarse de tiempo real estricto, con cotas de tiempo de respuesta a los eventos conocidas de antemano. Esta característica puede considerarse propia de los sistemas basados en microcontroladores, donde en muchos casos la cantidad de eventos por unidad de tiempo es acotada y el software que se ejecuta es sencillo de analizar y también estimar en cuanto a tiempo de ejecución.

Una vez decidida la estrategia de sincronización, es importante el análisis exhaustivo y lo más formal posible desde dos puntos de vista:

- Requerimientos impuestos a los microcontroladores y a la red de interconexión de los mismos. Esto debe tener en cuenta que los microcontroladores no se dedican exclusivamente a la sincronización.
- Cotas de error de sincronización, ya que toda sincronización tiene en sí misma una definición y especificación de error.

A efectos de obtener factores de mérito de la solución, ha sido importante la implementación y experimentación de un entorno real. Como mínimo, fue necesario contar con resultados experimentales para comparar y analizar, en función de los cuales avanzar hacia una implementación probada. A continuación se hace referencia a aspectos generales de la problemática de sincronización, combinados con aspectos de la implementación particular, efectuada con microcontroladores de la línea de un fabricante del mercado.

Concentrador

El Concentrador es un elemento que fue definido inicialmente para operar entre dos redes y adicionalmente, tiene capacidad de comunicarse mediante otro canal serie bajo norma RS-232. El Concentrador gestiona las comunicaciones entre un Centro de Control (donde reside el Motor SCADA de la Figura 9.1) y las unidades de control. Para la implementación de referencia que se describe en esta sección, este dispositivo está compuesto por un hardware y un software específicos. El hardware

está basado en el microcontrolador PIC 18F97J60 de Microchip, unidad con la ventaja de poseer embebido un módulo Ethernet y dos puertos serie. El software del Concentrador fue concebido sobre la base de la técnica de programación denominada multitarea colaborativa. Por lo tanto, la estructura del Concentrador es la de un programa con un único hilo de ejecución, que invoca secuencialmente a diferentes tareas (colaborativas) de manera iterativa. Las tareas colaboran con el fin común, resolviendo su trabajo en un tiempo apropiado o resolviendo parte de su trabajo y entregando el control. De esta forma, se asegura que todas las tareas tendrán oportunidad de realizar su trabajo.

El reloj de tiempo real que tiene el Concentrador para mantener su propia hora es obtenido mediante un temporizador (timer), hardware interno alimentado directamente por un cristal de cuarzo de 32768 Hz. Este reloj, con una elevada exactitud inherente (del orden de 30 ppm), es tomado como "patrón" de sincronización interna para la subred subyacente.

Estrategia de Sincronización Interna

Dado que los microcontroladores a sincronizar son de mediana o baja capacidad, se decidió una característica importante de la red de interconexión, casi directamente orientada hacia la posibilidad de sincronización, cual es el broadcast físico de datos. La gran mayoría de las redes físicas más populares cuentan con esta posibilidad, entre ellas, las que están bajo normas EIA-485 y Ethernet. Ello, permite que haya un controlador que establezca la hora en toda la red con una única transferencia de datos.

De manera prácticamente simultánea, todos los microcontroladores reciben la información necesaria para la sincronización. Se decidió entonces, implementar una estrategia de sincronización en la cual el dispositivo que arbitra el flujo de datos en cada subred multipunto (cada Concentrador), sea el encargado de enviar en forma periódica y entrelazada con los mensajes de datos, un mensaje de sincronismo en broadcast para todas las unidades de la subred. Con esta metodología, cada unidad de la subred que recibe el mensaje de sincronismo efectúa inmediatamente de su arribo, el ajuste de su reloj interno.

En una primera aproximación, el envío de hora a la red se realizó dentro de la rutina principal del Concentrador, enviando cada uno de los bytes como una tarea más dentro del ciclo de tareas colaborativas, es decir por encuesta (polling) de la UART. Para ello, se construye el paquete de Puesta en Hora, que llamamos **PeH**, definido en el Capítulo 3, correspondiente a la Capa de Aplicación, que contiene desde el año actual hasta una fracción de segundo con cuatro cifras significativas (décimas de milisegundo o **ddm**).

Este paquete se comienza a enviar desde su primer byte a la red RS-485, inmediatamente después de requerida la hora local. Enviado el primer byte, se van enviando los sucesivos, hasta un total de **15 bytes** integrantes del paquete, a medida que la UART está disponible (con encuesta). Enviado el último byte, se inicia un temporizador de algunos segundos, terminado éste, se envía otra nueva PeH y así sucesivamente. Este intervalo del temporizador entre una PeH y la siguiente, es regulable entre 1 y 255 segundos y en el experimento inicial se lo fijó en valores que oscilaban entre 2 y 20 segundos.

Se fijó como parámetros del mensaje **9600, N, 8, 1** (velocidad en bps, paridad, largo de palabra, bits de stop). Para una velocidad determinada se cumple que:

$$\Delta t = N \times 10 \times 1/Vt \quad (1)$$

$$h_{pdo} = h_o + \Delta t \quad (2)$$

donde:

Δt = tiempo de transmisión del mensaje en segundos.
N = Cantidad de bytes del paquete de PeH.
Vt = Velocidad de transmisión, en bps, del canal RS-485.
h_o = Hora inicial (hora enviada en la PeH).
h_{pdo} = Hora a imponer al destino con la PeH.

Mientras que la ecuación (1) otorga el tiempo que el mensaje tarda en llegar al destino, la ecuación (2) otorga la hora corregida a partir de la hora patrón **ho** que otorga el Concentrador.

En la implementación se pudo observar que con esta técnica de inclusión de la sincronización entre las tareas estándares del Concentrador, se obtiene un ajuste de hora aceptable, pero con un delta temporal que resultó con fluctuaciones por encima del valor teórico que se obtiene de la ecuación (1). Aunque esa variación es pequeña, es fluctuante e impacta linealmente sobre la hora con que se corrige la hora local en cada UC como se aprecia por la ecuación (2). Las mediciones efectuadas con esta técnica arrojaron una fluctuación que oscilaba en +/- 1 ms, teniendo en cuenta que el valor teórico que resulta de la ecuación (1) para los parámetros del mensaje consignados es de 15,625 ms. Este error se consideró excesivo, conforme a la expectativa de exactitud que se esperaba, por lo que se efectuó un análisis más exhaustivo del problema

Se observó que independientemente del tiempo que tarde el mensaje, las fluctuaciones que posee este tiempo, provocadas por la modalidad de envío, impactan directamente sobre la hora que se le impone a cada una de las UC sincronizadas. En otras palabras, las UC reciben la fluctuación y están permanentemente adelantando y atrasando a consecuencia que el término Δt no es constante. Este efecto, es independiente de aquel de deriva que poseen los cristales locales. Por supuesto, ambos efectos actúan por superposición y en algún caso pueden compensarse resultando en una hora más exacta respecto a la hora real y en otro caso sumarse, empeorando aún más la situación.

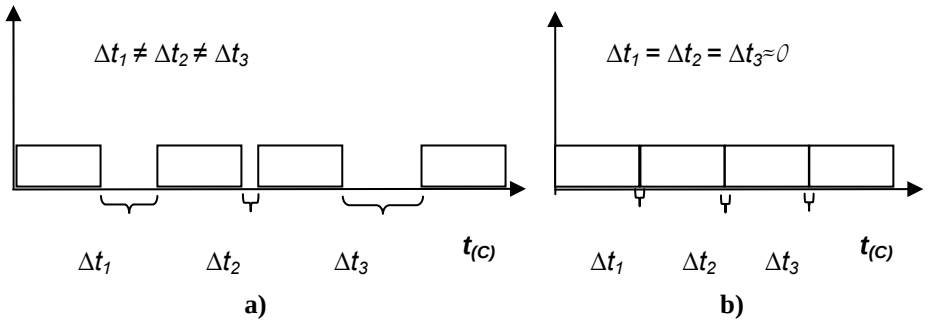


Figura 9.2: Secuencia de Bytes de PeH: a) por Polling, b) por Interrupciones.

Se dedujo y posteriormente se observó en osciloscopio, que entre cada uno de los bytes integrantes del paquete de PeH, existían pequeños tiempos variables y de duración aleatoria, que se sumaban al tiempo total de transmisión y eran los responsables de la fluctuación de la PeH que se aplicaba a las UCs. Este efecto

puede apreciarse en la Figura 9.2 a), dibujado fuera de escala para hacer más sencilla su identificación.

Para resolver el problema de la falta de regularidad en el envío de los bytes de sincronización, fue necesario cambiar el esquema de tareas colaborativas dentro del Concentrador, haciendo que dentro de las mismas el envío de puesta en hora se realice con una cadencia más exacta y que Δt pase a un valor cuasi-constante, con fluctuaciones casi inexistentes, tal como se muestra en la Figura 9.2 b).

La implementación de esta solución consiste en cambiar la forma en que se envía el comando de puesta en hora, resolviéndolo mediante interrupciones. En efecto, al llegar el instante correspondiente a la emisión periódica de PeH - determinado ahora también por un timer independiente que interrumpe -, se ejecuta la rutina de rescate de la hora local, armado del paquete y activación del mecanismo de interrupciones de la UART a efectos que cada byte de la PeH sea requerido por interrupciones [7].

La interesante particularidad de este mecanismo hace que los tiempos existentes entre byte y byte del mensaje de PeH sean muy cercanos a cero, convirtiendo el mensaje asincrónico por naturaleza en cuasi-sincrónico y con una duración más predecible tal cual se aprecia en la Figura 9.2.b). De esta forma, una vez que llegue el paquete de PeH a las unidades de la red, éstas podrán sumar a sus relojes propios un Δt, que ahora pasa a ser una magnitud fija y conocida. Se debe tener en cuenta que este valor es una constante teórica conforme a cada velocidad, que es introducida en el software de las UCs como una constante y que no tiene ningún refresco dinámico, de ahí la importancia de contar con un valor predecible y acotado.

9.3. Escenario de Prueba y Resultados Obtenidos

La Figura 9.3 muestra el pseudocódigo del software que se ejecuta en cada UC de la red a sincronizar, que no deja de ser similar a la mayoría del software de control embebido: una iteración con un conjunto de evaluaciones para registrar cambios de estado y procesar comandos de control. Su esquema simplificado es:

```
(1)  Configurar/Inicializar (timer/s, UART/s, etc.)
(2)  Loop forerever Do
(3)    If (hay un trama a procesar)
(4)          Procesar trama;
(5)    If (hay un comando recibido a procesar)
(6)          Procesar recepción de un comando;
(7)    If (hay respuesta de comando a procesar)
(8)          Preparar la respuesta del comando;
(9)          Iniciar la respuesta del comando;
(10)   Registrar estado
(14) End Loop
```

Figura 9.3: Pseudocódigo del programa de una UC.

Lo relacionado con las tramas (línea 3) corresponde al procesamiento de las comunicaciones y lo relacionado con los comandos (línea 6), corresponde al sistema SCADA en general y con el mecanismo de sincronización en particular, dado que uno de los tipos de comandos a procesar es el de sincronización, donde debe asignar como nueva hora local la recibida. Como el comando PeH es emitido en broadcast, no habrá mensaje de respuesta.

Con el fin de demostrar las mejoras que suponen teóricamente los cambios propuestos en el apartado anterior, se realizaron una serie de pruebas en un entorno de trabajo que respeta la filosofía esbozada en la Figura 9.1, con tres unidades de control y un concentrador. Además, una Terminal de Diagnóstico oficia de Centro de Control y emite mensajes mediante bajo el protocolo desarrollado. Dicho entorno se presenta en la Figura 9.3.

La placa prototipo contiene tres microcontroladores de la familia Microchip (PIC 16F877 y 16F876 [6]): UC1, UC2 y UC3 que son las unidades principales. Contienen CPU, capacidad de memoria y dispositivos de E/S suficientes como para las pruebas de software del sistema. Cada UC posee un cristal de 32768 Hz asociado a su Timer 1, que lleva el RTC en cada unidad.

En la Figura 9.3. se destaca como ejemplo para la UC1, un detalle de la periferia que poseen las unidades principales: a) Entradas analógicas b) Entradas digitales y c) Leds como salidas digitales y su cristal de RTC. El resto de las unidades cuenta con igual periferia, aunque ello no fue representado en el dibujo por mayor claridad.

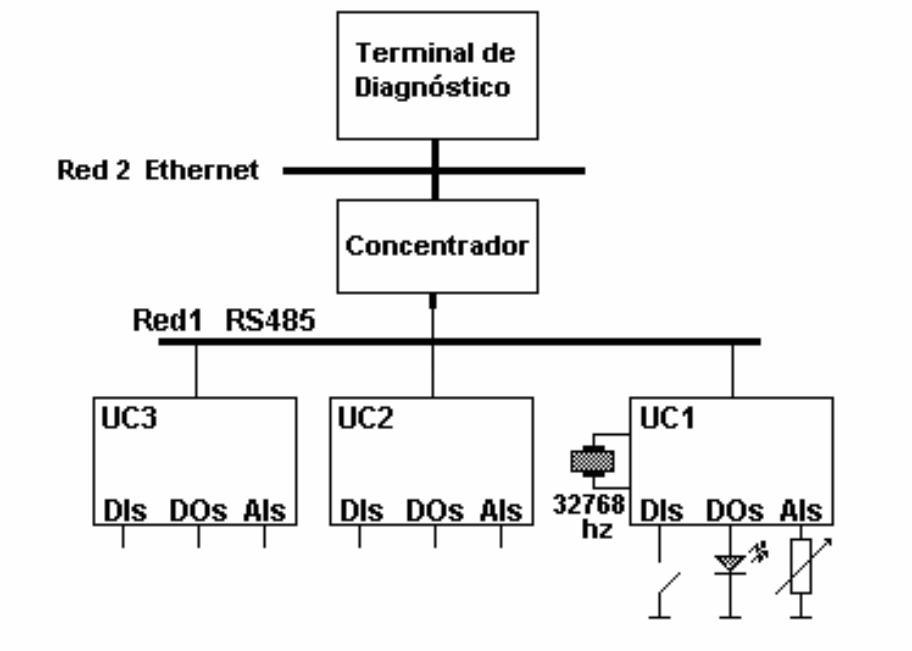


Figura 9.3: Esquema del Ambiente de Evaluación para Sincronización.

Por otro lado, se ha tomado una placa ya construida sobre la que se ha desarrollado el software del Concentrador sobre una arquitectura basada en un microcontrolador de la línea microchip 18F97J60 con conexión Ethernet y serie embebidas. Éste posee también un cristal de 32768 Hz asociado a su Timer 1, que lleva su propio

RTC. El Concentrador está vinculado a la Terminal de diagnóstico montado sobre una PC mediante la conexión Ethernet. El Concentrador emite periódicamente el comando de PeH, cuya estructura fue definida en el Capítulo 3 y que se repite a continuación por comodidad:

Denominación	**Preámbulo de comunicaciones**	**Año, día, Hora, Segundo**	**Fracción de segundo**	**BCC**
Cant. de bytes	**5**	**4**	**1**	**2**

Donde:

- El preámbulo de comunicaciones incluye la cantidad de bytes del mensaje (15) y las direcciones de origen y destino de UCs (en este caso FFh: Broadcast).
- En un bloque de 4 bytes se codifica el año, el día dentro de éste, la hora y el segundo dentro de ésta.
- En un bloque de dos bytes se codifica la fracción de segundo.
- El bloque final del mensaje de 2 bytes es el checksum de todo el paquete.

Todo el software involucrado se implementó de manera efectiva. Las condiciones destacables de las pruebas fueron:

- En la llegada a una UC, la misma calcula el **sesgo** existente entre su hora local y la que impone el mensaje de PeH (un valor positivo del sesgo indica un adelanto de la UC) y se deja registrado como una variable. Este sesgo se calcula a efectos de poseer datos de la calidad de la PeH. El Valor del sesgo se lo presenta en el protocolo, en unidades del contador TMR1 dominado por la frecuencia del cristal de 32768 Hz. Una unidad de tiempo de este contador determina una granularidad de:

  (1 / 32768) s = 0,030 ms

  Si bien esa es la granularidad máxima que puede obtenerse por el cristal utilizado, se ha elegido que el tiempo se represente con el grado de exactitud ya apuntado de 4 cifras significativas de la fracción de segundo. O sea que la exactitud elegida en la representación, de 1 décima de milisegundo **(0,1 ms),** es un poco peor que la granularidad que se posee e inherente al cristal utilizado.

- El Terminal por otro lado y con intervalos regulares, mediante un comando particular (Comando 00h, descrito en el Capítulo 3), recaba el sesgo calculado en la última PeH ocurrida, conjuntamente con un número identificador de cada PeH, a efectos de comparar sesgos de UC correspondientes a una misma PeH. Estos valores son presentados en la Terminal y luego comparados en una planilla de cálculo.
- En el experimento se tomaron 4896 muestras durante un intervalo de 40 minutos. Las muestras fueron obtenidas en 4 intervalos de 10 minutos, modificando el periodo de envío de la PeH por parte del Concentrador con valores de 20, 10, 5 y 2 segundos.

En la Tabla B.1 del apéndice B, se muestra un extracto parcial de 254 muestras por razones de espacio, considerándolas representativas del experimento dado que las condiciones de interés se repiten para el resto de las muestras.

Convergencia del tiempo en la UCs

Se compararon los sesgos correspondientes entre sí y el resultado arroja las siguientes conclusiones:

- Como era de esperar, al aumentar la frecuencia de PeH el valor de los sesgos disminuye linealmente. No se intentó un intervalo entre PeH menor a los 2 segundos porque se pierde eficiencia en el canal. Con un periodo de PeH de 5 seg. el resultado es satisfactorio y acorde a las expectativas, por lo que nos referiremos a él en adelante.
- Para un periodo de PeH de 5 seg., se observó que la situación de caso peor arroja que las tres UCs diferían entre sí a lo sumo en **tres** unidades de sesgo (equivalente a 0,1 ms), como se presenta en las muestras 123 a 125 de la Tabla A.1 (que por comodidad se extracta en la Tabla 9.1). Estas muestras son correspondientes entre sí (pertenecen al mismo segundo IdSync = B4h). Las muestras 135 a 137 que se extractan, también correspondientes entre sí (pertenecen al mismo segundo IdSync = C8h) y son particularmente interesantes porque representan la **moda** del conjunto total de 1000 muestras con periodo de PeH de 5 segundos.

Las columnas de las Tablas tienen el siguiente significado:

 # : Numero secuencial de la muestra

UC: Dirección de la UC.

IdSync: Segundo en que fue impuesta la PeH. Sirve para relacionar los sesgos correspondientes entre sí.

Trama del mensaje: Mensaje de Estado, recibido por la Terminal de Prueba desde cada UC.

Sesgo (Hex): Valor Hexadecimal (complemento a 2) del sesgo tomado en la PeH otorgado en unidades de los timers tanto del Concentrador que impone la PeH como de la UC local.

Sesgo (Dec): Valor decimal equivalente del Sesgo.

Valor Extremo a Extremo: Diferencia entre los dos valores más lejanos del sesgo, para muestras correspondientes, resultantes de la misma peH.

#	UC	Id Sync	Trama del mensaje de respuesta de cada UC	Sesgo (Hex)	Sesgo (Dec)	Valor Extremo a Extremo
123	**04**	B4	04-00 09 02 B4 FF FF 03 0A 09 01 06 0C 00 24 00 E0 01	FFFF	-1	**3**
124	**02**	B4	02-00 09 02 B4 00 00 03 0A 00 01 06 08 00 2F 00 F4 01	0000	0	
125	**03**	B4	03-00 09 00 B4 FF FD 03 0A 01 01 06 08 00 27 00 E0 01	FFFD	-3	
135	**02**	C8	02-00 09 02 C8 FF FF 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFF	-1	**1**
136	**03**	C8	03-00 09 00 C8 FF FE 03 0A 09 01 06 0C 00 27 00 E4 01	FFFE	-2	
137	**04**	C8	04-00 09 02 C8 FF FF 03 0A 00 01 06 08 00 24 00 E4 01	FFFF	-1	

Tabla 9.1: Muestras de Mensajes con Contenido de Tiempos para Evaluación.

Cada unidad de contador equivale a 0,03 ms luego para el caso peor apuntado tenemos:

Sesgo máximo = 3 * 0.03 ms = 0.09 ms

Este valor extremo a extremo permite concluir que la convergencia de tiempo para las UCs es de aproximadamente + / - 0.05 ms. Algunos autores utilizan el término **precisión** para definir este valor de mérito. En este caso se utilizó el término convergencia de la hora con que las distintas UCs registran un mismo evento.

Si se toma el intervalo que media entre cada PeH de 5 s = 5000 ms y se lo relaciona con el factor máximo obtenido de extremo a extremo se tiene:

P = 0.1 ms / 5000ms = 20 x 10^-6 = 20 ppm

Lo cual es comparable con los valores esperados máximos de 30 ppm que asegura el fabricante respecto a los cristales.

Estampa de tiempo en eventos

La estampa de tiempo o TimeStamp es un componente deseable ante la ocurrencia de un evento.

A efectos de reforzar la apreciación ya observada en el apartado anterior respecto a la convergencia, al prototipo de diagnóstico se le incorporó una etapa electrónica que permite la emisión de eventos simultáneos a UCs interconectadas en red. En efecto, tomada la placa prototipo, se le incorporó una etapa cuyo esquema funcional se presenta en la Figura 9. 4.

En la figura se representan tres UCs de direcciones 2 a 4 y un concentrador, todos vinculados en red bajo norma RS-485. El Concentrador y las UCs poseen cristales de 32768 Hz para llevar la hora de su propio RTC. El Concentrador pondrá en hora a las UCs, cada 5 segundos, mediante un comando PeH emitido en broadcast.

Cada UC permite la entrada de un evento digital (por la parte izquierda del gráfico), mediante la operación de la llave SW respectiva. En cualquiera de las UCs el evento ingresa por el bit 7 del puerto B, de dirección relativa 04h.

La operación manual de las tres llaves SW2 a SW4 - en forma simultánea, por única vez, pulsando el botón y liberándolo -, provoca el registro de dos eventos en cada una de las UC, como es sabido, tales eventos se almacenan automáticamente en su pila.

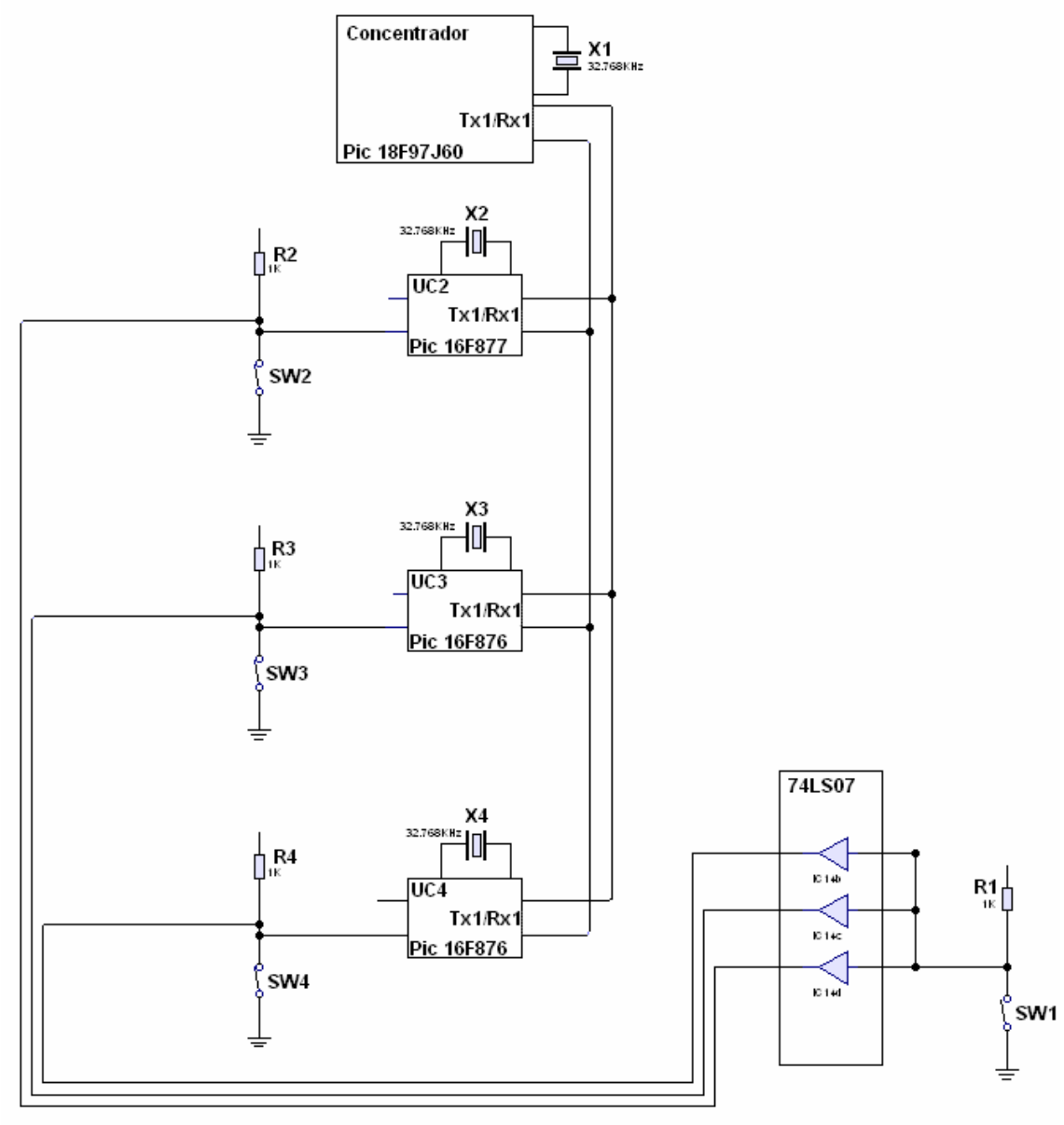


Figura 9.4: Esquema del ingreso de eventos en las UCs de la Red 1.

Cuando efectuamos el requerimiento de tales eventos mediante la terminal de diagnóstico y observamos la hora de ocurrencia de cada uno de los eventos registrados, tenemos una situación como la que se aprecia en la siguiente gráfica, extraída de la terminal de diagnóstico:

Tipo	UC	Puerto	Bit	Estado	TimeStamp
0	02	4	7	0	2010-03-20 11:55:53.7**267**
0	03	4	7	0	2010-03-20 11:55:53.7**398**
0	04	4	7	0	2010-03-20 11:55:53.7**588**
0	04	4	7	1	2010-03-20 11:55:56.1**157**
0	02	4	7	1	2010-03-20 11:55:56.1**306**
0	03	4	7	1	2010-03-20 11:55:56.1**261**

Figura 9.5: listado de eventos producidos por pulsación manual de llaves.

En la figura destacamos las siguientes variables según protocolo:

- El Tipo de evento 0, indicativo de un evento de campo.
- La Dirección de cada UC involucrada con el orden en que les llega la consulta por parte de la terminal (ordenamiento cíclico del round robin).
- El puerto de entrada del evento en cada UC: 04h, relativo debido a la tabla de indirección que relaciona una dirección lógica a una física.
- El Bit (físico) del Puerto.
- El Estado al que va el bit provocado por el evento. Al pulsar se pone el bit a potencial 0 (0 lógico). Al liberar se lo pone a 5 volts (1 lógico).
- El Tiempo de ocurrencia del evento o Time Stamp. El experimento fue realizado el 20 de marzo de 2010 entre las 11:55:53 y 11:55:56 hs. (en tres segundos).

Obsérvese que debido a mecanismo de pulsación manual de las llaves, el Tiempo de ocurrencia de cada evento no puede tener simultaneidad. Difícilmente se podrá hacer converger la pulsación manual, con diferencias inferiores a una centésima de segundo. Con esta mecánica que posee diferencia de pulsación que rondan los centisegundos no podemos apreciar la convergencia que ronda en las décimas de milisegundo.

Por ello, a efectos de provocar una pulsación simultánea para las tres UCS, el circuito de la placa experimental fue modificado con el módulo que se presenta en la parte inferior de la Figura 9.4, que consta de un circuito integrado 74LS07, que reúne al menos tres compuertas buffer con característica tri-state u open collector. Las salidas de las compuertas del integrado son aplicadas en paralelo con los SWITCHs con la finalidad de no entorpecer su funcionamiento individual, pero si provocar un evento simultáneo, mediante la operación de una única llave SW1 como se presenta en la figura.

Con este arreglo se asegura que el ingreso de cada evento diferirá en alrededor de 1 nanosegundo, que es el tiempo de propagación especificado por el fabricante para cada compuerta. A los efectos de la granularidad que disponemos para

discriminación del tiempo de eventos, el ingreso de eventos con esta nueva modalidad podrá considerarse **simultáneo**.

En efecto, se efectuó la operación un tiempo más tarde, arrojándonos los valores que se presentan en la siguiente figura:

Tipo	UC	Puerto	Bit	Estado	TimeStamp
0	02	4	7	0	2010-03-20 12:27:41.69**40**
0	03	4	7	0	2010-03-20 12:27:41.69**40**
0	04	4	7	0	2010-03-20 12:27:41.69**41**
0	02	4	7	1	2010-03-20 12:27:50.36**45**
0	03	4	7	1	2010-03-20 12:27:50.36**46**
0	04	4	7	1	2010-03-20 12:27:50.36**45**

Figura 9.6: listado de eventos producidos por pulsación simultánea de llaves.

En la nueva situación observamos que los eventos simultáneos poseen un Time Stamp muy convergente ya que no difiere en más de 1/10 de milisegundo. Esta diferencia está motivada por las derivas del reloj de cada UC pero se condice con lo consignado en la Tabla 9.1, donde el sesgo entre UCs era del orden de 0.09 ms.

Cálculo del tiempo de transmisión del mensaje

Como lo define la ecuación (1) que repetimos más abajo por comodidad, la hora impuesta a cada UC por el Patrón de Hora, depende del tiempo de transmisión del mensaje.

$$\Delta t = N \times 10 \times 1/Vt \qquad (1)$$

$$h_{pdo} = h_o + \Delta t \qquad (2)$$

donde:

Δt = tiempo de transmisión del mensaje en segundos.
N = Cantidad de bytes del paquete de PeH.
Vt = Velocidad de transmisión en bps, del canal RS485.
h_o = Hora inicial (hora enviada en la PeH).
h_{pdo} = Hora a imponer al destino con la PeH.

La determinación teórica de **Δt** está relacionada con la velocidad de transmisión del dispositivo que impone la hora (el Concentrador) y requiere que no existan tiempos adicionales inter-bytes, en el modo consignado en la Figura 9.2. b).

La velocidad de transmisión asignada al Concentrador en función de su cristal de 25 MHZ es de **9586 bps**, ligeramente inferior a la nominal de 9600 bps, debido a los divisores enteros de 8 bits, que generan la velocidad de la UART a partir del cristal principal del microcontrolador. Existen microcontroladores que hacen esta división

con divisores enteros en 16 bits y que otorgan mayor precisión a la velocidad de transmisión.

En las UCs, debido a que su cristal principal es de 20 MHZ la velocidad de transmisión más cercana a la nominal se sitúa en **9615 bps** por la misma razón apuntada para el Concentrador.

Esta ligera diferencia de velocidades entre emisor y receptor no presenta un problema importante, dado que la velocidad se resincroniza en cada Start bit de los bytes del mensaje emitido. Por ello la velocidad real a la que el mensaje se transmite queda determinada por el emisor. En realidad existe un pequeño sesgo en el último byte del mensaje donde las UCs deciden (en función de su velocidad de transmisión más alta) que el Stop bit termina antes, pero esto es en una proporción de 9615/9586 = 1,003. Este efecto además se debiera dividir por 15 que es la totalidad de los bytes del mensaje, ya que el fenómeno de adelanto sólo se produce en el último byte. Este error es imperceptible y puede afirmarse que el retardo de llegada del mensaje de PeH (15 bytes) emitido por el Concentrador es de:

$$\Delta t = N \times 10 \times 1/Vt = 15 * 10 * 1 / 9586 = 150 * 0{,}104 \text{ ms} = 15{,}64 \text{ ms} \qquad (3)$$

En esta fórmula se supone que el tiempo de proceso del Concentrador desde que obtiene la hora considerada patrón hasta la emisión del primer byte, es nulo (en proporción), lo mismo que es nulo el tiempo de proceso en cada UC desde el momento que ha recibido el mensaje y hasta el instante de imponer la hora a su propio RTC local. O sea, la aproximación es considerar que el retardo **Δt** se debe exclusivamente a tiempo de transmisión.

El valor de **Δt** calculado, es el que cada UC debe adicionar al valor de tiempo del mensaje de PeH previo a imponer a su RTC local.

Verificación experimental

En el Concentrador se instrumentó una rutina para permitir la medición del tiempo de transmisión del mensaje de PeH. El mecanismo es el siguiente:

1. Se lee el valor del timer en el instante que se produce la primera interrupción debido a la emisión del primer byte del mensaje PeH.
2. Se lee el valor del timer en el instante que se produce la última interrupción. Esto ocurre en el momento que el byte nro 14 ha sido transferido ya que el byte Nro 15 no se emite por interrupción de UART pues ya está encolado y existe la interrupción de un timer accesorio según se detallara en el apartado “Conformación del Driver de manejo de la UART” del capítulo 5.
3. Inmediatamente, se presenta en el display del Concentrador, la diferencia entre los dos valores de timer registrados, valor que se refresca en cada emisión de la PeH.

A continuación, se presenta un listado de 20 valores consecutivos que se han tomado como lectura experimental del Display, según la modalidad descripta en los párrafos anteriores, refrescándose estos cada 5 segundos (periodo de la PeH):

480, 481, 483, 482, 482, 481, 483, 482, **484**, 483, 480, 482, 482, 483, 481, 480, 483, 482, 481, 482.

Estos valores son representativos del tiempo que se tarda para enviar 14 bytes, medidos en valores del contador TMR1, controlado por el Timer1 del microcontrolador. Los valores extremos que arrojan las mediciones oscilan entre **480 y 484**.

Tomando los valores extremos de la medición, podemos calcular los tiempos medidos (para 14 bytes) y extrapolarlos para el largo del mensaje de PeH (15 bytes).

$$\Delta t\ 1 = 480 * 32768 / 1000 * (15 / 14) = 15{,}69\ ms \qquad (4)$$

$$\Delta t\ 2 = 484 * 32768 / 1000 * (15 / 14) = 15{,}82\ ms \qquad (5)$$

$$Variación = (\ 15{,}82\ ms - 15{,}69\ ms) = 0{,}13\ ms \qquad (6)$$

La primera observación es que el valor tomado experimentalmente y que resulta de (4), es ligeramente superior (en 0,05 ms) que el calculado con la velocidad de transmisión derivada del cristal de 25 MHZ, según indica la fórmula (3).

Luego, tomando la diferencia entre los valores dados por (4) y (5) según se indica en (6), observamos que ya sea por variaciones en el tiempo de proceso o por el propio cristal del Concentrador, el tiempo de transmisión del mensaje presupone alrededor de 0,2 ms extremo a extremo que deben sumarse como error de hora en la sincronización interna para cada unidad, respecto a la hora patrón supuesta para el Concentrador.

Luego, teniendo en cuenta las dos fuentes de error a saber:

a) Error de convergencia entre las UCs debido a la deriva de los cristales de sus RTC locales (del orden de 0,1 ms) y,

b) Error provocado por la variación en la duración del mensaje de PeH (del orden de 0,2 ms).

Debido a ello, se estima que la exactitud de sincronización interna es del orden de 0,3 ms extremo a extremo o de + / - 0,15 ms.

Conclusiones

Se ha llegado a una definición e implementación definida de sincronización de microcontroladores de baja gama, interconectados por una red RS-485. El valor de exactitud de Puesta en Hora mediante broadcast es altamente satisfactorio. El sesgo en cualquiera de las unidades respecto al dispositivo que oficia de patrón otorga valores comparables a la propia exactitud del patrón (20 ppm). Este resultado permite a las unidades de una subred de microcontroladores sincronizados mediante esta técnica, elaborar un registro de eventos consistente en relación causa–efecto, aún cuando estos eventos son registrados por unidades diferentes.

Para niveles superiores de exactitud puede trabajarse sobre la búsqueda de cristales de mayor calidad y consecuentemente mayor exactitud o para aplicaciones más exigentes, colocar un dispositivo patrón de hora de elevada exactitud (GPS) **en cada UC**.

Como trabajo futuro – ya que el mismo superaría por razones de tiempo y extensión, el alcance de esta tesis -, quedarían dos aspectos a evaluar:

a) Sincronizar varias subredes con sendos concentradores y comparar los registros de eventos provocados en distintas UCs y de distintas subredes en base a una técnica como la esbozada en la Figura 9.4.

b) Efectuar la sincronización con un patrón externo que provea el UTC, logrando así la sincronización externa de una subred y, de hecho, todas las UCs del sistema distribuido completo.

Siguiendo en esta línea, la sincronización lograda para las distintas subredes permitiría relacionar eventos ocurridos en cualquier punto del sistema global con una exactitud a determinar.

CAPÍTULO 10: CONCLUSIONES / LÍNEAS FUTURAS DE INVESTIGACIÓN

RESUMEN

En el presente capítulo se efectúa una ponderación del trabajo efectuado y se otorgan las conclusiones que a entender de este tesista merece tal ponderación.

Dado que en todo trabajo prolongado surgen nuevas líneas que podrían haber merecido ser seguidas y que no fueron abordadas por razones de su complejidad o por desviar el objetivo central de esta tesis, las mismas se consignan como posibles a seguir en emprendimientos futuros.

10.1. OBSERVACIÓN DE RESULTADOS Y CONCLUSIONES

Las pruebas otorgaron resultados altamente satisfactorios en opinión de este tesista. El proyecto concluye con la concreción de una gran parte de los objetivos previstos entre los cuales se citan:

- **Concreción del desarrollo de un prototipo Hardware – Software que sirvió con fines de evaluación esperados**: Su desarrollo la utilización de la gran mayoría del tiempo de esta tesis, especialmente las dedicadas a las pruebas y puesta a punto de funcionamiento.
- **Se desarrolló un protocolo de comunicaciones en tres capas que se utilizó en el proyecto y que será útil para acciones futuras**: El mismo fue utilizado y mejorado en sucesivas versiones y en emprendimientos contemporáneos con el desarrollo de esta tesis.
- **Se desarrolló un mecanismo de sincronización de unidades distribuidas mediante técnica de broadcast, que fue probado en una red multipunto:** El mismo abre un panorama propio de investigación.
- **Se adaptó un software para un concentrador de comunicaciones lo que permitió la utilización de Ethernet y TPC/IP para la comunicación entre microcontroladores:** Este concepto le otorga una mayor probabilidad de inserción a los sistemas embebidos que surjan debido a la normalización que implican los desarrollos en torno a ethernet e Internet.
- **Se completó una aplicación de verificación y diagnóstico de fallas**.

Merece especial atención el logro de sincronización interna con precisión del orden de décimas de milisegundo. Esto se logró utilizando un medio de comunicaciones con una velocidad relativamente baja (9600 bps), quedando aún un margen holgado para mejorar resultados. Podría existir algún aporte trabajando con cristales de precisión certificada por el fabricante ya que los utilizados fueron genéricos y de bajo costo.

La sincronización externa con el agregado de GPS para múltiples concentradores y con varias subredes de microcontroladores estaría resuelta quedando, la evaluación de sus factores de mérito la que no fue efectuada por razones de extensión.

Quedaron por ser resueltas y verificadas algunas condiciones del protocolo Multidrop, especialmente aquellas que implican exigencias y mayor eficiencia al canal RS-485. En efecto, hubo situaciones especiales de exigencia (Scan por debajo de 50 ms) que arrojaron desconexiones y hasta bloqueo de las comunicaciones. Las mismas no fueron investigadas por razones de tiempo.

Respecto a indicadores que surgieron de las pruebas, algunos de ellos surgieron de la verificación directa de resultados:

- ✓ La red multipunto trabajando a una velocidad de 9600 bps (es posible su aumento hasta 57600 bps y aún posiblemente a 115200 bps) arrojó un error de sincronismo entre unidades del orden de 0.3 décimas de milisegundo.
- ✓ Si bien no se efectuaron pruebas con largas extensiones de cable para la interconexión de UCs, se efectuó una evaluación experimental de 4000 consultas consecutivas, con periodo de Scan de 50 ms (20 consultas por segundo) y no se observó ninguna pérdida de trama ni requerimiento

repetitivo (Retry) sobre ninguna de las 4 UCs que se utilizaron en el escenario de prueba.

- ✓ La red Ethernet en 10MBPS, posee una tasa efectiva inferior a 2MBPS debido a que usa un canal sincrónico (SSP) para comunicación entre el microcontrolador y el módulo embebido. Sin embargo esta velocidad es altamente suficiente para la mayoría de los casos y la red misma demostró un desempeño muy adecuado y persistencia en la conexión, lo que permite avizorar que su uso puede ser extendido a las UCs. En ambientes con mucho ruido se podrían efectuar conexiones mediante fibra óptica, con conversores normalizados sin prácticamente cambios en la aplicación prototipo.

Respecto al funcionamiento del software de las UCs se determinó empíricamente (mediante simulación) que, con cristales de 20 MHZ:

- ✓ La rutina principal en situación típica se resuelve en 250us. Esto es, el bucle principal programa principal se cursa 4000 veces por segundo.
- ✓ En condiciones de toma de estado el tiempo de ejecución de la mencionada rutina se eleva a 350 us.
- ✓ Las rutinas de interrupciones se cursan en tiempos inferiores a los 50 us.

10.2. LÍNEAS FUTURAS DE INVESTIGACIÓN

Vale apuntar que en opinión de este tesista, ha sido interesante el trabajo llevado a cabo y que el mismo no está cerrado sino que abre otras posibilidades de desarrollo relacionadas. El trabajo con microcontroladores puede ser extendido a aplicaciones con mayores requerimientos de procesamiento, como el procesamiento de señales involucrado en muchas aplicaciones de control, por ejemplo.

Para estas aplicaciones es necesaria la utilización de controladores más potentes y posiblemente con mejores interfaces de E/S, como los DSP (Digital Signal Processor). En este sentido se han establecido tanto las bases técnicas, como la experiencia necesaria como para que la utilización de DSC (Digital Signal Controllers, como los dsPICxxx), no signifique más que la adquisición del hardware y un entrenamiento mínimo a partir de lo que ya se ha aprendido y experimentado.

En este sentido y como una línea adicional de investigación relacionada con esta tesis, pero no incluida en ella, se estima una sub-red de procesadores digitales de señales (DSP), que conteniendo varias unidades de estos dispositivos (cada una abocada a una medición en particular), se vinculen a una UC que las concentre y recabe sus valores de medición. Esta concepción, evita la concentración de cables en una única UC, lográndose un procesamiento más distribuido, de más bajo costo y mayor facilidad de instalación.

Otra de las líneas de investigación que bien podría estar asociada a la utilización de DSP, es la de Sincronización externa de varias subredes de microcontroladores.

REFERENCIAS

1. F. M. Cady, Microcontrollers and Microcomputers: Principles of Software and Hardware Engineering, Oxford University Press, 1997, ISBN: 0195110080.
2. S. Tanenbaum, Computer Networks, 4th Ed, Prentice Hall Ptr, ISBN 0130661023, 2002.
3. Kurose J., Ross K., Redes de Computadores: Un enfoque descendente basado en Internet, Pearson Addison Wesley, 2003, ISBN 84782900613
4. Stevens W. R., Fenner B., Rudoff A. M., Stevens R. W., Unix Network Programming, Vol. 1: The Sockets Networking API, Addison-Wesley Professional, 3rd Edition, ISBN 0131411551, 2003.
5. Institute of Electrical and Electronics Engineers, Local Area Network-CSMA/CD Access Method and Physical Layer Spec. ANSI/IEEE 802.3-IEEE Computer Society, 1985.
6. Microhip Technology Inc., PICmicro™ Mid-Range MCU Family Reference Manual, 1997. Disponible en http://ww1.microchip.com/downloads/en/DeviceDoc/33023a.pdf
7. Microhip Technology Inc., Tutorial USART - Using in Asynchronous Mode, 2001. Disponible en http://ww1.microchip.com/downloads/en/DeviceDoc/usart.pdf.
8. Telecommunications Industry Association, “Application Guidelines for TIA/EIA-485-A”, TIA/EIA Telecommunications Systems Bulletin, 1998
9. Texas Instruments, SN65176B, SN75176B Differential Bus Transceivers, 2003. Disponible en http://focus.ti.com/docs/prod/folders/print/sn75176b.htm
10. Adam Dunkels, uIP, Networked Embedded Systems group, Swedish Institute of Computer Science, disponible en http://www.sics.se/~adam/uip/index.php/Main_Page.
11. Stallings William, Data and Computer communications. 6th. Edition. 2000. Prentice Hall. ISBN 013-084370-9.
12. Stevens Richard, TCP Illustrated – The protocols. VOL1. Addison-Wesley, 1994, ISBN 0-201-63346-9.
13. Angulo J. - García B. - Angulo I. – Vicente J., Microcontroladores Avanzados, Thomson – 2005, ISBN 84-9732-385-8.
14. Microhip Technology Inc., Ethernet Theory of Operation - AN1120, 2008. Disponible en http://ww.microchip.com
15. 6. A. S. Tanenbaum, M. van Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, 2nd Edition, ISBN 0132392275, 2006.

AGRADECIMIENTOS

A mi esposa María Alicia y mis hijos: Belén, Matías y Julieta, que conocen que el emprendimiento de este Magíster, fue tomado como un desafío luego de un acontecimiento más, que marcó nuestras vidas.

A Fernando Tinetti mi tutor, que ha sido mi mentor y espejo no sólo en esta tesis, sino en varios de mis emprendimientos en estos últimos años.

A la Universidad de La Plata donde me recibí, la Facultad de Ingeniería y de Informática y por extensión a la Universidad Argentina, que nos permite estudiar en forma gratuita para realizarnos como personas.

A Marcelo Gómez y otros compañeros de la facultad, que me brindaron su aporte técnico en cuestiones específicas, conformando un buen equipo en actividades que nos divierten, como son la docencia y la investigación.

A Gloria Bianchi y Marta Sáenz López que en una cena casual, me indujeron a emprender esto.

Por último a Transpa, que me da todo el apoyo y tiempo para que pueda estudiar y desarrollar lo aprendido, en un marco profesional y contenedor.

Ricardo A. López

21/05/10

APÉNDICE A

En el diagrama de la Figura A.1, se otorga una visión complementaria al diagrama de flujo ya expuesto en el capítulo correspondiente, donde se muestra una dependencia de las rutinas del programa principal con una descripción del rol que cumplen. En el diagrama, se han omitido expresamente las llamadas reiteradas a una misma rutina cuando esto ocurre.

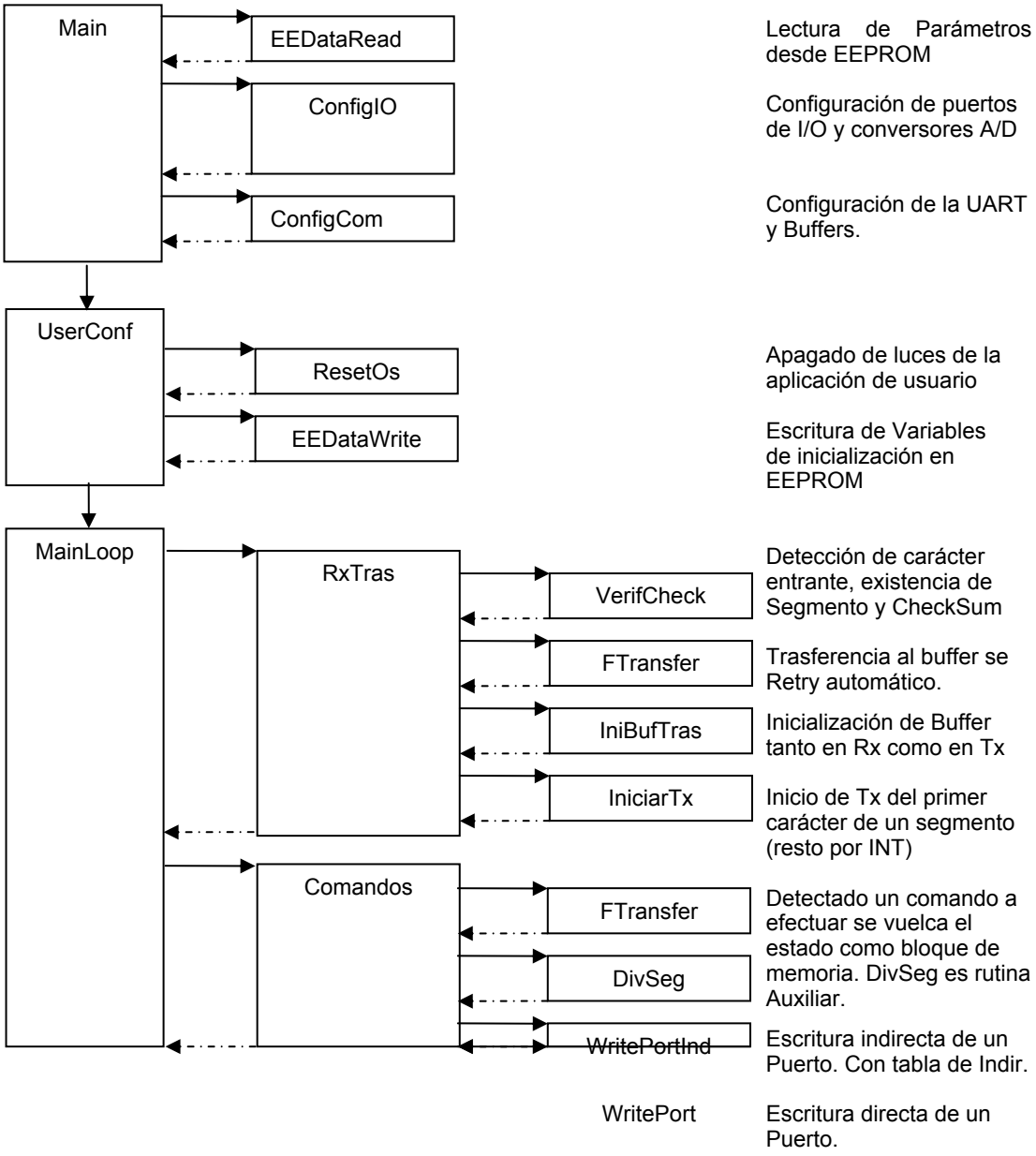


Figura A.1: Gráfico de dependencia de rutinas del programa principal.

En el grafico de la Figura A.2, se otorga una visión similar a la anterior pero enteramente para la aplicación de usuario que se sitúa seguidamente al pie de la rutina **MainLoop** del gráfico anterior.

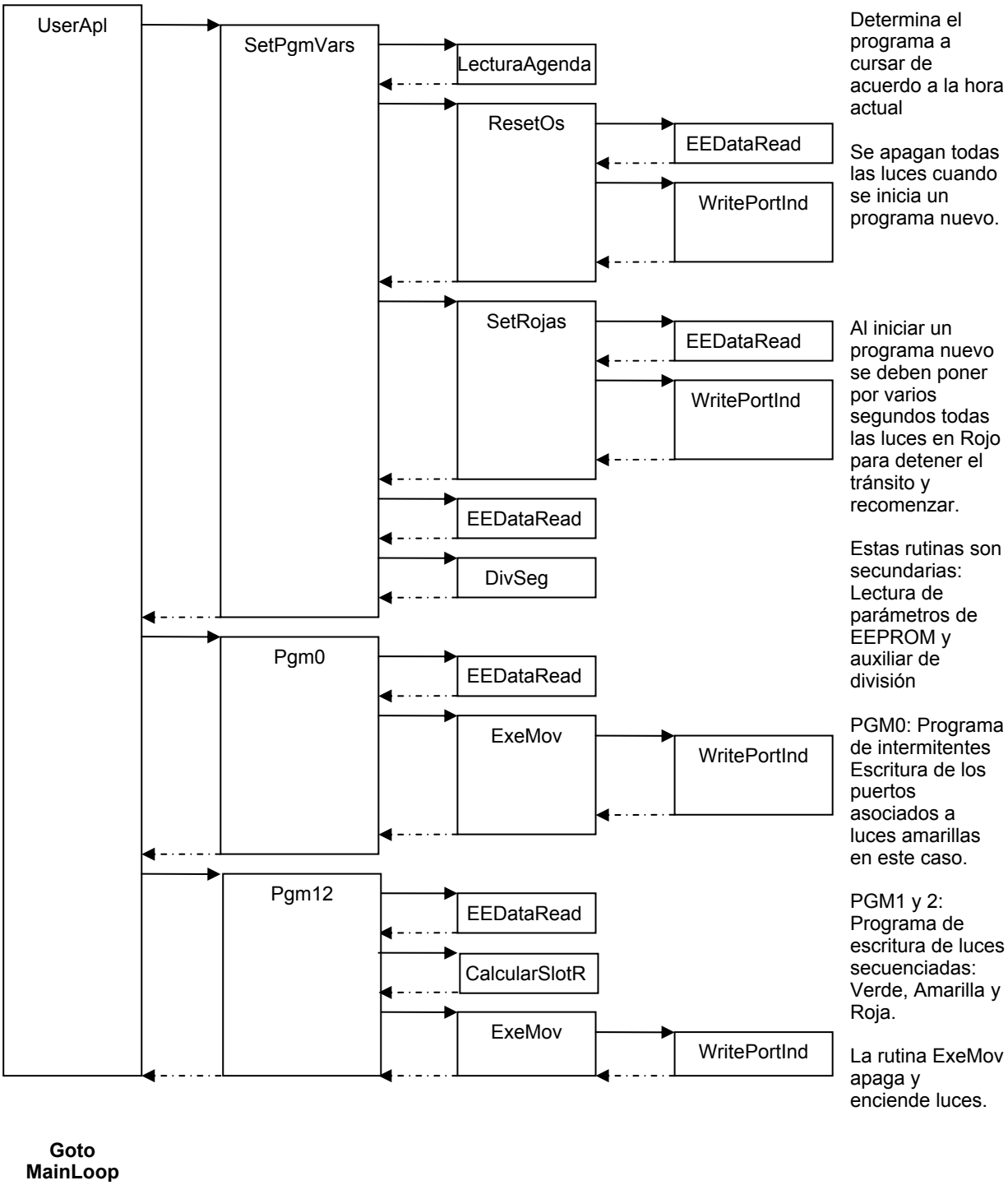


Figura A.2: Gráfico de dependencia de rutinas de la aplicación de usuario.

Se otorga alguna descripción de más detalle en relación con la aplicación de usuario desarrollada:

El programa 0 (Pgm0) es el de luces amarillas intermitentes. En él se entra en cualquier condición de contingencia de tránsito. El programa 1 y 2 (Pgm1 y Pgm2) son programas con distintas temporizaciones de luces: Por ejemplo el Pgm1 puede ser un programa diurno con intervalos de verde del orden de 40 segundos, típico en semáforos, para el control de un tránsito de intensidad media y alta. El Pgm2 podría estar fijado con tiempos más cortos (15 ó 20 segundos), para tránsito de intensidad baja, en horas de la noche.

El sistema en cada frontera de 10 minutos verifica la agenda y decide cuál es el programa a cursar. Esto permite que por ejemplo, se curse el Pgm1 entre las 8 hs y las 22 hs, luego se pase a Pgm2 entre las 22 hs y las 24 hs y luego se pase a Pgm0 (amarillo intermitente) entre las 24 hs y las 8 hs del día siguiente. Tal agenda puede ser repetida cada día de la semana.

Como puede observarse en el diagrama de dependencia anterior, las rutinas de ejecución de cualquiera de los programas está en secuencia, pero su ejecución está condicionada por la agenda.

Por su parte, la rutina **ExeMov** es de llamado recurrente cada vez que se desea encender una luz. Para ello, la dirección del puerto correspondiente es un parámetro en la llamada. Ahora bien, cada vez que se enciende una luz en el sistema, previamente se verifica el apagado de su predecesora en secuencia. Esto se hace a efectos de evitar que dos luces antagónicas (verde – roja) se enciendan simultáneamente, pudiendo provocar un accidente. En modo similar cada vez que se enciende una luz, a los pocos milisegundos se verifica si existe corriente de encendido circulante y en base a ello se determina si existe lámpara quemada. Esto se efectúa mediante un hardware externo, cuya salida se lee desde uno de los bits entrantes a la UC.

Cualquiera de las dos situaciones: Encendido simultáneo de luces antagónicas o lámpara quemada, llevan el semáforo a situación de alerta y pasaje automático a Pgm0 (Amarillo intermitente). Además, se provoca el registro local de un evento, que luego será rescatado en la Toma de Estado habitual que ejerce el Centro de Control.

En el grafico de la Figura A.3, se otorga una visión de la dependencia de rutinas de interrupciones.

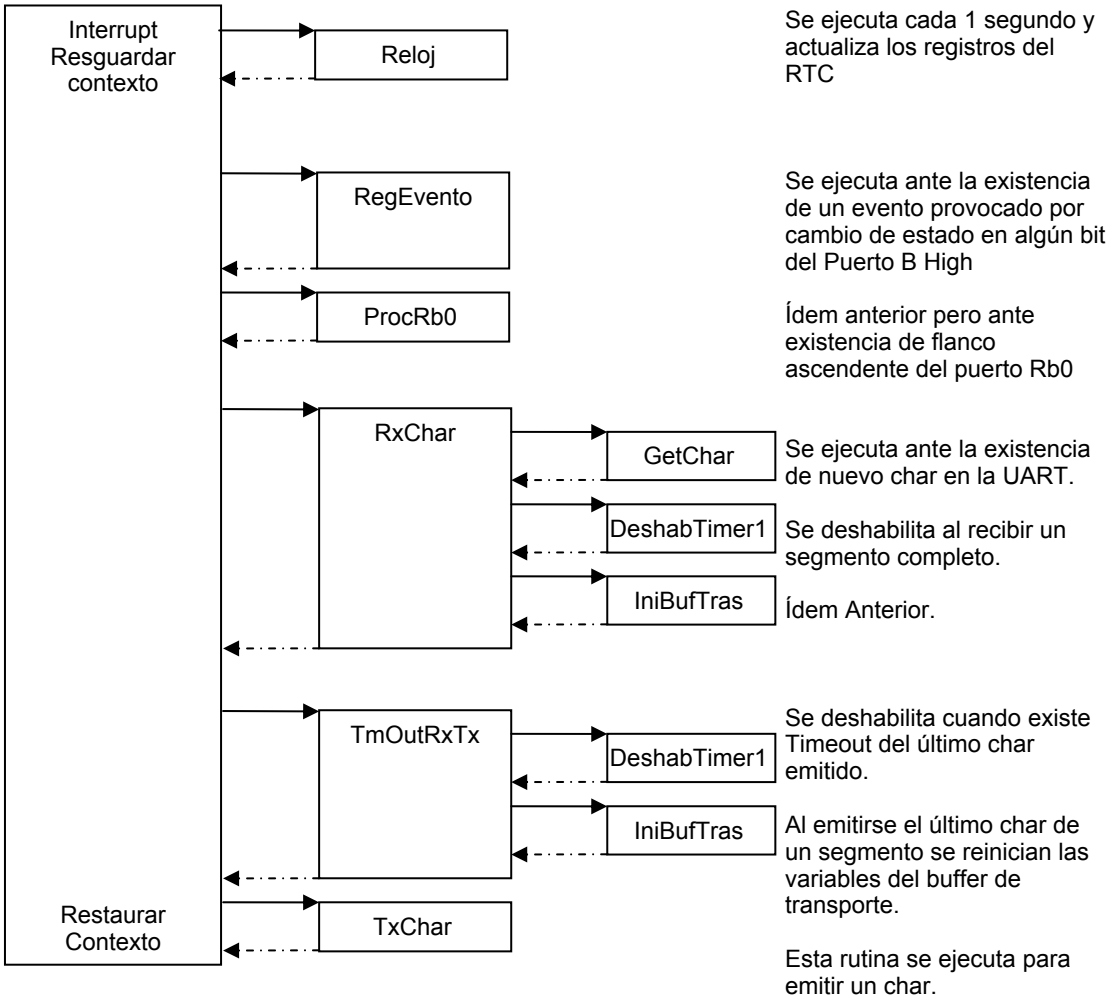


Figura A.3: Gráfico de dependencia de rutinas de la aplicación de usuario.

APÉNDICE B

La siguiente tabla otorga una cantidad mayor de valores del experimento efectuado para la sincronización, utilizando distintos periodos de emisión de la PeH por parte del Concentrador. En la columna del extremo derecho, se destacan en gris los sesgos máximos obtenidos en cada caso (en unidades de contador, equivalente a 0.03 ms).

Tabla A.1: Muestras tomadas con periodo de PeH variable.

#	UC	Id Sync	Trama del mensaje de respuesta de cada UC	Sesgo (Hex)	Sesgo (Dec)	Valor Extremo a Extremo
			Periodo de PeH = 20 seg.			**7**
1	04	31	04-00 09 02 31 FF F2 03 0A 09 01 06 0C 00 24 00 E0 01	FFF2	-14	6
2	02	31	02-00 09 02 31 FF F4 03 0A 02 01 06 08 00 2F 00 F0 01	FFF4	-12	
3	03	31	03-00 09 00 31 FF EE 03 0A 05 01 06 0C 00 07 00 E0 01	FFEE	-18	
4	04	45	04-00 09 02 45 FF EE 03 0A 07 01 06 0C 00 24 00 E4 01	FFEE	-18	7
5	02	45	02-00 09 02 45 FF F2 03 0A 00 01 06 08 00 2F 00 F4 01	FFF2	-14	
6	03	45	03-00 09 00 45 FF EB 03 0A 03 01 06 0C 00 07 00 E4 01	FFEB	-21	
7	02	59	02-00 09 02 59 FF F5 03 0A 09 01 06 0C 00 2F 00 F0 01	FFF5	-11	5
8	03	59	03-00 09 00 59 FF F0 03 0A 02 01 06 08 00 27 00 E4 01	FFF0	-16	
9	04	59	04-00 09 02 59 FF F3 03 0A 05 01 06 0C 00 04 00 E0 01	FFF3	-13	
10	02	6D	02-00 09 02 6D FF F1 03 0A 07 01 06 0C 00 2F 00 F4 01	FFF1	-15	6
11	03	6D	03-00 09 00 6D FF EB 03 0A 00 01 06 08 00 27 00 E4 01	FFEB	-21	
12	04	6D	04-00 09 02 6D FF EF 03 0A 03 01 06 0C 00 04 00 E4 01	FFEF	-17	
13	03	81	03-00 09 00 81 FF EC 03 0A 09 01 06 0C 00 27 00 E0 01	FFEC	-20	6
14	04	81	04-00 09 02 81 FF F0 03 0A 02 01 06 08 00 24 00 E4 01	FFF0	-16	
15	03	95	03-00 09 00 95 FF EA 03 0A 07 01 06 0C 00 27 00 E4 01	FFEA	-22	6
16	04	95	04-00 09 02 95 FF EF 03 0A 00 01 06 08 00 24 00 E4 01	FFEF	-17	
17	02	95	02-00 09 02 95 FF F0 03 0A 03 01 06 0C 00 2F 00 F4 01	FFF0	-16	
18	04	A9	04-00 09 02 A9 FF EF 03 0A 09 01 06 0C 00 24 00 E0 01	FFEF	-17	5
19	02	A9	02-00 09 02 A9 FF F0 03 0A 02 01 06 08 00 2F 00 F0 01	FFF0	-16	
20	03	A9	03-00 09 00 A9 FF EB 03 0A 05 01 06 0C 00 07 00 E4 01	FFEB	-21	
21	04	BD	04-00 09 02 BD FF F2 03 0A 07 01 06 0C 00 24 00 E4 01	FFF2	-14	6
22	02	BD	02-00 09 02 BD FF F4 03 0A 00 01 06 08 00 2F 00 F4 01	FFF4	-12	
23	03	BD	03-00 09 00 BD FF EE 03 0A 03 01 06 0C 00 07 00 E4 01	FFEE	-18	
24	02	D1	02-00 09 02 D1 FF F0 03 0A 09 01 06 0C 00 2F 00 F0 01	FFF0	-16	5
25	03	D1	03-00 09 00 D1 FF EB 03 0A 02 01 06 08 00 27 00 E0 01	FFEB	-21	
26	04	D1	04-00 09 02 D1 FF EF 03 0A 05 01 06 0C 00 04 00 E0 01	FFEF	-17	
27	02	E5	02-00 09 02 E5 FF ED 03 0A 07 01 06 0C 00 2F 00 F4 01	FFED	-19	6
28	03	E5	03-00 09 00 E5 FF E7 03 0A 00 01 06 08 00 27 00 E4 01	FFE7	-25	
29	04	E5	04-00 09 02 E5 FF EB 03 0A 03 01 06 0C 00 04 00 E4 01	FFEB	-21	
30	03	F9	03-00 09 00 F9 FF EE 03 0A 09 01 06 0C 00 27 00 E0 01	FFEE	-18	5
31	04	F9	04-00 09 02 F9 FF F2 03 0A 02 01 06 08 00 24 00 E0 01	FFF2	-14	

32	02	F9	02-00 09 02 F9 FF F3 03 0A 05 01 06 0C 00 2F 00 F0 01	FFF3	-13	
33	03	0D	03-00 09 00 0D FF EB 03 0A 07 01 06 0C 00 27 00 E4 01	FFEB	-21	5
34	04	0D	04-00 09 02 0D FF EF 03 0A 00 01 06 08 00 24 00 E4 01	FFEF	-17	
35	02	0D	02-00 09 02 0D FF F0 03 0A 03 01 06 0C 00 2F 00 F4 01	FFF0	-16	
36	04	11	04-00 09 02 11 FF F0 03 0A 09 01 06 0C 00 24 00 E0 01	FFF0	-16	6
37	02	11	02-00 09 02 11 FF F2 03 0A 02 01 06 08 00 2F 00 F0 01	FFF2	-14	
38	03	11	03-00 09 00 11 FF EC 03 0A 05 01 06 0C 00 07 00 E0 01	FFEC	-20	
39	04	25	04-00 09 02 25 FF EE 03 0A 07 01 06 0C 00 24 00 E4 01	FFEE	-18	6
40	02	25	02-00 09 02 25 FF F1 03 0A 00 01 06 08 00 2F 00 F4 01	FFF1	-15	
41	03	25	03-00 09 00 25 FF EB 03 0A 03 01 06 0C 00 07 00 E4 01	FFEB	-21	
42	02	39	02-00 09 02 39 FF F1 03 0A 09 01 06 0C 00 2F 00 F4 01	FFF1	-15	5
43	03	39	03-00 09 00 39 FF EC 03 0A 02 01 06 08 00 27 00 E0 01	FFEC	-20	
44	04	39	04-00 09 02 39 FF F0 03 0A 05 01 06 0C 00 04 00 E4 01	FFF0	-16	
45	02	4D	02-00 09 02 4D FF F3 03 0A 07 01 06 0C 00 2F 00 F4 01	FFF3	-13	5
46	03	4D	03-00 09 00 4D FF EE 03 0A 00 01 06 08 00 27 00 E4 01	FFEE	-18	
47	04	4D	04-00 09 02 4D FF F2 03 0A 04 01 06 0C 00 04 00 E0 01	FFF2	-14	
48	03	61	03-00 09 00 61 FF EC 03 0A 09 01 06 0C 00 27 00 E0 01	FFEC	-20	6
49	04	61	04-00 09 02 61 FF F0 03 0A 02 01 06 08 00 24 00 E4 01	FFF0	-16	
50	02	61	02-00 09 02 61 FF F2 03 0A 05 01 06 0C 00 2F 00 F4 01	FFF2	-14	
51	03	75	03-00 09 00 75 FF EA 03 0A 07 01 06 0C 00 27 00 E4 01	FFEA	-22	6
52	04	75	04-00 09 02 75 FF ED 03 0A 01 01 06 08 00 24 00 E4 01	FFED	-19	
53	02	75	02-00 09 02 75 FF F0 03 0A 04 01 06 0C 00 2F 00 F0 01	FFF0	-16	
54	04	89	04-00 09 02 89 FF F3 03 0A 09 01 06 0C 00 24 00 E0 01	FFF3	-13	7
55	02	89	02-00 09 02 89 FF F5 03 0A 02 01 06 08 00 2F 00 F4 01	FFF5	-11	
56	03	89	03-00 09 00 89 FF EE 03 0A 05 01 06 0C 00 07 00 E4 01	FFEE	-18	
57	04	9D	04-00 09 02 9D FF ED 03 0A 07 01 06 0C 00 24 00 E4 01	FFED	-19	5
58	02	9D	02-00 09 02 9D FF EF 03 0A 01 01 06 08 00 2F 00 F0 01	FFEF	-17	
59	03	9D	03-00 09 00 9D FF EA 03 0A 04 01 06 0C 00 07 00 E0 01	FFEA	-22	
60	02	B1	02-00 09 02 B1 FF F3 03 0A 09 01 06 0C 00 2F 00 F4 01	FFF3	-13	5
61	03	B1	03-00 09 00 B1 FF EE 03 0A 02 01 06 08 00 27 00 E4 01	FFEE	-18	
62	04	B1	04-00 09 02 B1 FF F1 03 0A 05 01 06 0C 00 04 00 E4 01	FFF1	-15	
			Periodo de PeH = 10 seg.			**4**
63	02	AF	02-00 09 02 AF FF FA 03 0A 09 01 06 0C 00 2F 00 F4 01	FFFA	-6	1
64	03	AF	03-00 09 00 AF FF F9 03 0A 01 01 06 08 00 27 00 E4 01	FFF9	-7	
65	04	AF	04-00 09 02 AF FF FA 03 0A 03 01 06 0C 00 04 00 E4 01	FFFA	-6	
66	04	B9	04-00 09 02 B9 FF F5 03 0A 07 01 06 0C 00 24 00 E4 01	FFF5	-11	4
67	02	B9	02-00 09 02 B9 FF F7 03 0A 09 01 06 0C 00 2F 00 F0 01	FFF7	-9	
68	03	B9	03-00 09 00 B9 FF F3 03 0A 01 01 06 08 00 27 00 E0 01	FFF3	-13	
69	02	C3	02-00 09 02 C3 FF FA 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFA	-6	3
70	03	C3	03-00 09 00 C3 FF F7 03 0A 00 01 06 08 00 27 00 E4 01	FFF7	-9	
71	04	C3	04-00 09 02 C3 FF F9 03 0A 02 01 06 08 00 24 00 E0 01	FFF9	-7	
72	02	CD	02-00 09 02 CD FF F7 03 0A 08 01 06 0C 00 2F 00 F0 01	FFF7	-9	4
73	03	CD	03-00 09 00 CD FF F3 03 0A 00 01 06 08 00 27 00 E0 01	FFF3	-13	
74	04	CD	04-00 09 02 CD FF F6 03 0A 01 01 06 08 00 24 00 E4 01	FFF6	-10	
75	02	D7	02-00 09 02 D7 FF F9 03 0A 07 01 06 0C 00 2F 00 F4 01	FFF9	-7	3
76	03	D7	03-00 09 00 D7 FF F6 03 0A 09 01 06 0C 00 27 00 E4 01	FFF6	-10	
77	04	D7	04-00 09 02 D7 FF F8 03 0A 01 01 06 08 00 24 00 E0 01	FFF8	-8	
78	03	E1	03-00 09 00 E1 FF F7 03 0A 09 01 06 0C 00 27 00 E4 01	FFF7	-9	3

79	04	E1	04-00 09 02 E1 FF F8 03 0A 00 01 06 08 00 24 00 E4 01	FFF8	-8	
80	02	E1	02-00 09 02 E1 FF FA 03 0A 02 01 06 08 00 2F 00 F0 01	FFFA	-6	
81	03	EB	03-00 09 00 EB FF F6 03 0A 08 01 06 0C 00 27 00 E4 01	FFF6	-10	3
82	04	EB	04-00 09 02 EB FF F8 03 0A 00 01 06 08 00 24 00 E0 01	FFF8	-8	
83	02	EB	02-00 09 02 EB FF F9 03 0A 01 01 06 08 00 2F 00 F4 01	FFF9	-7	
84	03	F5	03-00 09 00 F5 FF F4 03 0A 08 01 06 0C 00 27 00 E4 01	FFF4	-12	4
85	04	F5	04-00 09 02 F5 FF F7 03 0A 09 01 06 0C 00 24 00 E4 01	FFF7	-9	
86	02	F5	02-00 09 02 F5 FF F8 03 0A 01 01 06 08 00 2F 00 F0 01	FFF8	-8	
87	03	FF	03-00 09 00 FF FF F6 03 0A 07 01 06 0C 00 27 00 E4 01	FFF6	-10	3
88	04	FF	04-00 09 02 FF FF F8 03 0A 09 01 06 0C 00 24 00 E0 01	FFF8	-8	
89	02	FF	02-00 09 02 FF FF F9 03 0A 00 01 06 08 00 2F 00 F4 01	FFF9	-7	
90	04	09	04-00 09 02 09 FF F8 03 0A 08 01 06 0C 00 24 00 E4 01	FFF8	-8	3
91	02	09	02-00 09 02 09 FF FA 03 0A 00 01 06 08 00 2F 00 F0 01	FFFA	-6	
92	03	09	03-00 09 00 09 FF F7 03 0A 01 01 06 08 00 27 00 E4 01	FFF7	-9	
93	04	13	04-00 09 02 13 FF F5 03 0A 08 01 06 0C 00 24 00 E0 01	FFF5	-11	2
94	02	13	02-00 09 02 13 FF F5 03 0A 09 01 06 0C 00 2F 00 F4 01	FFF5	-11	
95	03	13	03-00 09 00 13 FF F3 03 0A 01 01 06 08 00 27 00 E4 01	FFF3	-13	
96	04	1D	04-00 09 02 1D FF F8 03 0A 07 01 06 0C 00 24 00 E4 01	FFF8	-8	3
97	02	1D	02-00 09 02 1D FF F9 03 0A 09 01 06 0C 00 2F 00 F0 01	FFF9	-7	
98	03	1D	03-00 09 00 1D FF F6 03 0A 00 01 06 08 00 27 00 E4 01	FFF6	-10	
99	02	27	02-00 09 02 27 FF FA 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFA	-6	4
100	03	27	03-00 09 00 27 FF F6 03 0A 00 01 06 08 00 27 00 E0 01	FFF6	-10	
101	04	27	04-00 09 02 27 FF F8 03 0A 01 01 06 08 00 24 00 E4 01	FFF8	-8	
			Periodo de PeH = 5 seg.			**3**
102	02	91	02-00 09 02 91 FF FC 03 0A 05 01 06 0C 00 2F 00 F0 01	FFFC	-4	1
103	03	91	03-00 09 00 91 FF FB 03 0A 06 01 06 0C 00 07 00 E0 01	FFFB	-5	
104	04	91	04-00 09 02 91 FF FB 03 0A 07 01 06 0C 00 24 00 E0 01	FFFB	-5	
105	03	96	03-00 09 00 96 FF F8 03 0A 09 01 06 0C 00 27 00 E4 01	FFF8	-8	2
106	04	96	04-00 09 02 96 FF F9 03 0A 09 01 06 0C 00 24 00 E4 01	FFF9	-7	
107	02	96	02-00 09 02 96 FF FA 03 0A 00 01 06 08 00 2F 00 F4 01	FFFA	-6	
108	03	9B	03-00 09 00 9B FF FB 03 0A 04 01 06 0C 00 07 00 E0 01	FFFB	-5	2
109	04	9B	04-00 09 02 9B FF FC 03 0A 05 01 06 0C 00 04 00 E0 01	FFFC	-4	
110	02	9B	02-00 09 02 9B FF FD 03 0A 06 01 06 0C 00 2F 00 F0 01	FFFD	-3	
111	02	A0	02-00 09 02 A0 FF FC 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFC	-4	1
112	03	A0	03-00 09 00 A0 FF FB 03 0A 09 01 06 0C 00 27 00 E4 01	FFFB	-5	
113	04	A0	04-00 09 02 A0 FF FC 03 0A 00 01 06 08 00 24 00 E4 01	FFFC	-4	
114	02	A5	02-00 09 02 A5 FF FC 03 0A 04 01 06 0C 00 2F 00 F0 01	FFFC	-4	1
115	03	A5	03-00 09 00 A5 FF FB 03 0A 05 01 06 0C 00 07 00 E0 01	FFFB	-5	
116	04	A5	04-00 09 02 A5 FF FB 03 0A 05 01 06 0C 00 04 00 E4 01	FFFB	-5	
117	04	AA	04-00 09 02 AA FF FD 03 0A 08 01 06 0C 00 24 00 E4 01	FFFD	-3	2
118	02	AA	02-00 09 02 AA FF FE 03 0A 09 01 06 0C 00 2F 00 F4 01	FFFE	-2	
119	03	AA	03-00 09 00 AA FF FC 03 0A 00 01 06 08 00 27 00 E0 01	FFFC	-4	
120	04	AF	04-00 09 02 AF FF F9 03 0A 03 01 06 0C 00 04 00 E4 01	FFF9	-7	2
121	02	AF	02-00 09 02 AF FF FA 03 0A 04 01 06 0C 00 2F 00 F4 01	FFFA	-6	
122	03	AF	03-00 09 00 AF FF F8 03 0A 05 01 06 0C 00 07 00 E4 01	FFF8	-8	
123	04	B4	04-00 09 02 B4 FF FF 03 0A 09 01 06 0C 00 24 00 E0 01	FFFF	-1	3
124	02	B4	02-00 09 02 B4 00 00 03 0A 00 01 06 08 00 2F 00 F4 01	0000	0	
125	03	B4	03-00 09 00 B4 FF FD 03 0A 01 01 06 08 00 27 00 E0 01	FFFD	-3	

126	03	B9	03-00 09 00 B9 FF F9 03 0A 03 01 06 0C 00 07 00 E4 01	FFF9	-7	2
127	04	B9	04-00 09 02 B9 FF F9 03 0A 04 01 06 0C 00 04 00 E4 01	FFF9	-7	
128	02	B9	02-00 09 02 B9 FF FB 03 0A 05 01 06 0C 00 2F 00 F4 01	FFFB	-5	
129	03	BE	03-00 09 00 BE FF FC 03 0A 09 01 06 0C 00 27 00 E0 01	FFFC	-4	1
130	04	BE	04-00 09 02 BE FF FD 03 0A 09 01 06 0C 00 24 00 E4 01	FFFD	-3	
131	02	BE	02-00 09 02 BE FF FD 03 0A 00 01 06 08 00 2F 00 F4 01	FFFD	-3	
132	02	C3	02-00 09 02 C3 FF F9 03 0A 03 01 06 0C 00 2F 00 F4 01	FFF9	-7	1
133	03	C3	03-00 09 00 C3 FF F8 03 0A 04 01 06 0C 00 07 00 E0 01	FFF8	-8	
134	02	C3	02-00 09 02 C3 FF F9 03 0A 06 01 06 0C 00 2F 00 F0 01	FFF9	-7	
135	02	C8	02-00 09 02 C8 FF FF 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFF	-1	1
136	03	C8	03-00 09 00 C8 FF FE 03 0A 09 01 06 0C 00 27 00 E4 01	FFFE	-2	
137	04	C8	04-00 09 02 C8 FF FF 03 0A 00 01 06 08 00 24 00 E4 01	FFFF	-1	
138	02	CD	02-00 09 02 CD FF FD 03 0A 04 01 06 0C 00 2F 00 F0 01	FFFD	-3	1
139	03	CD	03-00 09 00 CD FF FC 03 0A 05 01 06 0C 00 07 00 E0 01	FFFC	-4	
140	04	CD	04-00 09 02 CD FF FD 03 0A 06 01 06 0C 00 04 00 E4 01	FFFD	-3	
141	04	D2	04-00 09 02 D2 FF FC 03 0A 08 01 06 0C 00 24 00 E4 01	FFFC	-4	2
142	02	D2	02-00 09 02 D2 FF FC 03 0A 09 01 06 0C 00 2F 00 F4 01	FFFC	-4	
143	03	D2	03-00 09 00 D2 FF FA 03 0A 00 01 06 08 00 27 00 E4 01	FFFA	-6	
144	04	D7	04-00 09 02 D7 FF FB 03 0A 04 01 06 0C 00 04 00 E0 01	FFFB	-5	2
145	02	D7	02-00 09 02 D7 FF FC 03 0A 04 01 06 0C 00 2F 00 F4 01	FFFC	-4	
146	03	D7	03-00 09 00 D7 FF FA 03 0A 05 01 06 0C 00 07 00 E4 01	FFFA	-6	
147	03	DC	03-00 09 00 DC FF FD 03 0A 08 01 06 0C 00 27 00 E4 01	FFFD	-3	1
148	04	DC	04-00 09 02 DC FF FD 03 0A 09 01 06 0C 00 24 00 E0 01	FFFD	-3	
149	02	DC	02-00 09 02 DC FF FE 03 0A 00 01 06 08 00 2F 00 F0 01	FFFE	-2	
150	03	E1	03-00 09 00 E1 FF FA 03 0A 03 01 06 0C 00 07 00 E4 01	FFFA	-6	1
151	04	E1	04-00 09 02 E1 FF FA 03 0A 04 01 06 0C 00 04 00 E4 01	FFFA	-6	
152	02	E1	02-00 09 02 E1 FF FB 03 0A 05 01 06 0C 00 2F 00 F4 01	FFFB	-5	
153	03	E6	03-00 09 00 E6 FF FB 03 0A 09 01 06 0C 00 27 00 E0 01	FFFB	-5	2
154	04	E6	04-00 09 02 E6 FF FC 03 0A 00 01 06 08 00 24 00 E0 01	FFFC	-4	
155	02	E6	02-00 09 02 E6 FF FD 03 0A 01 01 06 08 00 2F 00 F0 01	FFFD	-3	
156	02	EB	02-00 09 02 EB FF FD 03 0A 03 01 06 0C 00 2F 00 F4 01	FFFD	-3	2
157	03	EB	03-00 09 00 EB FF FB 03 0A 04 01 06 0C 00 07 00 E4 01	FFFB	-5	
158	04	EB	04-00 09 02 EB FF FC 03 0A 05 01 06 0C 00 04 00 E0 01	FFFC	-4	
159	02	F0	02-00 09 02 F0 FF FB 03 0A 09 01 06 0C 00 2F 00 F0 01	FFFB	-5	1
160	03	F0	03-00 09 00 F0 FF FB 03 0A 09 01 06 0C 00 27 00 E4 01	FFFB	-5	
161	04	F0	04-00 09 02 F0 FF FA 03 0A 00 01 06 08 00 24 00 E4 01	FFFA	-6	
162	04	F5	04-00 09 02 F5 FF FD 03 0A 03 01 06 0C 00 04 00 E4 01	FFFD	-3	1
163	02	F5	02-00 09 02 F5 FF FD 03 0A 04 01 06 0C 00 2F 00 F0 01	FFFD	-3	
164	03	F5	03-00 09 00 F5 FF FC 03 0A 05 01 06 0C 00 07 00 E0 01	FFFC	-4	
165	04	FA	04-00 09 02 FA FF FA 03 0A 08 01 06 0C 00 24 00 E4 01	FFFA	-6	2
166	02	FA	02-00 09 02 FA FF FA 03 0A 09 01 06 0C 00 2F 00 F4 01	FFFA	-6	
167	03	FA	03-00 09 00 FA FF F8 03 0A 00 01 06 08 00 27 00 E4 01	FFF8	-8	
168	04	FF	04-00 09 02 FF FF FD 03 0A 04 01 06 0C 00 04 00 E4 01	FFFD	-3	1
169	02	FF	02-00 09 02 FF FF FD 03 0A 05 01 06 0C 00 2F 00 F0 01	FFFD	-3	
170	03	FF	03-00 09 00 FF FF FC 03 0A 06 01 06 0C 00 07 00 E0 01	FFFC	-4	
			Periodo de PeH = 2 seg.			**1**
171	02	89	02-00 09 02 89 FF FE 03 0A 07 01 06 0C 00 2F 00 F4 01	FFFE	-2	1
172	03	89	03-00 09 00 89 FF FD 03 0A 08 01 06 0C 00 27 00 E4 01	FFFD	-3	

173	04	89	04-00 09 02 89 FF FD 03 0A 08 01 06 0C 00 24 00 E4 01	FFFD	-3	
174	04	8B	04-00 09 02 8B FF FD 03 0A 09 01 06 0C 00 24 00 E4 01	FFFD	-3	0
175	02	8B	02-00 09 02 8B FF FD 03 0A 00 01 06 08 00 2F 00 F0 01	FFFD	-3	
176	03	8B	03-00 09 00 8B FF FD 03 0A 00 01 06 08 00 27 00 E4 01	FFFD	-3	
177	03	8D	03-00 09 00 8D FF FE 03 0A 01 01 06 08 00 27 00 E4 01	FFFE	-2	0
178	04	8D	04-00 09 02 8D FF FE 03 0A 02 01 06 08 00 24 00 E0 01	FFFE	-2	
179	02	8D	02-00 09 02 8D FF FE 03 0A 02 01 06 08 00 2F 00 F4 01	FFFE	-2	
180	02	8F	02-00 09 02 8F FF FE 03 0A 03 01 06 0C 00 2F 00 F4 01	FFFE	-2	0
181	03	8F	03-00 09 00 8F FF FE 03 0A 04 01 06 0C 00 07 00 E0 01	FFFE	-2	
182	04	8F	04-00 09 02 8F FF FE 03 0A 04 01 06 0C 00 04 00 E4 01	FFFE	-2	
183	04	91	04-00 09 02 91 FF FF 03 0A 05 01 06 0C 00 04 00 E4 01	FFFF	-1	1
184	02	91	02-00 09 02 91 FF FF 03 0A 06 01 06 0C 00 2F 00 F4 01	FFFF	-1	
185	03	91	03-00 09 00 91 FF FE 03 0A 06 01 06 0C 00 07 00 E4 01	FFFE	-2	
186	03	93	03-00 09 00 93 FF FC 03 0A 07 01 06 0C 00 27 00 E4 01	FFFC	-4	1
187	04	93	04-00 09 02 93 FF FD 03 0A 08 01 06 0C 00 24 00 E4 01	FFFD	-3	
188	02	93	02-00 09 02 93 FF FD 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFD	-3	
189	02	95	02-00 09 02 95 FF FE 03 0A 09 01 06 0C 00 2F 00 F4 01	FFFE	-2	1
190	03	95	03-00 09 00 95 FF FE 03 0A 00 01 06 08 00 27 00 E0 01	FFFE	-2	
191	04	95	04-00 09 02 95 FF FF 03 0A 00 01 06 08 00 24 00 E4 01	FFFF	-1	
192	04	97	04-00 09 02 97 FF FD 03 0A 01 01 06 08 00 24 00 E4 01	FFFD	-3	1
193	02	97	02-00 09 02 97 FF FC 03 0A 02 01 06 08 00 2F 00 F0 01	FFFC	-4	
194	03	97	03-00 09 00 97 FF FC 03 0A 02 01 06 08 00 27 00 E4 01	FFFC	-4	
195	03	99	03-00 09 00 99 FF FC 03 0A 03 01 06 0C 00 07 00 E4 01	FFFC	-4	1
196	04	99	04-00 09 02 99 FF FD 03 0A 04 01 06 0C 00 04 00 E0 01	FFFD	-3	
197	02	99	02-00 09 02 99 FF FD 03 0A 04 01 06 0C 00 2F 00 F4 01	FFFD	-3	
198	02	9B	02-00 09 02 9B FF FE 03 0A 05 01 06 0C 00 2F 00 F4 01	FFFE	-2	1
199	03	9B	03-00 09 00 9B FF FD 03 0A 06 01 06 0C 00 07 00 E0 01	FFFD	-3	
200	04	9B	04-00 09 02 9B FF FD 03 0A 06 01 06 0C 00 04 00 E4 01	FFFD	-3	
201	04	9D	04-00 09 02 9D FF FF 03 0A 07 01 06 0C 00 24 00 E4 01	FFFF	-1	1
202	02	9D	02-00 09 02 9D FF FF 03 0A 08 01 06 0C 00 2F 00 F0 01	FFFF	-1	
203	03	9D	03-00 09 00 9D FF FE 03 0A 08 01 06 0C 00 27 00 E4 01	FFFE	-2	
204	03	9F	03-00 09 00 9F FF FE 03 0A 09 01 06 0C 00 27 00 E4 01	FFFE	-2	1
205	04	9F	04-00 09 02 9F FF FF 03 0A 00 01 06 08 00 24 00 E4 01	FFFF	-1	
206	02	9F	02-00 09 02 9F FF FE 03 0A 00 01 06 08 00 2F 00 F4 01	FFFE	-2	
207	02	A1	02-00 09 02 A1 FF FD 03 0A 01 01 06 08 00 2F 00 F4 01	FFFD	-3	1
208	03	A1	03-00 09 00 A1 FF FD 03 0A 02 01 06 08 00 27 00 E0 01	FFFD	-3	
209	04	A1	04-00 09 02 A1 FF FE 03 0A 02 01 06 08 00 24 00 E4 01	FFFE	-2	
210	04	A3	04-00 09 02 A3 FF FD 03 0A 03 01 06 0C 00 04 00 E4 01	FFFD	-3	1
211	02	A3	02-00 09 02 A3 FF FE 03 0A 04 01 06 0C 00 2F 00 F0 01	FFFE	-2	
212	03	A3	03-00 09 00 A3 FF FE 03 0A 04 01 06 0C 00 07 00 E4 01	FFFE	-2	
213	03	A5	03-00 09 00 A5 00 00 03 0A 05 01 06 0C 00 07 00 E4 01	0000	0	0
214	04	A5	04-00 09 02 A5 00 00 03 0A 06 01 06 0C 00 04 00 E0 01	0000	0	
215	02	A5	02-00 09 02 A5 00 00 03 0A 06 01 06 0C 00 2F 00 F4 01	0000	0	
216	02	A7	02-00 09 02 A7 FF FD 03 0A 07 01 06 0C 00 2F 00 F4 01	FFFD	-3	0
217	03	A7	03-00 09 00 A7 FF FD 03 0A 08 01 06 0C 00 27 00 E0 01	FFFD	-3	
218	04	A7	04-00 09 02 A7 FF FD 03 0A 08 01 06 0C 00 24 00 E4 01	FFFD	-3	
219	04	A9	04-00 09 02 A9 FF FE 03 0A 09 01 06 0C 00 24 00 E4 01	FFFE	-2	0
220	02	A9	02-00 09 02 A9 FF FE 03 0A 00 01 06 08 00 2F 00 F0 01	FFFE	-2	

221	03	A9	03-00 09 00 A9 FF FE 03 0A 00 01 06 08 00 27 00 E4 01	FFFE	-2	
222	03	AB	03-00 09 00 AB FF FD 03 0A 01 01 06 08 00 27 00 E4 01	FFFD	-3	0
223	04	AB	04-00 09 02 AB FF FD 03 0A 02 01 06 08 00 24 00 E0 01	FFFD	-3	
224	02	AB	02-00 09 02 AB FF FD 03 0A 02 01 06 08 00 2F 00 F4 01	FFFD	-3	
225	02	AD	02-00 09 02 AD FF FE 03 0A 03 01 06 0C 00 2F 00 F4 01	FFFE	-2	1
226	03	AD	03-00 09 00 AD FF FD 03 0A 04 01 06 0C 00 07 00 E0 01	FFFD	-3	
227	04	AD	04-00 09 02 AD FF FE 03 0A 04 01 06 0C 00 04 00 E4 01	FFFE	-2	
228	04	AF	04-00 09 02 AF 00 01 03 0A 05 01 06 0C 00 04 00 E4 01	0001	1	1
229	02	AF	02-00 09 02 AF 00 02 03 0A 06 01 06 0C 00 2F 00 F0 01	0002	2	
230	03	AF	03-00 09 00 AF 00 01 03 0A 06 01 06 0C 00 07 00 E4 01	0001	1	
231	03	B1	03-00 09 00 B1 FF FD 03 0A 07 01 06 0C 00 27 00 E4 01	FFFD	-3	1
232	04	B1	04-00 09 02 B1 FF FD 03 0A 08 01 06 0C 00 24 00 E0 01	FFFD	-3	
233	02	B1	02-00 09 02 B1 FF FE 03 0A 08 01 06 0C 00 2F 00 F4 01	FFFE	-2	
234	02	B3	02-00 09 02 B3 FF FC 03 0A 09 01 06 0C 00 2F 00 F4 01	FFFC	-4	1
235	03	B3	03-00 09 00 B3 FF FB 03 0A 00 01 06 08 00 27 00 E0 01	FFFB	-5	
236	04	B3	04-00 09 02 B3 FF FC 03 0A 00 01 06 08 00 24 00 E4 01	FFFC	-4	
237	04	B5	04-00 09 02 B5 FF FD 03 0A 01 01 06 08 00 24 00 E4 01	FFFD	-3	0
238	02	B5	02-00 09 02 B5 FF FD 03 0A 02 01 06 08 00 2F 00 F0 01	FFFD	-3	
239	03	B5	03-00 09 00 B5 FF FD 03 0A 02 01 06 08 00 27 00 E4 01	FFFD	-3	
240	03	B7	03-00 09 00 B7 00 00 03 0A 03 01 06 0C 00 07 00 E4 01	0000	0	1
241	04	B7	04-00 09 02 B7 FF FF 03 0A 04 01 06 0C 00 04 00 E0 01	FFFF	-1	
242	02	B7	02-00 09 02 B7 00 00 03 0A 04 01 06 0C 00 2F 00 F4 01	0000	0	
243	02	B9	02-00 09 02 B9 FF FC 03 0A 05 01 06 0C 00 2F 00 F4 01	FFFC	-4	0
244	03	B9	03-00 09 00 B9 FF FC 03 0A 06 01 06 0C 00 07 00 E0 01	FFFC	-4	
245	04	B9	04-00 09 02 B9 FF FC 03 0A 06 01 06 0C 00 04 00 E4 01	FFFC	-4	
246	04	BB	04-00 09 02 BB 00 01 03 0A 07 01 06 0C 00 24 00 E4 01	0001	1	1
247	02	BB	02-00 09 02 BB 00 01 03 0A 08 01 06 0C 00 2F 00 F4 01	0001	1	
248	03	BB	03-00 09 00 BB 00 00 03 0A 08 01 06 0C 00 27 00 E4 01	0000	0	
249	03	BD	03-00 09 00 BD FF FB 03 0A 09 01 06 0C 00 27 00 E4 01	FFFB	-5	0
250	04	BD	04-00 09 02 BD FF FB 03 0A 00 01 06 08 00 24 00 E0 01	FFFB	-5	
251	02	BD	02-00 09 02 BD FF FB 03 0A 00 01 06 08 00 2F 00 F4 01	FFFB	-5	
252	02	BF	02-00 09 02 BF 00 00 03 0A 01 01 06 08 00 2F 00 F4 01	0000	0	1
253	03	BF	03-00 09 00 BF FF FF 03 0A 02 01 06 08 00 27 00 E0 01	FFFF	-1	
254	04	BF	04-00 09 02 BF 00 00 03 0A 02 01 06 08 00 24 00 E4 01	0000	0	