TESIS DOCTORAL

Extensiones al protocolo MODBUS en el ámbito de los sistemas de control de edificios

Autor:

Francisco Simón Muñiz

Director:

Carlos León de Mora

Departamento de Tecnología Electrónica, Universidad de Sevilla.

C/ Virgen de África, 7. Sevilla 41011 (España)

Tlfno.: (+34) 954 55 27 89, Fax: (+34) 954 55 28 33.

fsimon@us.es

Agradecimientos

A Carlos, mi director de tesis, por su ayuda.

A mi familia, por el tiempo que no les he podido dedicar para realizar este trabajo.

A mis compañeros de departamento, por su apoyo y amistad.

Resumen

En este trabajo se analiza el protocolo MODBUS, en el contexto de los sistemas de control de edificios y se proponen una serie de extensiones, para su mejora, que pretenden aumentar la interoperabilidad del protocolo y facilitar y automatizar los procesos de configuración de la red.

Contenido

0 INTRODUCCIÓN

0 INTRODUCCIÓN

En este trabajo se proponen una serie de extensiones al protocolo Modbus en el ámbito de uso del protocolo en los sistemas de control de edificios.

En el capítulo 1 se realiza una revisión del estado del arte en la que se revisa el uso de los protocolos de comunicación más usuales en distintas áreas de aplicación. Se concreta el estudio en el protocolo MODBUS, núcleo de este trabajo y se hace una revisión bibliográfica sobre trabajos realizados con este protocolo. Se revisa también el uso de la tecnología de objetos empleada en diversos protocolos.

En el capítulo 2 se hace un estudio, con un cierto nivel de detalle, de los protocolos más usados, actualmente, en los sistemas de control de edificios. Este estudio nos permite encontrar líneas de actuación para las propuestas de ampliación del protocolo realizadas en este trabajo.

En el capítulo 3 se muestra primero, la problemática planteada al no existir tipos de datos normalizados y a continuación se hace una propuesta de estandarización de tipos de datos con objeto de que en los mensajes de comunicación estas informaciones sean autoconsistentes y se facilite la interoperabilidad entre dispositivos de distintos fabricantes.

En el capítulo 4 se hace una propuesta de solución a los problemas relacionados con el asignamiento de direcciones:

- Posibilidad de asignar la dirección del esclavo a través del bus de comunicaciones
- El borrado de la dirección individual de un esclavo a través del bus
- El encendido del led de programación desde el bus para la identificación del esclavo
- El Asignamiento automático de direcciones a los esclavos que no tengan asignada dirección
- La resolución de conflictos de dirección debidos a direcciones repetidas

La propuesta desarrolla la creación de 3 nuevas funciones Modbus:

- Función 100: solicita a los esclavos su identificador ModbusID de 48 bits
- Función 101: asigna una dirección a un esclavo en base a su ModbusID
- Función 102: identifica a un esclavo con su led de programación

En el capítulo 5 se realiza una propuesta de creación de clases que permitan la definición de los recursos I/O de los dispositivos y la gestión de horarios locales en los mismos dispositivos.

En el capítulo 6 se proponen nuevas funciones Modbus con los códigos de función 103 y 104 que permitan la gestión de las informaciones contenidas en las instancias de las clases de los dispositivos.

- Función 103: obtención de los identificadores de los objetos
- Función 104: acceso a las propiedades de los objetos

En el capítulo 7, se analizan los resultados del trabajo realizado mediante:

- El estudio de las necesidades de almacenamiento de memoria de los objetos de cada una de las clases definidas en el capítulo 5
- Un estudio del consumo de memoria necesario para la implantación de este sistema sobre un controlador de uso general y se concluye la validez de la solución aportada
- Se propone una distribución de los objetos en los mapas de memoria Modbus convencionales que permitan el acceso a los objetos desde las funciones Modbus clásicas

1 ESTADO DEL ARTE

1 Estado del arte

1.1 PROTOCOLOS DE COMUNICACIÓN

Los sistemas de control modernos están constituidos por un conjunto de dispositivos que realizan las funciones de controladores, sensores o actuadores y que comparten información entre ellos o con un equipo central.

Esta estructura se denomina distribuida ya que cada componente del sistema aporta inteligencia y permite realizar funciones de control complejas en base a las informaciones recogidas de otros elementos de la red y sus propios algoritmos de control.

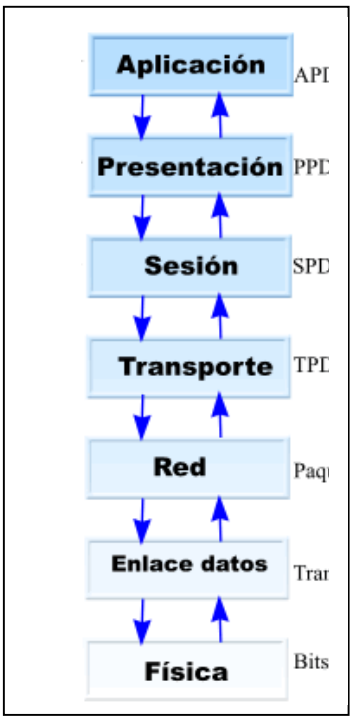
Para poder intercambiar información entre los distintos participantes del sistema de control necesitamos unas normas que sigan todos los dispositivos interconectados que llamamos protocolo de comunicación.

Un protocolo de comunicación es un conjunto de reglas que permiten la comunicación y transmisión de información entre distintos dispositivos. Estas reglas deben definir la sintaxis, semántica y sincronización de la comunicación.

Con objeto de unificar la estructura de los protocolos, distintos organismos se encargan de su estandarización. Los principales son:

- ISO (International Organization for Standarization)
- CCITT (Comité Consultatif International de Télégraphique et Téléphonique)
- EIA (Electronic Industries Association)
- ANSI (American National Standard Institute)

La organización ISO ha desarrollado el modelo OSI (Open Systems Interconnection) [1] que es un modelo de referencia para la mayoría de los protocolos. Este modelo está organizado en 7 niveles y en servicios que permiten el intercambio de información entre los niveles o capas.



Los servicios son invocados a través de primitivas:

- Request
- Indication
- Response
- Confirm

1.2 TIPOS DE PROTOCOLOS

Los protocolos de comunicación se utilizan para permitir el intercambio de información en redes de dispositivos que tienen usos específicos. Algunos ejemplos de redes son:

- Redes para realizar el control de una planta industrial
- Redes de control de energía eléctrica
- Redes públicas para el intercambio de información entre ordenadores
- Redes de dispositivos para el control de viviendas o de edificios
- Redes de dispositivos para el control de un vehículo

Según sea la aplicación, tendrán más importancia unos aspectos u otros de la comunicación como:

- Seguridad
- Velocidad
- Sincronización
- Medios de comunicación soportados
- Requerimientos hardware/software de los dispositivos

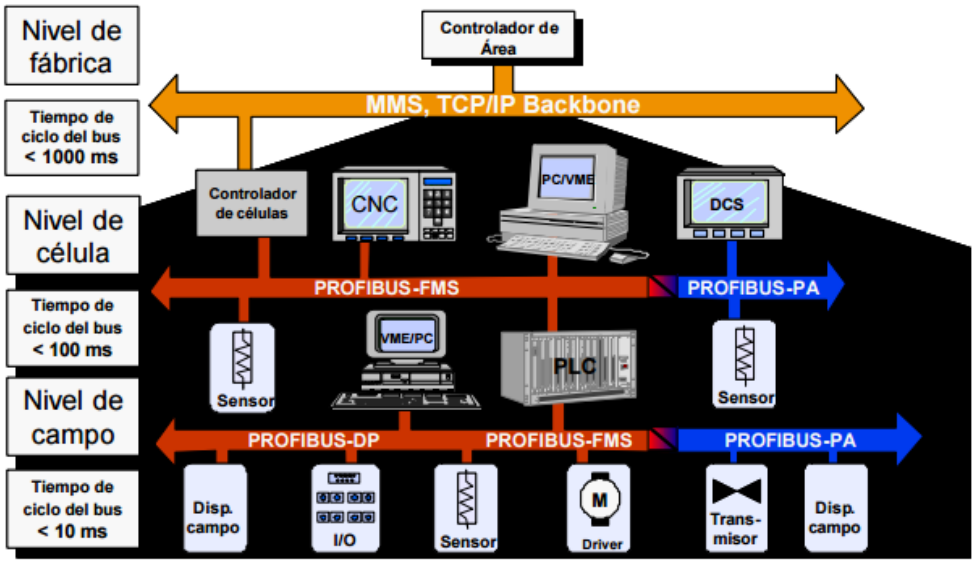
- Necesidad de licencias por dispositivo

1.2.1 PROTOCOLOS INDUSTRIALES

En el control de plantas industriales se usan los protocolos específicos de algunos fabricantes o consorcios de empresas, también protocolos genéricos y también se adaptan protocolos existentes modificándolos o añadiendo nuevas funcionalidades.

Los protocolos de uso industrial más habituales son:

- **Profibus** que está desarrollado por un consorcio de empresas y varios institutos de investigación alemanes. Es un estándar europeo EN50170. Está gestionado por Profibus Internacional (PI) [2]. Hay tres variantes del protocolo (DP, PA y FMS) que cubren los tres niveles de la pirámide de automatización.



- **Modbus** desarrollado por Modicom que pertenece actualmente a Schneider Electric. No está estandarizado aunque sus especificaciones están disponibles [3] [4] [5]. Es uno de los protocolos más veteranos y utiliza una estructura maestro-esclavo. Existe una versión del protocolo para comunicar a través de RS485 (MOBBUS RTU) y otra a través de Ethernet con el protocolo TCP/IP (MODBUS TCP/IP) [6]. Es muy eficiente en aplicaciones de control industrial donde existen muchos puntos de control. Actualmente está utilizado en la línea de autómatas programables de Schneider y de otros muchos fabricantes.

- **Devicenet** fue diseñado por el fabricante de autómatas programables Allen-Bradley aunque ahora es un protocolo abierto gestionado por la asociación ODVA (Open DeviceNet Vendor Association) [7]. Está desarrollado sobre el bus CAN aunque utiliza RS485 en el nivel físico.

- **Compobus** es un bus de comunicaciones implementado por OMRON y derivado de bus DeviceNet. Este sistema no precisa configuración. Se utiliza un cable de dos hilos entre la unidad maestra y los esclavos que suministra la alimentación y los datos.

- **ASI** (Actuator-Sensor Interface) es un sistema para eliminar cableado entre lo sensores y los actuadores. El cable de conexión suministra la alimentación a los dispositivos. Es un sistema abierto definido por el estándar EN50295 y por IEC62026-2.

- **Foundation Fieldbus** [8] agrupa a las asociaciones WorldFip (World Factory Instrumentation Protocol) [9] e ISP (Interoperable Systems Project). Es intrínsecamente seguro y permite longitudes de bus de hasta 1900 metros sin repetidores.

1.2.2 PROTOCOLOS PARA EL CONTROL ELÉCTRICO

Estos protocolos están especializados en el control de sistemas eléctricos como las subestaciones.

Los protocolos más usados son:

- **IEC61850** [10] es un estándar de la Comisión Electrotécnica Internacional creado para lograr una solución completa de la automatización de las subestaciones. Se basa en la interoperabilidad que permite interconectar a IEDs (Intelligent Electronic Device) de distintos fabricantes. Establece un lenguaje de descripción de subestaciones denominado SCL (Substation Configuration Description Language). Incluye también procedimientos de transmisión de tiempo real y alta prioridad GOOSE (Generic Object Oriented Systems Event) para la gestión de eventos críticos como los disparos.
- **DNP3** (Distributed Network Protocol) [11] es un protocolo para la comunicación entre IEDs ampliamente utilizado en el sector eléctrico de Estados Unidos y Canadá. Usa las capas OSI de enlace, transporte y aplicación
- **IEC60870-5** es un protocolo serie asíncrono en su especificación 101 [12] para el telecontrol de canales entre DTE y DCE, adecuado para distintas topologías de interconexión como estrella y punto a punto y con comunicación TCP/IP en su especificación 104 [13]. Normalmente se usa la especificación 104 para los centros de telecontrol y la especificación 101 para la comunicación entre IEDs.
- **MODBUS** se utiliza también en la comunicación de los IEDs de los sistemas eléctricos. Fabricantes como GE, Schneider Electric disponen de IEDs con este protocolo. Su interconexión con el sistema IEC 61850 es sencilla a través de pasarelas de distintos fabricantes [14] [15]

1.2.3 PROTOCOLOS PARA EL CONTROL DE VIVIENDAS O DE EDIFICIOS

Estos sistemas de control presentan particularidades importantes, derivadas del tipo de dispositivos que conforman la red de control. Algunos dispositivos son pequeños como los pulsadores de encendido de iluminación y otros son complejos como los controladores de salas de producción de climatización.

Hoy en día existen muchos protocolos de comunicación utilizados para la automatización de edificios. Se diferencian entre ellos en muchos aspectos como:

- el medio físico empleado en la comunicación (cable de bus, cable eléctrico, etc.)
- la estrategia para establecer la comunicación (maestro-esclavo, canal abierto)
- la estandarización de la información (tipos de datos, perfiles funcionales)

- los procedimientos de programación y configuración de los nodos
- la forma de compartir información entre nodos

Los protocolos más utilizados son:

- **KNX** estándar ISO/IEC 14543, basado en OSI y heredero del bus EIB. Está gestionado por la asociación KNX [16] y tiene una gran implantación en viviendas y en instalaciones de edificios preferentemente eléctricas. Tiene una gran implantación en Europa, Latinoamérica y en China. Es un protocolo vivo con alrededor de 400 fabricantes de productos y que cuenta con más de 40000 técnicos certificados. Este protocolo se estudia en detalle en el capítulo 2.1 de este trabajo.

- **Lonworks** abreviadamente LON es un protocolo creado por la empresa norteamericana Echelon [17] en 1988 y registrado como estándar ISO/IEC 14908-1 Control Network Protocol.

  El mantenimiento del estándar se realiza a través de la organización LONMARK [18]. Durante muchos años ha sido el protocolo dominante en los sistemas de control de edificios debido a la calidad de los productos de control de climatización suministrados por las empresas fabricantes.

  Tiene soluciones de control sobre línea eléctrica (Powerline) muy utilizadas para el control de iluminación pública en calles.

  Hoy en día ha perdido su hegemonía y busca avanzar de nuevo a través de nuevas soluciones y sistemas.

  Una de ellas es el sistema LUMEWAVE que permite controlar la iluminación pública a través de sistemas de comunicaciones inalámbricas bluetooth de largo alcance, con sensores de microondas capaces de detectar vehículos y personas y acomodar la iluminación pública a las necesidades de uso.

  Otra de las nuevas soluciones es una nueva línea de chips controladores IZOT que buscan su integración con los sistemas IOT y que usan el protocolo LON conjuntamente con el protocolo Bacnet sobre sus canales clásicos de comunicación. Este protocolo se estudia con mayor profundidad en el capítulo 2.2 de este trabajo

	Control optimized chips with built-in stacks			Control stacks for generic processors	
	FT 6050	FT 6010	Neuron 6050	Device Stack DX	Device Stack EX
Supports Classic LONWORKS	Yes	No	Yes	Yes	Yes
IP Enabled	Yes	Yes	Yes	Yes	Yes
Protocols Supported	LonTalk, LonTalk/IP-FT, BACnet MS/TP, BACnet/IP, Any UDP sockets based protocol	LonTalk/IP-FT, BACnet MS/TP, BACnet/IP, Any UDP sockets based protocol	LonTalk, LonTalk/IP-FT, BACnet MS/TP, BACnet/IP, Any UDP sockets based protocol	LonTalk, LonTalk/IP-FT, Any UDP sockets based protocol	LonTalk, LonTal/IP-FT, Any UDP sockets based protocol
Popular Channels/PHY	FT-10 (on-chip), RS-485 (external)	FT-10 (on-chip), RS-485 (external)	RS-485 (external)	Eth, Wi-Fi, RF	Eth, Wi-Fi, RF
Class of Processor Required	Neuron (8-bit, 4-core, 80MHz)	Neuron (8-bit, 4-core, 80MHz)	Neuron (8-bit, 4-core, 80MHz)	32-bit, ARM Cortex M3, 200MHz Eg: Marvel 88MC200	32-bit, ARM11, 700MHz Eg: Raspberry Pi
Control Stack Footprint (L4-L6)	6KB	6KB	6KB	60KB	1MB
Number of Address Table Entries	254	254	254	254	32,767
Number of Simultaneous Transactions (in/out)	2 / 2	2 / 2	2 / 2	256 / 256	32,767 / 32,767
Need for OS services (eg: Linux)	None	None	None	Yes	Yes

- **Bacnet** es un protocolo creado por la asociación de fabricantes de climatización de Estados unidos y tiene certificación ISO 16484-5 y ANSI 135-1995.

  Es un protocolo con una estructura de comunicación maestro-esclavo y hoy en día se ha convertido en el protocolo por excelencia en el control de edificios, dominando el mercado sin excluir a la competencia ya que en la misma definición de su estándar permite su interconexión con otros protocolos de campo como KNX, LON y Zigbee. Existe una gran variedad de pasarelas para comunicar también otros sistemas con Bacnet.

  La comunicación de red se realiza a través de UDP, aunque el estándar define también una comunicación sobre RS485 denominada Bacnet MS/TP, con objeto de disponer también de un protocolo de campo que permite implementar dispositivos de bajo coste.

  La información del protocolo es compleja (1022 páginas para la edición de 2010) y debe ser adquirida a través de ASHRAE [19].

  El protocolo se basa en objetos que describen el comportamiento de cada dispositivo que es visto por la red como una colección de objetos. La información de todos los objetos de la red puede ser descubierta y vista por el sistema de control del edificio que es capaz de mostrar todo el contenido del sistema de control sin conocimiento previo de los equipos instalados.

  El protocolo incluye objetos para las alarmas, históricos de datos, calendarios y una gran variedad de objetos de entrada y salida. Este protocolo se describe con más detalle en el capítulo 2.3 de este documento.

- **Modbus** se utiliza también con mucha frecuencia en el control de edificios formando una única red de control o conjuntamente con otros sistemas de control como Bacnet, LON o KNX. Son bastante frecuentes controladores de otros protocolos que incorporan puertos de comunicación Modbus haciendo la función de controladores y pasarelas simultáneamente [20].

Los dispositivos Modbus en edificios están especializados en módulos de entradas y salidas, en controladores de climatización y en medidores eléctricos.

Debido a su estructura maestro-esclavo no es adecuado para algunas tareas de control como los pulsadores de iluminación, salvo que pertenezcan a un controlador de iluminación y su uso sea local al controlador. Este inconveniente es característico de la estructura maestro-esclavo y por tanto otros protocolos como Bacnet tienen el mismo problema.

Este protocolo se describe con más detalle en el capítulo 2.4 de este documento.

1.3 EL PROTOCOLO MODBUS EN EL CONTROL DE EDIFICIOS

El protocolo Modbus, como hemos mostrado anteriormente, es un protocolo muy versátil que tiene una amplia utilización en los sistemas de control en general y en el control de edificios en particular.

Su utilización es muy ventajosa para los fabricantes de equipos, ya que la documentación del protocolo es pública, no precisa de hardware especial ya que las comunicaciones se realizan sobre RS485 o sobre IP y la implementación del protocolo es relativamente sencilla sobre cualquier procesador contando además con varias librerías de código abierto en C que implementan el núcleo del protocolo [21] [22], librerías en java [23], librerías para Labview [24], librerías en PHP [25] y sobre otras muchas plataformas como Arduino etc.

Varios trabajos están también publicados sobre el diseño de equipos Modbus usando los procesadores modernos ARM para la implementación de Modbus TCP/IP [36] y la implementación del protocolo RTU sobre núcleos M0 [26] y M3 [27] y sobre plataformas Linux empotradas [28] y sobre la integración de sensores de variables típicas como tensión, corriente y temperatura [29]. También hay varios trabajos sobre integración del protocolo Modbus con sistemas inalámbricos Zigbee [30].

Para el usuario final, los productos Modbus representan, frecuentemente, un ahorro económico importante frente a productos realizados con otros protocolos. Se dispone, además, de muchos fabricantes de software que incorporan Modbus en sus productos de control de edificios (Building Management System) y también de servidores OPC que facilitan la integración de los datos de dispositivos Modbus en sistemas SCADA convencionales. Existen también pasarelas con el resto de protocolos estándares.

A pesar de las virtudes del protocolo Modbus, si comparamos su funcionalidad con otros protocolos dedicados al control de edificios vemos que otros aportan funcionalidades muy interesantes que no están disponibles en Modbus.

1. LON utiliza las variables de red y KNX las direcciones de grupo que permiten que una información enviada por un dispositivo se reciba simultáneamente por otros dispositivos. Además la elección del destino de estas informaciones se define en el proceso de instalación y no en el de programación.
2. La asignación de las direcciones de los dispositivos se realiza en LON y KNX sin elementos de hardware especiales. Se usa solo un pulsador en el dispositivo para aceptar la dirección propuesta por el software de configuración. En el caso de LON, es posible además asignar la dirección sin el pulsador a través de los códigos de barras del NeuronID suministrados por el fabricante.

3. Bacnet permite a través de sus servicios obtener toda la información de la instalación de los dispositivos Bacnet IP y de los dispositivos Bacnet MS/TP a través de los routers.
4. LON, KNX y Bacnet usan tipos de datos definidos lo que permite interpretar la información recibida desde el canal de comunicación sin ambigüedades.

Las características definidas en el punto 1 anterior no están permitidas en el protocolo Modbus ni en Bacnet por la estructura maestro-esclavo de los protocolos. No obstante es posible simular este comportamiento a través de mensajes broadcast que si pueden ser recibidos por todos los dispositivos de la red.

1.4 TRABAJOS PUBLICADOS SOBRE MEJORAS AL PROTOCOLO MODBUS

Durante años se han realizado trabajos, propuestas de modificaciones y patentes relacionadas con el protocolo Modbus RTU y TCP/IP.

1.4.1 SEGURIDAD EN LAS COMUNICACIONES

La seguridad en las comunicaciones es un aspecto muy importante sobre todo cuando se controlan instalaciones públicas o económicamente sensibles como es el caso de las redes de abastecimiento.

La comunicación con el exterior de las instalaciones puede ser controlada con cortafuegos y equipos especiales, pero si el ataque se produce desde el interior de las instalaciones no es posible generalmente evitarlo.

Los equipos y routers Modbus TC/IP pueden incorporar una tabla de acceso que solo permita la conexión desde direcciones IP fijas y previamente conocidas, de esta forma se realiza la acción DROP para descartar los paquetes si la dirección IP no está en la tabla de filtros y ACCEPT para aceptarla si está contenida en la tabla de filtros.

Si el ataque se produce desde una dirección IP válida, el sistema de filtros es incapaz de actuar sobre el ataque.

Si el ataque se realiza sobre la red RS485, es decir sobre el protocolo Modbus RTU, no tenemos mecanismos para detener la intrusión ya que los esclavos no son capaces de detectar qué maestro ha realizado el envío del mensaje de la petición de datos.

El problema planteado sugiere que el procedimiento de trabajo debe centrarse en la detección de los ataques en primer lugar y no en los procesos de bloqueo, que como se ha comentado, resultas ineficaces.

Diversos trabajos están publicados para la mejora de la seguridad de las comunicaciones, relacionados con el protocolo Modbus TCP/IP. Abdalhossein Rezai y otros [39] presentan un trabajo para la mejora de la seguridad basado en un nuevo sistema de claves criptográficas, Niv Goldemberg y Avishai Wool describen en [40] un sistema de detección de intrusión analizando la repercusión de los malware en estos sistemas de comunicación, Igor Nai Fovino y otros [41] también estudian la vulnerabilidad de los sistemas de comunicaciones a los ataques de virus y programa maliciosos, Abdulmohsen Almalawi y otros [42] proponen un sistema de identificación de intrusión basado en la

identificación de estados consistentes e inconsistentes del sistema de control y una extracción de reglas de los estados detectados, Peter Huitsing y otros [43] hacen un estudio basado en taxonomías para evaluar el riesgo de ataques sobre Modbus serie y TCP/IP, Cristina Alcaraz y otros [44] analizan los problemas de seguridad en las comunicaciones TCP/IP en el contexto de las IOT, William Knowles y otros [45] analizan una métrica de seguridad como una propuesta de línea de trabajo para la detección de los ataques, Thomas Morris y otros [46] crean un banco de pruebas universitario basado en los sistemas de seguridad de las instalaciones críticas para la validación de estos sistemas.

Dentro del ámbito europeo, se han desarrollado diversos proyectos dentro de los programas marco 6 y 7 relacionados con la seguridad de las comunicaciones. CI2RCO [47] se ha centrado en los mecanismos de protección de las comunicaciones relacionados con las infraestructuras críticas, CRUTIAL [48] usa un modelo de interdependencias para proponer nuevas arquitecturas y diseños. En el séptimo programa marco, diversos proyectos han trabajado en el soluciones arquitecturales y de gestión de riesgos, EMILI [49], SERCIS [50], INSPIRE [51], ESCoRTS [52], EURACOM [53] etc.

1.4.2 MODIFICACIONES DEL PROTOCOLO

Diversas propuestas se han realizado para simplificar, mejorar y adaptar el protocolo Modbus RTU y TCP/IP.

Juraj Dud'ak y otros [32] proponen una modificación del protocolo Modbus RTU para su utilización en pequeños dispositivos sensores. El protocolo modificado recibe el nombre de uBUS. Las modificaciones propuestas en este trabajo son:

- Modo multiesclavo: que permite que un único dispositivo físico se comporte como varios dispositivos lógicos
- Expansión del campo de direcciones del esclavo que pasa de 1 byte a 2 bytes. Los 12 bits de mayor peso definen la dirección física del esclavo y los 4 bits de menor peso permiten acceder a uno de los 16 dispositivos virtuales del esclavo.
- Modificación del CRC para permitir encriptación asimétrica RSA.

Giuliano B. M. Guarese y otros [33] [34] plantean el uso conjunto del protocolo inalámbrico Zigbee con el protocolo Modbus RTU. El sistema se articula a través de pasarelas con comunicación dual y se introduce el concepto de multimaestro a través del multiplexado de los maestros.

Yinglan Fang y otros [35] presentan una alternativa al sistema de comunicación maestro-esclavo en base a un sistema de detección de colisiones. El trabajo propone un doble uso del canal de comunicación:

- Modo maestro-esclavo: en este modo el canal está ocupado por el control del maestro sobre uno o varios esclavos. Los nodos no implicados en la comunicación están en modo monitorización y no pueden acceder a la transmisión en el bus.
- Fase de comunicación libre: en este modo el canal de comunicación no está ocupado. Los nodos pueden entonces competir entre ellos para ocupar el canal. Solo el nodo que adquiere los derechos ocupa el canal y establece entonces una comunicación maestro esclavo con otros dispositivos.

El sistema usa una comunicación compartiendo el tiempo del canal de comunicación y demuestra que este sistema es más eficiente para tramas cortas menores que 100 bytes.

LIU Wen-jun y LI Xiang-yang [37] y Yung-Hsiang Liu y otros [38] describen un procedimiento para la resolución de conflictos en el asignamiento de direcciones de los dispositivos esclavos en una red.

Este método identifica a los esclavos cuyas direcciones producen conflicto en la red y modifica las direcciones de los esclavos conflictivos.

1.5 ESTRUCTURA DE OBJETOS

Los sistemas de control precisan, para poder controlar la red, del conocimiento previo de los dispositivos instalados, sus recursos externos de I/O y de información complementaria para que el equipo central pueda realizar el intercambio de información con los dispositivos de campo y los puntos de control.

Si no conocemos esa información no es posible, por parte del sistema central, realizar los ciclos de petición y el intercambio de datos con los dispositivos de la red.

Algunos protocolos son capaces de detectar los dispositivos y recursos instalados en una red de manera automática, otros protocolos precisan de ficheros de configuración o bases de datos con la información de la red. Esta información se debe crear en la fase de instalación y normalmente no es recuperable desde la propia red.

La solución más adecuada para documentar las características I/O de los dispositivos, sus entradas y sus salidas es en base a una estructura clásica de objetos.

Normalmente se definen los siguientes tipos de elementos:

- *Clases*: contienen la definición de los objetos a través de un conjunto de propiedades o atributos. Estas propiedades sirven para la descripción de la propia clase, para modificar el comportamiento por defecto de la clase y para contener los valores asociados al objeto, Cada propiedad podrá usarse en operaciones de lectura, escritura o ambas según se especifique en la descripción del objeto.
- *Instancias*: son las colecciones de objetos pertenecientes a una clase concreta. Cada instancia de una clase tiene entidad propia de forma que podemos leer o escribir los valores de las propiedades de ese objeto de manera independiente al resto de objetos de la misma clase.
- *Servicios*: son las operaciones que podemos realizar sobre los objetos. De manera habitual, tendremos una función para leer el valor de las propiedades “GET” y otra función para escribir y modificar el valor de una propiedad “SET”. Podemos tener servicios distintos en función del tipo de clase. Así por ejemplo, además de las clases relacionadas con sus I/O, un dispositivo suele tener una clase que describe al propio dispositivo “CLASE DEVICE”. Sobre esta clase se pueden realizar operaciones especiales como RESET, ONLINE, OFFLINE etc., que no existen en otras clases.

Un ejemplo que ilustra esta estructura, es la información relacionada con las salidas binarias de un dispositivo.

Necesitamos primero la definición de una clase específica, para la gestión de este tipo de recurso, que contenga las informaciones necesarias para su identificación a través de un nombre y/o un código y otros datos de interés como el valor binario de la salida. Cada una de estas informaciones recibe el nombre de *propiedad*.

Así pues, definimos una clase con esta estructura:

Clase: SALIDAS BINARIAS

Propiedades:

Nombre de la salida

Código de la salida

Valor de la salida

Una instancia de la clase SALIDAS BINARIAS será un objeto que contendrá las informaciones de su Nombre, Código y Valor.

Si un dispositivo tiene 2 salidas binarias, cada una de ellas precisa de las informaciones de Nombre, Código y Valor por lo que precisaremos de 2 instancias de la clase SALIDAS BINARIAS para describir al dispositivo.

Instancia 1:

Nombre: SB1

Código: 1

Valor: ON

Instancia 2:

Nombre: SB2

Código: 2

Valor: OFF

Los servicios nos indican qué tipo de operaciones podemos realizar sobre las propiedades. Si suponemos que el nombre y el código se han asignado en el proceso de programación del dispositivo y sus valores no se pueden alterar, permitiremos la operación de lectura pero no la de escritura. En el caso de la propiedad Valor, debemos poder leer el valor actual de la salida y también modificarlo forzando el estado ON u OFF en la salida. Necesitamos, por tanto, poder realizar la operación de lectura y también la de escritura. Esto lo expresaremos en la definición de la clase indicando las operaciones permitidas con la letra R para la lectura y W para la escritura.

Clase: SALIDAS BINARIAS

Propiedades:

Nombre de la salida, R

Código de la salida, R

Valor de la salida, RW

No existe un procedimiento unificado en los distintos protocolos que incluya una estructura de objetos común, ni procedimientos para la configuración ni tampoco para la recuperación de la información de la red una vez instalada.

1.5.1 PROTOCOLO BACNET

Uno de los sistemas más avanzados en cuanto a la autodocumentación de la red se refiere, es el protocolo Bacnet.

En este protocolo, se utiliza una estructura de objetos que documenta los recursos y funcionalidades de cada dispositivo de una instalación y que permite a partir de los servicios WHO-IS, I-AM, WHO-HAVE y I-HAVE identificar los equipos de la instalación y posteriormente sus recursos de entradas y salidas a través de su estructura de objetos.

Los sistemas de control especializados, integran un procedimiento de exploración y visualización directa de toda la red (BROWSE) que, sin conocimiento previo de la misma, permite mostrar los puntos de control e integrarlos en su árbol de puntos.

Cada fabricante suministra con sus equipos un fichero, denominado PICS, de descripción de los recursos y servicios implementados en cada equipo.

1.5.2 PROTOCOLO KNX

El protocolo KNX no usa estructura de objetos, la configuración se realiza a través del programa ETS. En esta aplicación se crean las direcciones de grupo y se enlazan a los objetos de comunicación de cada aparato.

El enlace con los programas de control BMS se realiza a través de las direcciones de grupo. Es necesaria una exportación de las direcciones de grupo desde el programa ETS y su posterior conversión al formato del BMS utilizado. Algunos fabricantes de BMS crean sus propios plugins para automatizar esta tarea de exportación/importación de las direcciones de grupo.

Si no disponemos de la base de datos ETS, creada en una instalación, el programa ETS no permite la creación de la base de datos desde los dispositivos de campo con lo que no será posible su integración en los programas de control y supervisión.

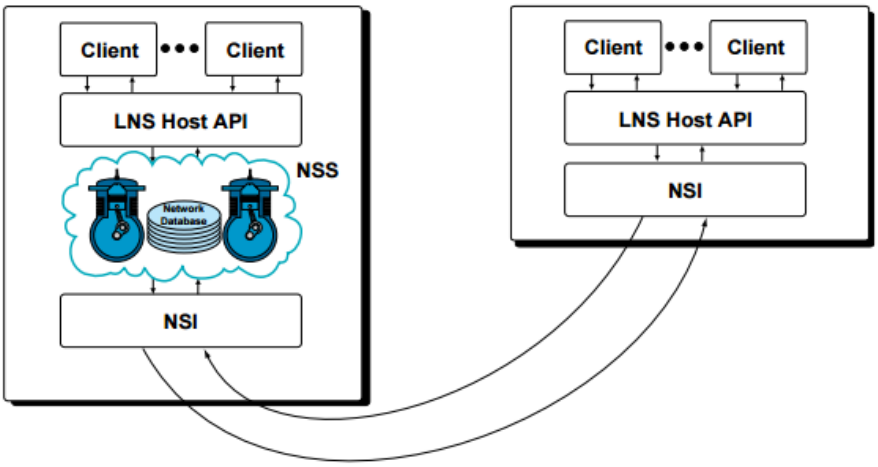
1.5.3 PROTOCOLO LON

En el protocolo LON se genera un fichero XIF de información de las variables de red y los perfiles funcionales de cada aparato. A partir de estos ficheros se pueden conectar las variables de red con programas especiales como LonMaker o LNS220.

En los sistemas LON, la información de la instalación reside en una base de datos gestionada por la aplicación LNS SERVER. Esta aplicación actúa como un servidor de datos local y remoto a diversos clientes conectados a ella para permitir el intercambio de información. Una forma típica de enlazar los

datos a un BMS es a través de un OPC server que se conecta como cliente al LNS SERVER y actúa como servidor OPC a la aplicación BMS que actúa a su vez como cliente OPC.

En teoría es posible regenerar la base de datos del LNS SERVER desde la propia instalación, pero en la práctica el procedimiento es muy lento y no siempre se consigue restaurar la información completa.



1.5.4 PROTOCOLO ZIGBEE

El protocolo Zigbee también define una estructura de objetos en su librería de clusters [60].

Los clusters están agrupados en las siguientes categorías:

General	Contiene clusters que suministran funciones y atributos que no son específicos de otro dominio funcional
Closures	Contiene clusters e información para construir dispositivos para aplicaciones de cerramiento, por ejemplo, controladores de toldos y puertas
HVAC	Contiene clusters e información para construir dispositivos en el dominio de la climatización, por ejemplo bombas
Lighting	Contiene clusters e información para construir dispositivos en el dominio de la iluminación, por ejemplo balastos
Measuremente and sensing	Contiene clusters e información para construir dispositivos en el dominio de la medida y sensores, por ejemplo sensores de temperatura o detectores de movimiento
Security and safety	Contiene clusters e información para construir dispositivos en el dominio de la seguridad, por ejemplo equipos de alarmas
Protocol interfaces	Contiene clusters e información para construir dispositivos en el dominio de la interfaz con otros protocolos, por ejemplo Bacnet

Estos clusters se integran después en los dispositivos definidos en cada uno de los perfiles como por ejemplo en Home Automation y Building Automation.

1.5.5 PROTOCOLO DEVICENET

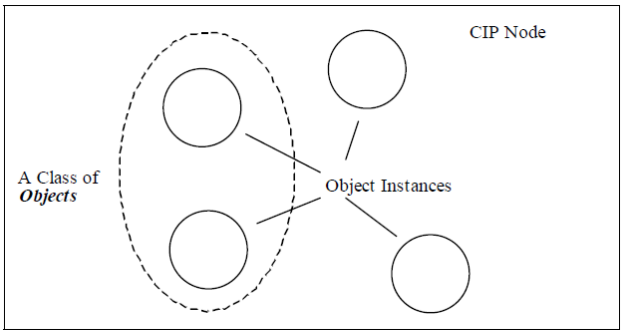
Otro ejemplo de sistema que usa la estructura de objetos es The Common Industrial Protocol (CIP™) publicado por Open DeviceNet Vendor Association, Inc. (ODVA) [7].

Este documento está definido para el protocolo deviceNet y en su capítulo 5 Object Library se define un amplio conjunto de clases.

CIP hace uso del modelo de objetos para describir:

- Los servicios de comunicación disponibles
- El comportamiento externo de un nodo CIP
- El intercambio de información entre nodos.

Cada nodo se modela como una colección de objetos.



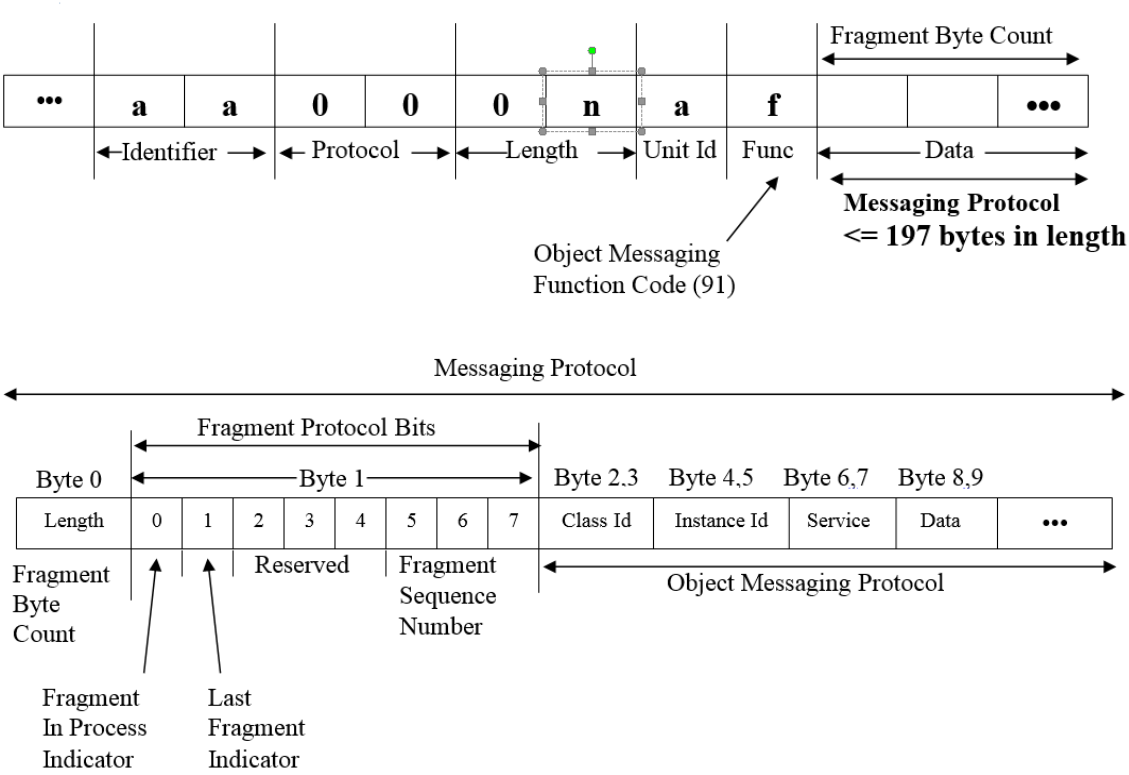
CIP introduce también el concepto de comportamiento (Behavior) que consiste en una especificación de cómo actúa un objeto en base a la acción de los eventos que el objeto maneja, por ejemplo timers o fallos internos.

1.5.6 PROTOCOLO MODBUS

Aunque el protocolo Modbus está basado en el modelo de funciones, es posible su adaptación para la utilización de objetos.

En [62] se describe un procedimiento para intercambiar informaciones de clases usando las funciones estándares de Modbus (funciones 3 y 16). Para distinguir el uso especial de estas funciones se utiliza una firma digital con el texto “SEMI” seguido de un checksum en los bytes que siguen al código de función.

En [61] se describe un proceso de utilización de clases, a través de una nueva función 91, para el protocolo Modbus TCP/IP. Aunque la definición es para la versión del protocolo TCP/IP es perfectamente válida también para el protocolo RTU.



No aparece en este documento ninguna referencia a cuales deben ser los tipos de clases y servicios. Tampoco se especifica el mensaje de respuesta del esclavo.

En el protocolo Modbus estándar, está definida la función 43 para leer la identificación del dispositivo con algunos datos descriptivos. No es realmente una clase tipo dispositivo sino solo unas cuantas informaciones meramente descriptivas.

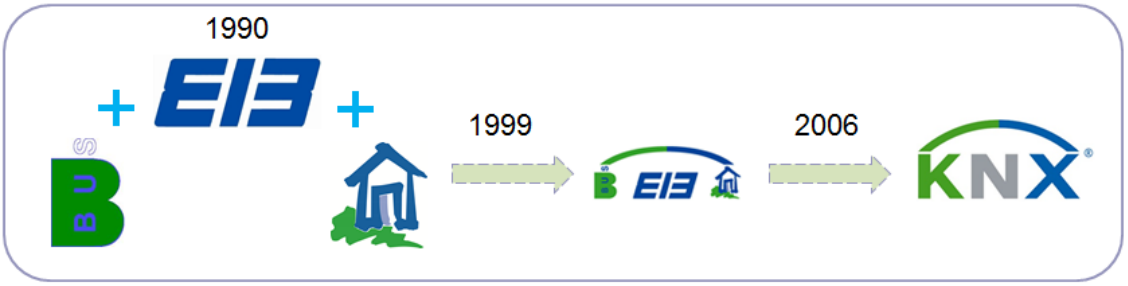
No hay ningún problema en seguir usando la función 43 con los datos especificados por esta función y posteriormente crear una clase del tipo “dispositivo” que contenga, además de esas mismas informaciones, las complementarias que se requieran para una completa gestión del dispositivo.

2 PROTOCOLOS ESTÁNDARES EN CONTROL DE EDIFICIOS

2 Protocolos estándares en control de edificios

2.1 KNX

Creado en 1990 con el nombre de EIB y fusionado posteriormente con los sistemas Batibus y EHS. La asociación KNX tiene su sede en Bruselas y cuenta con más de 300 fabricantes asociados.



Tiene certificaciones CENELEC EN 50090 para domótica, CEN EN 13321-1 (Medios + Protocolo) y EN 13321-2 (KNXnet/IP), ISO/IEC 14543-3-1 hasta 7, GB/T 20965 y ANSI/ASHRAE 135.

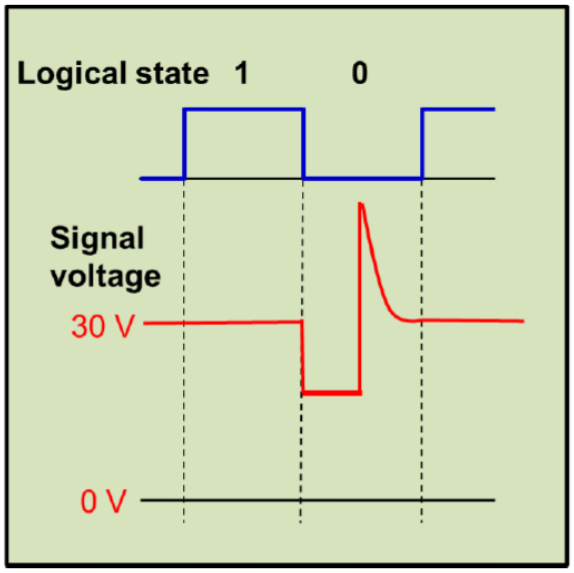
Soporta diversos medios de comunicación como par trenzado, línea eléctrica (PL), wireless KNX/RF2 a 868Mhz y Wifi y Ethernet IP. Permite dos modos de configuración System Mode y Easy Mode.

El medio de comunicación más habitual en las instalaciones es a través de cable de dos hilos sin apantallar (TP) por el que se transmiten los datos y se alimenta a la electrónica de los nodos que componen el sistema de control. La alimentación se realiza a través de una fuente de alimentación de 30 VDC. La velocidad de comunicación es de 9600 b/s.

El protocolo utilizado es de canal abierto de manera que cualquier nodo que desee enviar una información al bus, lo puede hacer en cualquier momento. Como son posibles las colisiones se utiliza el procedimiento **CSMA/CA** (**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance) para detectar y resolver las colisiones que se produzcan.

2.1.1 NIVEL FÍSICO TP

La comunicación en el bus se realiza con el envío de ceros y unos lógicos en forma de tensión de 30 VDC y de cortocircuitos de corta duración como se indica en la figura.



Cuando se transmite un uno lógico no circula corriente por el bus y la tensión se mantiene a los valores de la fuente de alimentación.

Cuando se transmite un cero lógico circula corriente por el bus (cortocircuito), la tensión cae durante un pequeño intervalo de tiempo y después se produce un pico de tensión que proviene de la bobina asociada a la fuente de alimentación.

Si se produce una colisión en el bus, transmisión de un cero y un uno simultáneamente, prevalece el cero

2.1.2 TOPOLOGÍA: DIRECCIONES FÍSICAS

Los equipos que componen el sistema de control se conectan entre sí a través del cable TP de manera estructurada.

La unidad de instalación es la línea que debe contener obligatoriamente una fuente de alimentación y de 1 a 64 dispositivos. Es posible aumentar este número a 256 dispositivos utilizando amplificadores para grupos de 64 dispositivos y nuevas fuentes de alimentación. A cada grupo de 64 dispositivos con su fuente de alimentación y amplificador se le denomina segmento de línea.

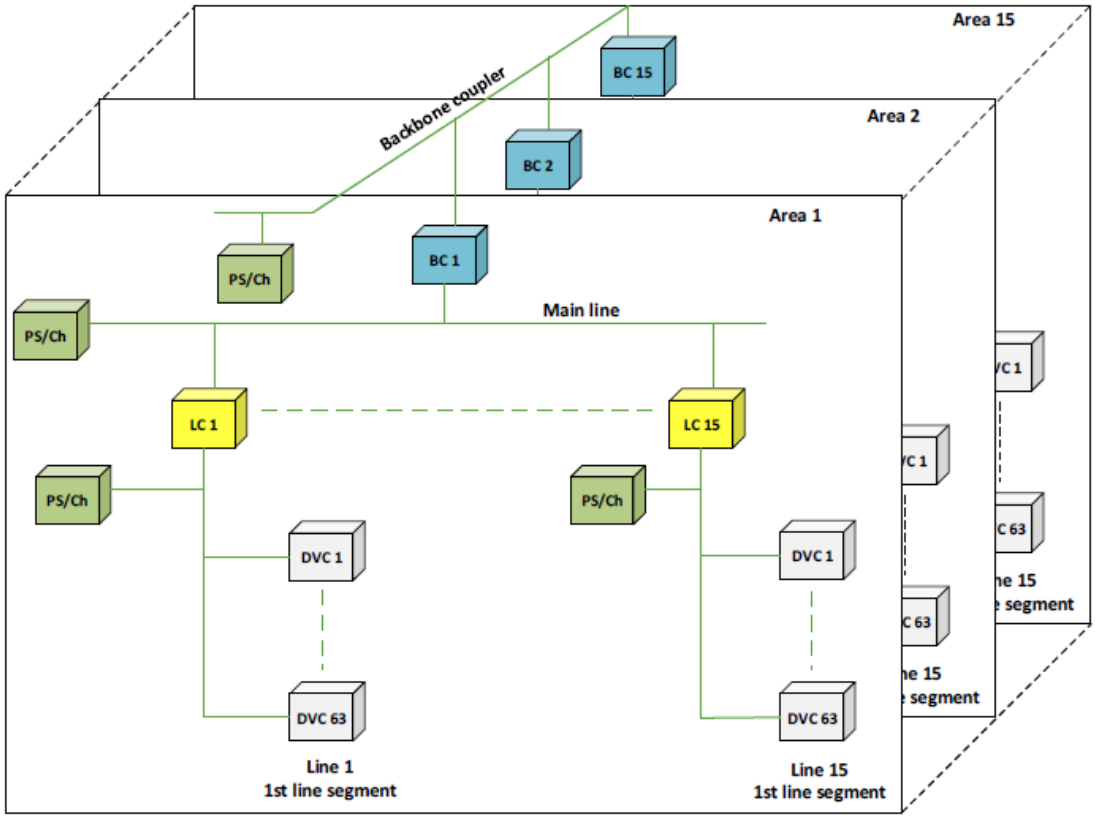
Si necesitamos instalar más dispositivos, podemos crear hasta 15 líneas interconectadas cada una de ellas por un acoplador de líneas. La misión del acoplador de líneas es filtrar los mensajes que no deben pasar de un lado al otro del acoplador y reducir por tanto el tráfico de mensajes en cada unidad del bus.

Los acopladores de línea se unen en una línea principal que también tiene instalada una fuente de alimentación y en la que también se pueden conectar hasta 64 dispositivos. El conjunto de las 15 líneas con sus acopladores se denomina área.

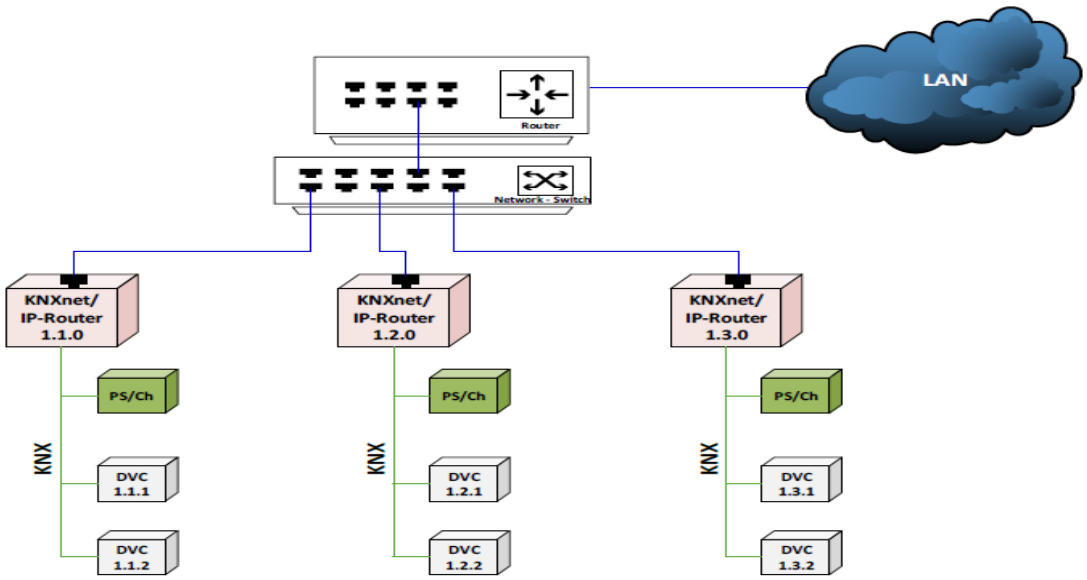
Si necesitamos instalar todavía más aparatos, podemos crear hasta 15 áreas, interconectadas entre sí a través de acopladores de área que realizan la función de filtrado entre los mensajes de las áreas.

Necesitamos también una fuente de alimentación en la línea principal de las áreas, en la cual pueden conectarse también hasta 64 aparatos.

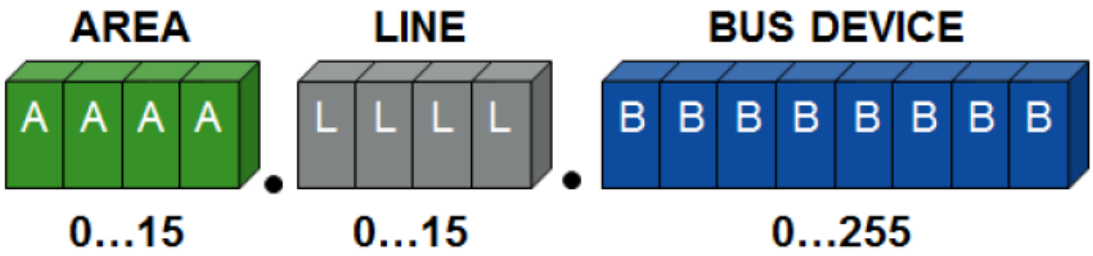
Los amplificadores, acopladores de línea y acopladores de área, son los mismos dispositivos y conectan cable TP en sus dos lados. Realizan su función en base a su localización dentro de la topología de la instalación.



Es posible también combinar aparatos TP con IP sustituyendo los acopladores de líneas o de áreas por Routers o Gateways TP/IP. De esta forma se eliminan las restricciones sobre el número de máximo de dispositivos a conectar y se mejora la velocidad en el bus principal al pasar de 9600 b/s a 100 Mb/s.



Cada dispositivo se identifica a través de su dirección física que es única en la instalación. La dirección física representa la localización de un dispositivo en la topología de la instalación y suministra la información de su área, línea y dispositivo.



La dirección física de un dispositivo se utiliza como dirección origen en los mensajes transmitidos por el dispositivo y también como dirección destino en los mensajes que se refieren a la puesta en marcha del dispositivo (programación de la aplicación y modificación de sus parámetros).

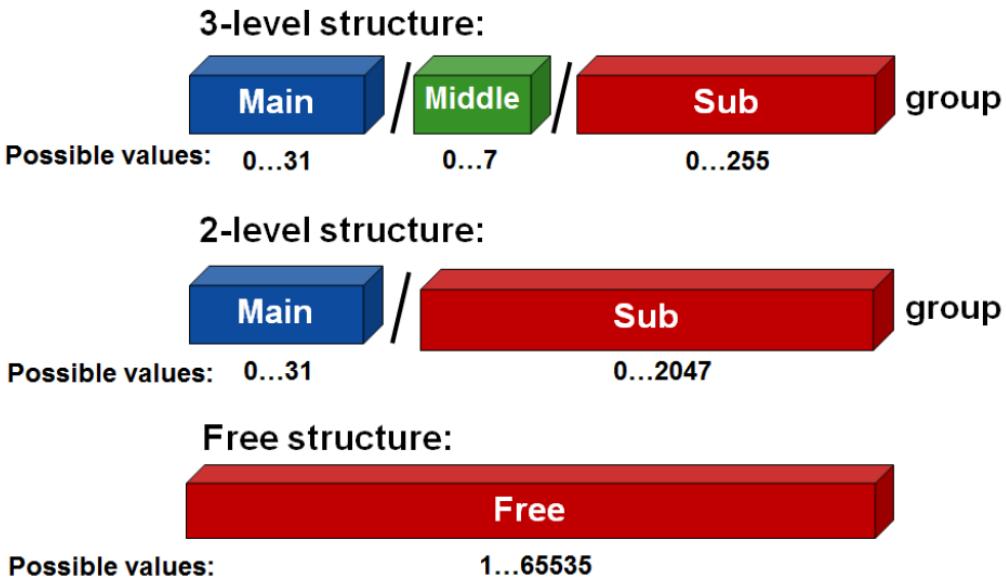
2.1.3 DIRECCIÓN DE GRUPO: OBJETOS DE COMUNICACIÓN

Una vez instalados los dispositivos en una red KNX y después de su programación y parametrización, el funcionamiento de la instalación se realiza a través de las direcciones de grupo.

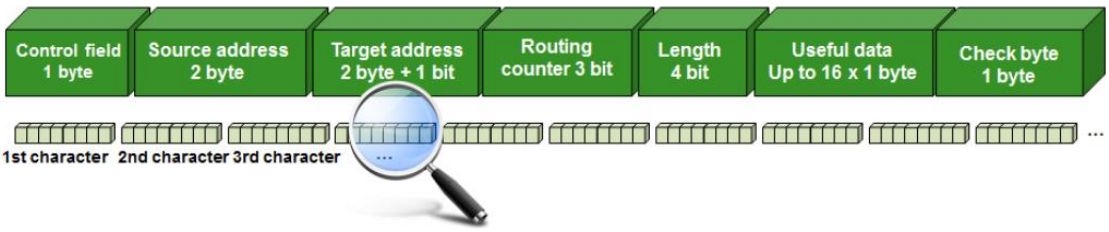
Una dirección de grupo representa una función de control de la instalación, por ejemplo encender las luces del circuito 7. Cada función de control precisa de una dirección de grupo distinta y todos los dispositivos que participen en esa función de control deben tener vinculados esa dirección de grupo.

Si en el encendido de una luz participan un pulsador y un actuador, los dos aparatos usarán la misma dirección de grupo ya que ambos colaboran en la función de control del encendido y apagado de esa luz.

Una dirección de grupo se representa por un valor de 16 bits que pueden estructurarse en uno, dos o tres campos.



Los mensajes que se transmiten desde los aparatos se denominan telegramas y tienen la siguiente estructura:



La dirección destino es la dirección de grupo una vez que los aparatos están programados y configurados.

Los programas que se ejecutan en un aparatos están creados por su fabricante y distribuidos a los usuarios en ficheros especiales que se denominan bases de datos.

El programa ETS que se distribuye a través de la asociación KNX permite importar las bases de datos de los aparatos y configurar la instalación asociando las direcciones físicas a los aparatos, creando las direcciones de grupo, modificando los parámetros de los programas y enlazando las direcciones de grupo con los objetos de comunicación.

Estos programas se pueden configurar para realizar funciones diferentes. Por ejemplo un pulsador se puede utilizar para la función de encender/apagar una luz si el actuador con el que colabora es binario. Si el actuador es de regulación (dimmer), el pulsador debe funcionar de manera diferente para poder encender/apagar la luz pero también para permitir la regulación de luz aumentando o disminuyendo los niveles de iluminación.

La forma que el programa de un dispositivo KNX tiene de intercambiar información con otros dispositivos conectados al bus es a través de objetos de comunicación, también denominados objetos de grupo.

Consideremos el ejemplo anterior del encendido y apagado de una luz. Usaremos un pulsador de 3 teclas y un actuador binario para realizar el control de la luz.

Una vez configurado con el programa ETS el pulsador para la función de conmutación nos aparecen los objetos de comunicación indicados en la figura.

Número	Nombre	Función del Objeto	Descripción	Direcciones de Grupo	Longit...	...	R	W	T	U	Tipo de Datos	Priorid...
6	Tecla 1	Telegrama de conexión			1 bit	C	-	W	T	-		Baja
8	Tecla 2	Telegrama de conexión			1 bit	C	-	W	T	-		Baja
10	Tecla 3	Telegrama de conexión			1 bit	C	-	W	T	-		Baja

Objetos de Grupo | Parámetros | Puesta en marcha

Cada tecla tiene un único objeto de comunicación de tipo bit que se denomina Telegrama de conexión. Cada vez que se pulsa la tecla 1 se transmitirá un telegrama al bus con el valor de encendido/apagado siempre que el objeto de comunicación esté enlazado con una dirección de grupo. Vemos en la gráfica que la columna de direcciones de grupo está vacía y por tanto no se transmitirá el telegrama.

Si observamos los objetos de comunicación del actuador binario, vemos dos objetos de tipo bit asociados al canal A del actuador que son *conmutar* y *estado de conmutación*.

Número	Nombre	Función del Objeto	Descripción	Direcciones de Grupo	Longit...	...	R	W	T	U	Tipo de Datos	Priorid...
10	Salida A	Conmutar			1 bit	C	-	W	-	-		Baja
29	Salida A	Estado de conmutación			1 bit	C	R	-	T	-		Baja
30	Salida B	Conmutar			1 bit	C	-	W	-	-		Baja
49	Salida B	Estado de conmutación			1 bit	C	R	-	T	-		Baja

Objetos de Grupo | Parámetros | Puesta en marcha

Cada vez que el actuador recibe un telegrama a través de una dirección de grupo vinculada al objeto conmutar, el actuador realizará la conmutación solicitada (encendido o apagado) y transmitirá al bus su nuevo estado a través de la dirección de grupo vinculada al objeto de comunicación estado conmutación.

Para que este sistema funcione debemos crear una dirección de grupo para la función encender/apagar y otra para la función estado conmutación. Usando el programa ETS crearemos la dirección de grupo 1/1/1 para la conmutación y 1/1/2 para el estado de conmutación y enlazaremos las direcciones de grupo a los objetos de comunicación.

Número	Nombre	Función del Objeto	Descripción	Direcciones de Grupo	Longit...	...	R	W	T	U	Tipo de Datos	Priorid...
6	Tecla 1	Telegrama de conexión		1/1/1 1/1/2	1 bit	C	-	W	T	-		Baja
8	Tecla 2	Telegrama de conexión			1 bit	C	-	W	T	-		Baja
10	Tecla 3	Telegrama de conexión			1 bit	C	-	W	T	-		Baja

Objetos de Grupo | Parámetros | Puesta en marcha

Número	Nombre	Función del Objeto	Descripción	Direcciones de Grupo	Longit...	...	R	W	T	U	Tipo de Datos	Priorid...
10	Salida A	Conmutar		1/1/1	1 bit	C	-	W	-	-		Baja
29	Salida A	Estado de conmutación		1/1/2	1 bit	C	R	-	T	-		Baja
30	Salida B	Conmutar			1 bit	C	-	W	-	-		Baja
49	Salida B	Estado de conmutación			1 bit	C	R	-	T	-		Baja

Objetos de Grupo | Parámetros | Puesta en marcha

La tecla 1 transmite el valor de encendido/apagado a través del objeto de comunicación Telegrama de conexión y la dirección de grupo 1/1/1.

El actuador recibe el valor del pulsador a través del objeto de comunicación Conmutar que está enlazado con la misma dirección de grupo 1/1/1, realiza la operación solicitada y transmite al bus su estado a través del objeto de comunicación Estado de conmutación enlazado a la dirección de grupo 1/1/2.

El pulsador recibe el valor de su estado para su sincronización. Debemos tener en cuenta que podíamos tener dos pulsadores distintos conectados a la dirección de grupo 1/1/1 y por tanto cada pulsador debe saber si otro dispositivo ha ejecutado una orden al actuador y debe sincronizar su estado con el nuevo valor del actuador.

El mismo pulsador se puede configurar a través de sus parámetros para hacer una regulación de luz. En este caso sus objetos de comunicación cambian a los mostrados en la figura.

Número	Nombre	Función del Objeto	Descripción	Direcciones de Grupo	Longit...	...	R	W	T	U	Tipo de Datos	Priorid...
6	Tecla 1 -corta	Telegrama de conexión			1 bit	C	-	W	T	-		Baja
7	Tecla 1 -larga	Telegrama de regulación			4 bits	C	-	W	T	-		Baja
8	Tecla 2	Telegrama de conexión			1 bit	C	-	W	T	-		Baja
10	Tecla 3	Telegrama de conexión			1 bit	C	-	W	T	-		Baja

Objetos de Grupo | Parámetros | Puesta en marcha

Vemos que hay ahora dos objetos de comunicación. El objeto Tecla1-corta tiene asociada una información de 1 bit y enviará al bus la orden de encender/apagar. El objeto Tecla1-larga es de 4 bits y se utiliza para enviar telegramas de regulación relativa que permitan incrementar o decrementar el nivel de iluminación. Este objeto no se puede vincular a ninguno del actuador binario pero si a los objetos de comunicación de un actuador regulador (dimmer).

En los objetos de conmutación del dimmer aparece el objeto de comunicación Atenuación relativa de 4 bits que es compatible por tanto con el tipo de datos del objeto de comunicación de la tecla larga.

Número	Nombre	Función del Objeto	Descripción	Direcciones de Grupo	Longit...	...	R	W	T	U	Tipo de Datos	Priorid...
10	Salida A	Conmutar			1 bit	C	-	W	T	-		Baja
12	Salida A	Atenuación relativa			4 bits	C	-	W	-	-		Baja
13	Salida A	Valor de luminosidad			1 Byte	C	-	W	T	-		Baja
32	Salida A	Tipo de carga			1 bit	C	R	-	T	-		Baja
33	Salida A	Mensaje de error			1 bit	C	R	-	T	-		Baja
34	Salida A	Byte de estado			1 Byte	C	R	-	T	-		Baja

Objetos de Grupo | Parámetros | Puesta en marcha

Con este concepto de direcciones de grupo, un sensor puede enviar una orden a múltiples destinatarios (actuadores) y a su vez un actuador puede estar conectado a múltiples direcciones de grupo de diversos sensores.

Es importante notar que los telegramas no se refieren exclusivamente a órdenes de control sino que pueden referirse también a informaciones compartidas entre los nodos. Por ejemplo un sensor de luz puede transmitir el valor medido en lux y este puede recibirse por un controlador de iluminación que decidirá posteriormente el nivel de salida en función del algoritmo que procese.

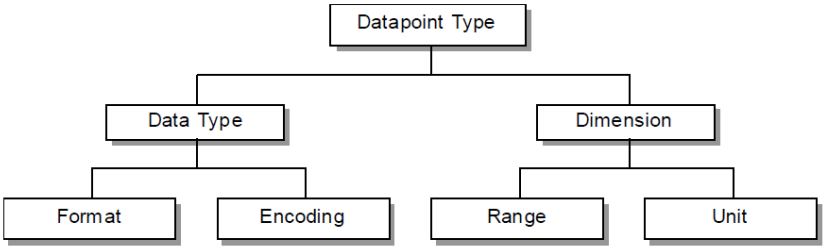
El programa de configuración ETS de KNX no permite reconstruir la estructura de la instalación a partir de los equipos de campo, por lo que una vez programada y configurada la instalación es necesario guardar el proyecto generado para poder realizar modificaciones futuras

2.1.4 TIPOS DE DATOS

Uno de los aspectos más importantes para estandarizar la comunicación es la definición de los tipos de datos que los distintos fabricantes pueden utilizar en los telegramas. Dado que KNX es un sistema multifabricante, se hace necesaria una definición clara y precisa de los tipos de datos y la publicación de los mismos dentro del estándar para que la información sea accesible por los distintos fabricantes.

El estándar KNX dispone del documento 03_07_02 DATAPOINTS TYPES. Este documento forma parte del Volumen 3 System Specifications, Parte 7 Interworking, Capítulo 2 y en él se define el concepto de Datapoint y se describen los tipos de datos aceptados en el estándar.

Los Datapoints se definen como la combinación del tipo de dato y la dimensión, conteniendo las informaciones de formato, codificación, rango y unidad.



Cada Datapoint se representa por un valor de 16 bits seguido por un punto y otro valor de 16 bits. Por ejemplo “7.002”. Los parte izquierda, denominada número principal, representa el formato y la codificación y la parte derecha, denominada subnúmero el rango y unidad. Los Datapoints que tienen el mismo número principal tienen el mismo tipo de dato.

Los Datapoints están estructurados según se indica en la tabla siguiente:

Application Domain	Subnumber	MAIN number			
		0 … 199	200 … 299	300 … 59 999	≥ 60 000
		mainly unstructured	structured		
Common use	**0 … 99**	DPT is • standard • mainly unstructured • common use	DPT is • standardised • structured • common use	reserved for future use	Reserved. These DPT-IDs shall not be used.
HVAC	**100 … 499**	DPT is • standardised • unstructured • HVAC specific use	DPT is • standardised • structured • HVAC LTE only	managed by WGI	
Load Management	**500 … 599**	DPT is • standardised • unstructured • LMM specific usage	DPT is • standardised • structured		
Lighting	**600 … 999**	DPT is • standardised • unstructured • lighting	DPT is • standardised • structured • lighting		
System	**1 000…1 199**	DPT is • standardised • unstructured • **system**	DPT is • standardised • structured • **system**		
Reserved	**1200… 50 999**	reserved for other applications (managed by WGI)			
Manufacturer specific	**≥ 60 000**	manufacturer specific extensions [a]			manufacturer specific extensions [a]

[a] For interpretation of these Datapoint Types the device type needs to be known.

Un problema relativamente importante en KNX es que **la información de los Datapoints no se transmite en los telegramas al bus**.

Para la especificación del formato se utiliza la siguiente notación:

Symbol	Field
A	Character
A[n]	String of n characters
B	Boolean / Bit set
C	Control
E	Exponent
F	Floating point value
M	Mantissa
N	eNumeration
r	Reserved bit or field
S	Sign
U	Unsigned value
V	2's Complement signed value
Z_8	Standardised Status/Command B_8. Encoding as in DPT_StatusGen

Por ejemplo U_{16} representa a un entero sin signo de 16 bits.

Algunos Datapoints pueden usarse de forma general sin restricción y otros están sujetos a limitaciones para su uso. Se utiliza la siguiente sintáxis para definir su uso:

- G general: se puede usar sin restricciones
- FB bloque funcional: solo se puede usar en un bloque funcional
- HVAC, HWH, TU .. dominios de aplicación: solo se pueden usar en el dominio especificado

Para entender la documentación de la especificación consideremos el caso del Datapoint 3.007

3.3.1 DPT_Control_Dimming

Format:	4 bit: B_1U_3
octet nr	1
field names	c | Step-Code
encoding	B | U U U
Range:	c = {0,1} StepCode = [000b…111b]
Unit:	none
Resol.:	(not applicable)
PDT:	PDT_GENERIC_01

Datapoint Types

ID:	Name:	Use:
3.007	DPT_Control_Dimming	FB

Data fields	Description	Encoding
c	Increase or decrease the brightness.	See 1.007 0 = Decrease 1 = Increase
StepCode	The amount of intervals into which the range of 0 % … 100 % is subdivided, or the break indication.	- 001b…111b:Step Number of intervals = $2^{\wedge(\text{stepcode}-1)}$ - 000b: Break

El formato es de 4 bits compuestos por B1U3. Es decir tenemos dos campos dentro del formato que representan a C y StepCode.

C es un campo de tamaño un bit (B_1) rango (0,1) y su significado es incrementar el nivel de brillo con valor 1 y decrementar el nivel de brillo con valor 0.

StepCode representa el tamaño del paso de incremento o decremento del brillo con un valor comprendido entre 0 y 100%. Tiene un tamaño de 3 bits su rango es (000b-111b) y el valor 000b representa break y el resto de valores el tamaño del paso calculado como $2^{\wedge(stepcode-1)}$.

Las unidades no están definidas y el uso del Datapoint es dentro de un bloque funcional.

2.2 PROTOCOLO LONWORKS

Fue creado por la empresa Echelon en 1988 y registrado como estándar ISO/IEC 14908-1 Control Network Protocol. El CNP ISO/IEC 14908-2 y 3 se refiere al nivel físico y el ISO/IEC 14908-4 a la comunicación IP tunneling.

Al igual que KNX, el protocolo es de libre acceso al medio y no es necesario ningún maestro ni ordenador de control para el funcionamiento normal de la red de control una vez que esté configurado el sistema.

Los objetivos perseguidos con la creación de este protocolo abierto son:

- Disponer de un protocolo optimizado para los sistemas de control pero que a la vez sea suficientemente genérico para adaptarse a diferentes tipos de control.
- El coste de la incorporación de este sistema de control a los productos de los fabricantes debe ser razonablemente bajo tanto en el hardware como en el desarrollo del software.
- Debe existir interoperabilidad entre los distintos fabricantes no solo en el protocolo sino también en las funcionalidades de los aparatos de control.

Con objeto de cumplir los requerimientos de bajo coste de fabricación y desarrollo de los aparatos, Echelon se centra en la creación de productos y herramientas para facilitar a los fabricantes la incorporación de este protocolo en sus productos.

Echelon no aborda el mercado de producto final y por tanto no compite con los fabricantes de controladores y sensores sino que les presta soporte con sus productos.

La plataforma Lonworks suministrada por Echelon se compone de Smarts transceivers, routers, interfaces de red, herramientas de programación, herramientas de configuración de red y de depuración y puesta en marcha.

La interoperabilidad se consigue definiendo un diccionario común de datos, de variables de red SNVTs, de propiedades de configuración SCPTs y de plantillas de funcionamiento de los equipos PROFILES.

El mantenimiento de estos estándares y la certificación de los equipos para la obtención del logo de conformidad la realiza la organización LONMARK.

El protocolo Lonworks se puede incorporar fácilmente a nuevos productos a través del NEURON CORE que es un chip diseñado por Echelon y fabricado por sus colaboradores que contiene 4 microcontroladores, memoria y subsistemas de comunicaciones y de I/O.

Dos de los procesadores se encargan de la ejecución de las capas 2 a la 6 del protocolo, un tercer procesador ejecuta la capa 7 de aplicación y un cuarto procesador gestiona el sistema de interrupciones.

Cada chip contiene un identificador único de 48 bits que se denomina NeuronID que conjuntamente con un pin especial que se denomina Pin Service permite la identificación remota del dispositivo para su programación.

Echelon suministra el sistema operativo LNS Server que gestiona una base de datos de configuración de los dispositivos conectados en la red de control y que suministra funciones avanzadas como la de sustitución de equipos. Este sistema operativo permite el acceso al bus de comunicaciones a otras aplicaciones como HMI, SCADAs y aplicaciones de configuración a través de una estructura cliente servidor que permite múltiples clientes conectados. También permite la conexión remota del sistema a través de Internet.

La aplicación LONMAKER permite configurar los aparatos de control de la red, programarlos a través del bus de comunicaciones, realizar funciones de diagnóstico y establecer relaciones entre los distintos dispositivos a través de variables compartidas (variables de red).

La aplicación LONSCANNER es un analizador de protocolos que permite observar, analizar y diagnosticar el comportamiento de la red. Los mensajes se muestran con la fecha y hora y se pueden ordenar y filtrar en base a los distintos campos de los mensajes.

2.2.1 ISO/IEC 14908-1 Control Network Protocol (CNP)

La base del Sistema Lonworks es el protocol ISO/IEC 14908-1. Hay varias implementaciones de este protocolo y la realizada por Echelon se denomina protocolo LONTALK.

El protocolo CNP está diseñado para satisfacer las necesidades de control para un amplio rango de aplicaciones y requerimientos. Para ello se basa en el modelo de referencia OSI implementando los 7 niveles de dicho modelo.

Si bien el CNP describe con detalle en su documento todos los niveles, los usuarios que desarrollen productos con esta tecnología pueden estar interesados solo en algunos de los niveles. En general un usuario que diseñe aplicaciones estará interesado en los niveles 6 y 7 y quizás también en las opciones del nivel 4. Un usuario que diseñe circuitos estará más interesado en el nivel 1.

Las características principales del protocolo CNP son las siguientes:

Envío eficiente de mensajes pequeños. La sobrecarga del protocolo CNP en la transmisión de un mensaje puede ser tan baja como 9 bytes. Además los mensajes se pueden dirigir a un único dispositivo o a un grupo.

Fiabilidad en el envío de los mensajes. CNP realiza el reenvío de los mensajes cuando ocurre un fallo de comunicación y además informa al dispositivo que envía el mensaje si se produce un fallo irrecuperable en la comunicación. Realiza también las funciones de sincronización que permiten que cuando se recupera un dispositivo destinatario de un mensaje, este pueda recibir el mensaje en uno de los reintentos.

Detección de mensajes duplicados. Esto es particularmente interesante cuando un dispositivo utiliza la información del mensaje para incrementar un contador. En este caso la recepción de mensajes duplicados generaría errores en los valores de esos contadores.

Múltiples medios de comunicación. CNP soporta comunicación sobre par trenzado con la posibilidad de que la alimentación de los dispositivos se realice sobre el mismo cable de

comunicaciones. También está permitida la comunicación sobre cable coaxial y sobre línea eléctrica. Los routers se encargan de pasar la comunicación de manera transparente desde un medio a otro.

Bajo coste de los dispositivos. El protocolo CNP está optimizado para que los requerimientos de memoria sean mínimos. Una aplicación básica puede ocupar tan poca memoria como 10 Kbytes de código y 1 Kbyte de RAM.

Bajo coste de instalación y mantenimiento. Solo es necesaria una aplicación (LONMAKER) para la instalación de la red. Esta aplicación no hace falta que esté permanentemente instalada en la red y puede estar en un ordenador portátil para compartirla con la instalación y mantenimiento de muchas redes. Es posible también un modo autónomo de configuración sin programa de instalación para redes pequeñas (Interoperable Self-Installation - ISI). Las funciones de diagnóstico, programación y sustitución de equipos se realizan desde la herramienta de configuración.

Uso eficiente del ancho de banda de comunicación. CNP permite compartir el canal de comunicaciones a muchos dispositivos. Los mecanismos implementados de control de acceso al medio permiten una alta tasa de comunicaciones incluso en condiciones de alta carga del bus. Es posible además compartir redes distintas sobre el mismo canal de comunicaciones separando las redes a través de dominios.

Interoperabilidad. CNP soporta interoperabilidad, permitiendo la instalación de productos de múltiples fabricantes sobre la misma red interaccionando entre ellos. CNP permite también usar las mismas herramientas de instalación y configuración para los productos de los diversos fabricantes.

Prevención de intrusión. CNP permite la transmisión de mensajes con autentificación impidiendo la conexión de usuarios no autorizados.

2.2.2 CAPAS CNP

El protocolo CNP está dividido en capas siguiendo el modelo OSI definido por la INTERNATIONAL ESTANDAR ORGANIZATION. Las siete capas del protocolo están implementadas en el protocolo CNP.

1. La **capa física** es la encargada de transmitir los bits a través del canal de comunicación. Asegura que un bit transmitido desde un dispositivo es recibido por todos los dispositivos destinatarios. Ya que el protocolo CNP es independiente del medio, múltiples protocolos de la capa física están soportados dependiendo del medio de comunicación.
2. La **capa de enlace** define mecanismos de control de acceso al medio y codificación de datos para asegurar un uso eficiente del canal de comunicación. La capa de enlace define los mecanismos para indicar a un dispositivo cuando puede transmitir un mensaje al bus y define también como los receptores reciben esta información y pueden detectar si se ha producido un error en la transmisión. Están definidos también niveles de prioridad que permiten que los mensajes más urgentes se envíen primero al bus.
3. La **capa de red** define los mecanismos por los cuales un mensaje puede ser rutado desde el dispositivo origen a uno o más destinatarios, incluso aunque estén en diferentes canales de comunicación. En esta capa se definen nombres y direcciones de los dispositivos para conseguir el correcto envío de los mensajes.

4. La **capa de transporte** permite el envío seguro de los mensajes. Los mensajes pueden enviarse usando reconocimiento y realizar retransmisiones en caso de que el envío no sea correcto. Esta capa define también como se tratan los mensajes duplicados en el caso de que estos se retransmitan como resultado de la no recepción del reconocimiento.

5. La **capa de sesión** añade control a los mensajes intercambiados por las capas inferiores. Soporta acciones remotas, así que un cliente remoto puede realizar una petición a un servidor y recibir la respuesta. En esta capa se procesa también la autentificación para aquellos casos en los que se dese una comunicación segura .En este caso el receptor de un mensaje puede determinar si el emisor de un mensaje está autorizado a enviarlo.

6. La **capa de presentación** añade estructura a los datos intercambiados por las capas inferiores definiendo la codificación del mensaje. Los mensajes se puede codificar como variables de red, mensajes de aplicación o tramas externas. La interoperabilidad de las variables de red se realiza a través de las SNVTs (standard network variable types) definidas y mantenidas por LONMARK.

7. La **capa de aplicación** define servicios de red estándares que usan los datos intercambiados por las capas inferiores. Estos servicios incluyen la configuración de la red, diagnósticos, transferencia de ficheros, configuración de la aplicación, alarmas, registros de datos y horarios. Estos servicios permiten que dispositivos de distintos fabricantes tengan los mismos mecanismos de configuración y estos puedan realizarse a través de las mismas herramientas.

CNP asegura que los mensajes enviados desde un dispositivo lleguen correctamente a los destinatarios. A través de las distintas capas del protocolo los datos enviados desde el nivel de aplicación se modifican a su paso por cada una de las capas inferiores hasta obtener la hilera de bits que se transmite por la capa uno. Estas modificaciones consisten en la inserción de cabeceras específicas en cada una de las capas. Los dispositivos receptores realizan las operaciones inversas eliminando las cabeceras a su paso por cada capa hasta llegar al nivel de aplicación.

CAPA		**DATO**
1	FÍSICA	HILERA DE BITS
2	ENLACE	TRAMA DE DATOS
3	RED	DATAGRAMA
4	TRANSPORTE	PAQUETE DE TRANSPORTE
5	SESION	PAQUETE DE SESION
6	PRESENTACIÓN	PAQUETE DE PRESENTACIÓN
7	APLICACIÓN	MENSAJE

La figura ilustra la estructura de la trama en el nivel de la capa física.

msb							lsb
Bits de sincronismo (número de unos configurable)							0
Cabecera de enlace (1 byte)							
Cabecera de red (4 a 16 bytes)							
Cabecera de transporte (0 a 1 bytes)							
Cabecera de sesión (0 a 1 bytes)							
Cabecera de presentación (1 a 2 bytes)							
Datos de aplicación (0 a 246 bytes)							
Protección de errores CRC (2 bytes)							

La información se transmite de arriba abajo. En las informaciones multibyte se transmite primero el byte menos significativo. Los bytes se transmiten comenzando con el bit de menor peso.

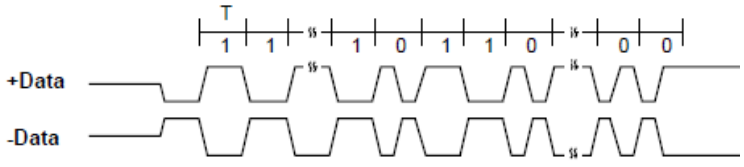
2.2.2.1 Capa 1. Nivel Físico

La capa 1 define la transmisión de las hileras de bits sobre un canal de comunicación. Un canal es un medio de transporte físico para los paquetes de datos. Esta capa asegura que un bit transmitido desde un dispositivo será recibido por los distintos destinatarios.

Como CNP es independiente del medio, distintos niveles físicos están soportados. Dependiendo del medio un dispositivo Lonworks usará un tipo u otro de transceiver. Una red Lonworks puede estar formada por distintos canales enlazados entre sí a través de routers.

Un segmento consiste en uno o varios canales enlazados por repetidores o pasarelas no por routers.

El medio más habitual usado para las comunicaciones es el par trenzado y el transceiver FTT-10A que permite una comunicación a 78Kb/s usando topología libre y uno o dos terminadores de línea. Este medio está estandarizado a través de la norma ISO/IEC 14908-2. La comunicación se realiza en Manchester diferencial y sobre un canal se pueden conectar 64 dispositivos. La longitud máxima del canal es de 500m para sistemas con un terminador y 2700m con dos terminadores de red. Utilizando routers podemos interconectar otros canales y ampliar la red a más de los 64 dispositivos permitidos en un canal.



2.2.2.2 Capa 2.Nivel de enlace

El nivel físico define los mecanismos de control de acceso al medio y la codificación de los datos para permitir un uso eficiente del canal de comunicación.

Las hileras de bits se descomponen en tramas y el nivel de enlace define cuando un dispositivo puede enviarla a través del canal de comunicación y como un receptor puede recibir la información y determinar si está libre de errores.

Cuando un dispositivo quiere transmitir una información debe esperar primero a que el canal esté libre. Una vez libre, cada dispositivo espera un tiempo aleatorio antes de comenzar la transmisión. Si un dispositivo detecta la comunicación de otro dispositivo debe comenzar el proceso.

CNP permite también la comunicación con prioridad en cuyo caso la transmisión comienza antes de los tiempos aleatorios. Se permiten hasta 127 niveles de prioridad y solo un dispositivo puede tener un nivel determinado de prioridad. El nivel de prioridad se asigna a través de las herramientas de configuración de la red.

Una característica importante del CNP es que el número de intervalos de tiempo aleatorios es dinámico y se incrementa con la carga del bus.

CNP usa un mecanismo de control de acceso al medio que es una variante del sistema CSMA (carrier sense multiple access), que se denomina “predictive p-persistent CSMA” y que permite a un canal con sobrecarga de comunicaciones trabajar cerca de su máxima capacidad.

El nivel de enlace añade a los datos a transmitir un byte de sincronismo, una cabecera de enlace y una protección de errores CRC16.

2.2.2.3 Capa 3. Nivel de red

La capa de red define como los mensajes son rutados desde el dispositivo origen a los distintos destinatarios del mensaje. En esta capa se definen nombres y direcciones para garantizar la correcta transmisión de la información.

Cada dispositivo Lonworks dispone de un identificador llamado Neuron ID de 48 bits que es único para cada dispositivo.

Una dirección es un identificador que identifica a un objeto o un grupo de objetos dentro de una clase y que puede ser modificado asignado y modificado en cualquier momento.

La capa de red define en un mensaje las direcciones del dispositivo que envía el mensaje y de los dispositivos que deben recibirlo.

Un Neuron ID puede usarse como dirección destino de un mensaje pero no es aconsejable ya que solo permitiría un mensaje par a par y además el número de butes necesario para expresar la dirección es muy elevado. Se suele usar solo en los mensajes de instalación y configuración de los nodos.

CNP define una estructura de direcciones basada en tres campos: dominio, subred y nodo. Esta estructura de direccionamiento permite identificar solo a un nodo o a un conjunto de ellos. Por ejemplo podemos direccionar a todos los nodos de una subred expresando en la dirección solo el dominio y la subred.

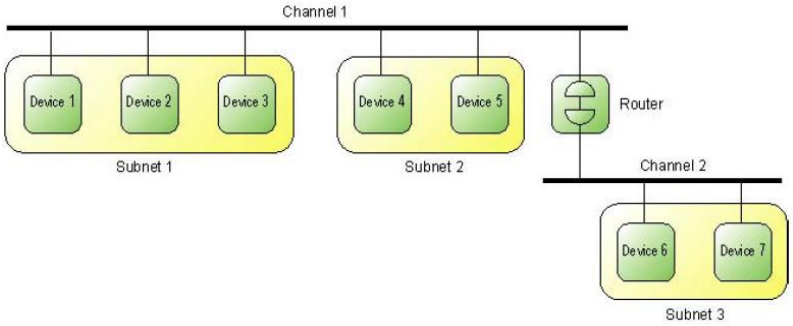
Dominio

Un dominio es una colección de dispositivos conectados en uno o varios canales. Constituye una red virtual ya que solo se permiten las comunicaciones entre dispositivos conectados en el mismo dominio.

Un dominio se expresa usando de 1 a 6 bytes por lo que el número total de dominios permitidos es de 2^{48}. Ya que la información del dominio se incluye en las comunicaciones es conveniente usar dominios lo más cortos posibles. Un byte puede ser una buena opción.

Subred

Una subred es una colección lógica de hasta 127 dispositivos dentro de un dominio. Se pueden definir hasta 255 subredes dentro de un dominio. Todos los dispositivos de una subred deben estar sobre el mismo segmento. Un router crea un nuevo segmento y por tanto una nueva subred.



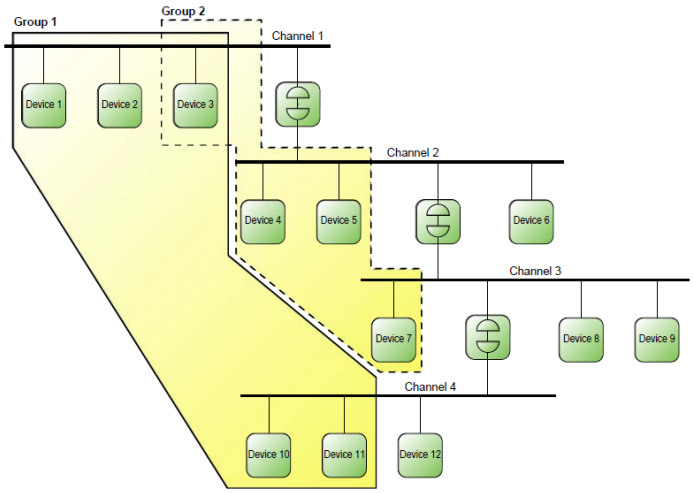
Un dispositivo puede pertenecer a dos dominios y actuar como pasarela entre los dos dominios. Debe ser asignado a una subred en cada dominio.

Un dominio puede contener hasta 255 subredes x 127 nodos = 32385 dispositivos. Un dispositivo está unívocamente representado por su dominio, subred y nodo.

Grupos

Los grupos son una colección de dispositivos dentro de un dominio que pueden pertenecer a distintas subredes. Un dominio permite definir hasta 255 grupos y un dispositivo creado con el Neuron firmware puede pertenecer a 15 grupos.

Los grupos son importantes porque permiten direccionar de una forma sencilla a un número ilimitado de dispositivos como destinatarios de un mensaje.



Formatos de direccionamiento

CNP usa en sus mensajes 5 tipos de direccionamiento usando los conceptos de dominio, subred, nodo, grupo y Neuron ID.

El número de bytes necesario para identificar el origen y destino de un mensaje depende del tipo de direccionamiento utilizado.

Address Mode	*Address Format*	*Destination*	*Address Size (bytes)*
Domain-wide Broadcast	Domain (Subnet = 0)	All devices in the domain	3
Subnet-wide Broadcast	Domain, Subnet	All devices in the subnet	3
Multicast	Domain, Group	All devices in the group	3
Unicast	Domain, Subnet, Node	Specific device within a subnet	4
Neuron ID	Domain, Neuron ID	Specific device	9

Cada dispositivo CNP dispone de una tabla de direcciones que puede ser manejada por las herramientas de configuración de red. Estas direcciones pueden ser utilizadas por la aplicación para identificar un destinatario del mensaje a través de un índice de la tabla. Este método de direccionamiento se denomina implícito. Los grupos también pueden estar incluidos en esta tabla de direcciones. Neuron firmware permite 15 entradas en la tabla de direcciones de un dispositivo.

Por el contrario el direccionamiento que precisa indicar el destino del mensaje a través de uno de los formatos indicados anteriormente se denomina explícito.

2.2.2.4 Capa 4. Capa de transporte

La capa de transporte añade fiabilidad en el envío de los mensajes. Los mensajes pueden ser enviados usando un servicio con reconocimiento. El dispositivo transmisor espera un tiempo la recepción de un reconocimiento y reenvía el mensaje si no lo recibe.

La capa de transporte gestiona también los mensajes duplicados y especifica como rechazar estos mensajes en caso de que se hubiera recibido el mensaje original.

La capa de transporte define también el envío de mensajes sin reconocimiento y mensajes repetidos y de petición/respuesta.

Los mensajes sin reconocimiento se usan cuando un dispositivo envía regularmente una información, por ejemplo una temperatura. Si se produce la pérdida del mensaje el receptor puede recibir de nuevo la información en el siguiente envío y por tanto no es crítica la pérdida del mensaje.

Un mensaje repetido se utiliza cuando el número de destinatarios del mensaje es alto y se necesitarían muchos reconocimientos para completar la transacción. En estos casos el envío repetido, por ejemplo 3 veces, del mensaje hace que la probabilidad de recepción del mensaje sea igualmente alta y se reduce el uso del canal de comunicación.

Cada uno de estos tipos de mensajes puede usar todos los direccionamiento de la capa 3: unicast, multicast, broadcast y Neuron ID.

Cuando se envía un mensaje broadcast con reconocimiento, el transmiso lo dá por finalizado a la recepción del primer reconocimiento. El resto de los reconocimientos son ignorados.

2.2.2.5 Capa 5. Capa de sesión

La capa de sesión soporta las acciones remotas, de forma que un cliente puede solicitar un dato a un dispositivo y recibir la respuesta.

Esta capa también define autentificación para prevenir accesos no autorizados a un dispositivo.

Los servicios de petición/respuesta se utilizan cuando un dispositivo envía una petición a uno o varios dispositivos y necesita recibir una respuesta de cada uno de ellos. El mensaje se procesa de manera similar a los mensajes con reconocimiento con la diferencia de que se debe recibir una respuesta de cada uno de los destinatarios en lugar de un reconocimiento. Los destinarios se pueden especificar usando todos los formatos de direccionamiento descritos en la capa 3. Si se envía un mensaje broadcast, solo se procesa la recepción de la primera respuesta, el resto se ignoran.

Cuando se envían mensajes autentificados el destinatario del mensaje determina si el dispositivo que envía el mensaje está autorizado para ello. La autentificación se realiza distribuyendo una clave de 48 bits por dominio durante el proceso de instalación. Para que un mensaje autentificado sea validado el transmisor y receptor deben tener la misma clave. Cuando un mensaje autentificado es recibido, el receptor solicita al transmisor su identificación enviándole un número aleatorio distinto cada vez. El transmisor debe devolver el número aleatorio codificado junto con los datos del mensaje. El receptor lo decodifica y si es correcto procesa el mensaje.

2.2.2.6 Capa 6. Capa de presentación

La capa de presentación añade estructura a los datos definiendo la codificación de los datos del mensaje. Cada mensaje consta de 1 byte como código del mensaje seguido de hasta 227 bytes de datos, salvo para el caso de variables de red en cuyo caso siguen de 1 a 31 bytes de datos.

Un mensaje puede ser codificado como una variable de red, un mensaje de aplicación o un mensaje externo.

Los mensajes de aplicación se utilizan cuando se necesita una codificación distinta a las variables de red o cuando hace falta enviar más de 31 bytes de datos. Mensajes de autoconfiguración, registro de datos y transferencia de ficheros son ejemplos de mensajes de aplicación.

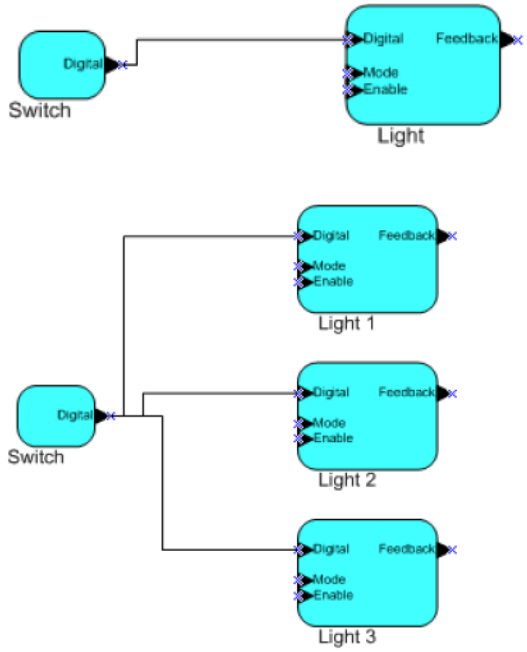
Los mensajes externos son grupos de bytes que no tienen significado para CNP y pueden ser por ejemplo informaciones en otro protocolo enviadas sobre el mismo canal.

Las aplicaciones utilizan habitualmente las variables de red para el intercambio de información. Este mecanismo permite compartir las informaciones entre múltiples dispositivos de la red. La interoperabilidad se consigue a través de la definición de las SNVTs (standard network variable types) que están recogidas en un fichero de recursos gestionados por la organización LONMARK INTERNATIONAL. Las variables de red pueden ser también arrays, de manera que cada elemento del array puede conectarse independientemente a otras variables de red.

Cada variable de red contiene una dirección (input, output) un tipo de dato (definido en el fichero de recursos) y una longitud. Variables de red del mismo tipo y longitud y direcciones opuestas puede conectarse entre sí para intercambiar información.

Esta conexión la realizan las herramientas de configuración de red (por ejemplo Lonmaker) y después de que el programa de aplicación esté creado. Es decir cuando se crea la aplicación no se conoce que dispositivos estarán conectados a sus variables de entrada y salida. Esta conexión se realizará en la fase de instalación de la red de control a través de un proceso que se denomina "binding". Una variable de red de salida se muestra en la herramienta de configuración como un triángulo (flecha) hacia afuera y una de entrada con el triángulo hacia dentro.

La conexión de una variable de red de salida se puede hacer a una (unicast) o varias (multicast) variables de entrada del mismo o de distintos aparatos.



Las variables de red suministran un protocolo de aplicación orientado al dato. Los datos de la aplicación (temperaturas, presiones, etc.) se intercambian entra los dispositivos a través de variables que contienen las mismas unidades de ingeniería. Cada dispositivo que recibe una variable de red la procesa de manera diferente a través de su propio programa de aplicación.

El concepto de variable re red simplifica notablemente el concepto de comunicaciones en el nivel de aplicación, ya que no se gestionan buffers de recepción, direcciones ni otros elementos sino que directamente se manipulan las variables de red como cualquier otra variable de programa.

CNP permite 4096 variables de red por dispositivo, aunque el neuron firmware soporta un máximo de 254. Cada variable de red se especifica dentro del dispositivo como un índice comenzando por cero, independientemente de que la variable sea de entrada o salida.

El direccionamiento de las variables de red en la capa 6 se realiza a través de un selector de variables de red de 14 bits. Cada dispositivo mantiene una tabla de selectores de variables de red que se denomina tabla de configuración de variables de red. En esta tabla los selectores están indexados al índice de la variable de red.

Cada variable de red contiene información del tipo de datos, unidades, rango y resolución. Están definidas actualmente unas 100 variables de red. Las definiciones se encuentran en un diccionario de datos de recursos de red disponible en types.lonmark.org.

Es posible editar este diccionario e incluir variables de red propietarias a expensas de la perdida de interoperabilidad que eso supone. Los tipos de datos asociados a las variables de red pueden ser escalares, estructuras y uniones. Los tipos escalares permitidos son de tipo flotante de simple y doble precisión, datos enumerados de 8 bits, campos de bits de 1 a 8 bits, datos con y sin signo de 8, 16 y 32 bits.

Ejemplo de SNVT

Amperage in alternating current: SNVT_amp_ac

"Amperage in alternating current in amperes AC"

Index	Size	Data Type	Minimum	Maximum	Scaling	Resolution
139	2 bytes	unsigned long	0	65535	$1\times10^{0}\times(Raw+0)$	1 amperes AC

2.2.2.7 Capa 7. Capa de aplicación

La capa de aplicación gestiona los servicios de configuración y diagnósticos de la red así como servicios específicos de la capa de aplicación.

El **servicio de configuración** de la red suministra un conjunto estándar de comandos para configurar los atributos de los dispositivos incluyendo las variables de red y sus enlaces.

El **servicio de diagnóstico** consta de un conjunto de comandos que pueden usar las herramientas de instalación de red para diagnosticar los problemas de la red o de los dispositivos.

Los **servicios de transferencia de ficheros** permiten transferir ficheros a través de la red con tamaños de hasta 2GBytes. La transferencia se realiza a través del protocolo Lonworks FTP que usa transferencia de bloques de 32 bytes.

El **servicio de configuración** de la aplicación permite configurar el comportamiento de un dispositivo a través de propiedades de configuración. Estas propiedades están estandarizadas de manera similar a las SNVTs con un diccionario de propiedades estándares de red SCPTs.

Los **servicios de diagnósticos** de la aplicación permiten comprobar el funcionamiento de los bloques funcionales de los dispositivos.

Los **servicios de gestión** de la aplicación permiten gestionar el funcionamiento de los bloques habilitando, deshabilitando y reseteando su funcionamiento.

Los **servicios de alarmas** suministran un interface estándar para que un dispositivo pueda informar sobre las alarmas producidas.

Los **servicios de registro de datos** suministran un interface estándar para coleccionar datos en ficheros log que después puedan ser enviados a un servidor.

Los **servicios de calendario** permiten definir eventos a ejecutar en fechas y horas establecidas.

Los **servicios de gestión de fecha y hora** permiten la sincronización de la fecha y hora de todos los dispositivos conectados a la red.

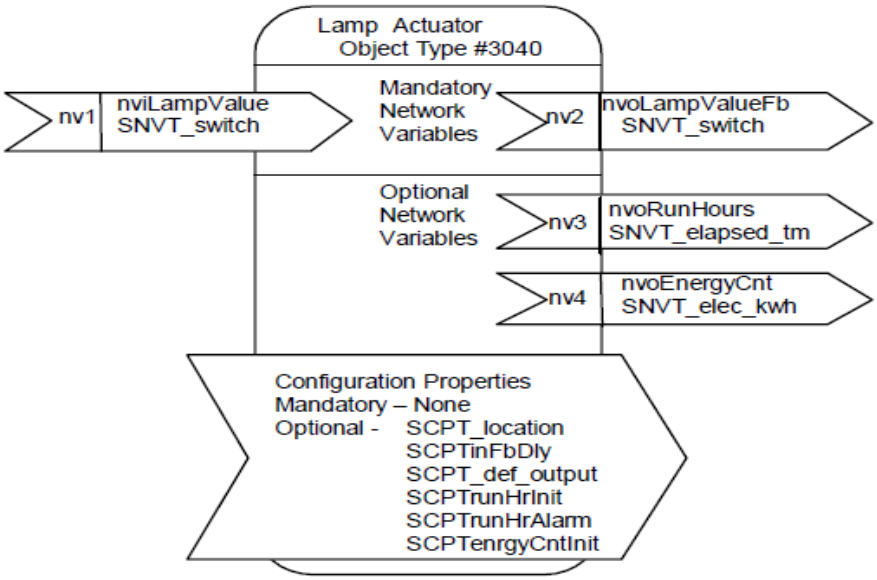
CNP suministra un conjunto de interfaces para que un dispositivo pueda documentar las tareas que realiza. Cada tarea se especifica como un FUNCTIONAL PROFILE que está definido como un conjunto de variables de red y de configuración contenidas dentro del bloque funcional. La interface del dispositivo llamada XIF puede estar incluida dentro del dispositivo y en un fichero externo de extensión XIF. La interface del dispositivo está identificada de manera unívoca por un identificador llamado PROGRAM ID.

Una aplicación de un dispositivo CNP está dividida en bloques funcionales. Un bloque funcional es una porción del código de la aplicación que ejecuta una tarea recibiendo datos de entrada de configuración y de operación, procesando los datos y enviando valores a las salidas operacionales. Un bloque funcional puede recibir datos desde la red, sus I/O y de otros bloques funcionales dentro del mismo dispositivo.

Cada bloque funcional está definido por un perfil funcional que es una plantilla de funcionamiento de un bloque. Los perfiles funcionales están definidos en los ficheros de recursos de Lonmark y están estandarizados para obtener la interoperabilidad entre fabricantes.

Cada perfil funcional tiene un nombre y número que lo identifica. Contiene variables de red de entrada y salida obligatorias, variables de red opcionales y específicas de fabricante. También contiene propiedades de configuración que pueden ser obligatorias, opcionales y específicas del fabricante.

Ejemplo de perfil funcional FP-3040 actuador de lámparas



Cada perfil funcional está descrito en un documento donde además se explica el comportamiento funcional del bloque de aplicación del dispositivo. De esta forma dos dispositivos de distintos fabricantes que implementen en sus dispositivos el mismo perfil funcional serán funcionalmente compatibles, es decir tendrán el mismo comportamiento y sus salidas los mismos valores para los mismos valores de entrada.

El PROGRAM ID es un identificador de 64 bits que identifica de manera unívoca a la aplicación contenida en un dispositivo. Se representa por ocho pares de dígitos hexadecimales separados por ':'. Cuando está formateado en la forma de un Program ID, contiene los valores de 6 campos según el formato FM:MM:MM:CC:CC:UU:TT:NN.

F es un identificador de formato de 4 bits que toma el valor 8 para productos certificados por Lonmark y el valor 9 para productos no certificados.

M es el código de 20 bits de fabricante suministrado por Lonmark. Si un fabricante no es miembro de Lonmark puede solicitar un código provisional.

C representa el identificador de 16 bits de la clase de dispositivo. Indica la función principal del dispositivo que normalmente coincide con la del perfil funcional implementado. También puede contener el código de una clase de dispositivo estándar.

U es un identificador de 8 bits que indica el uso del equipo. Se puede usar uno de los valores estándares preestablecidos.

T es un identificador de 8 bits que indica el tipo de canal soportado por el dispositivo.

M es un identificador de 8 bits que utiliza el fabricante del dispositivo para que el Program ID sea diferente para los distintos productos fabricados por él.

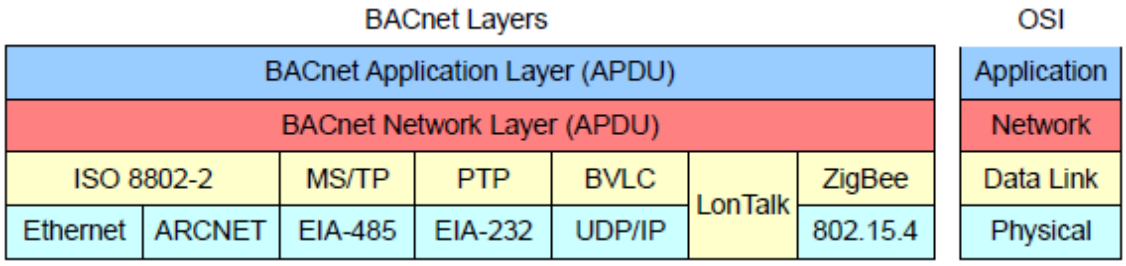
2.3 BACNET

El protocolo Bacnet fue desarrollado por la asociación ASHRAE. Los trabajos comienzan en 1987 y en 1995 se certifica como ANSI 135-1995. Posteriormente en 2003 se obtiene la certificación ISO 16484-5.

El protocolo está basado en el modelo OSI y se utiliza una arquitectura colapsada de 4 capas que se corresponde con las capas: física, de enlace, de red y de aplicación.

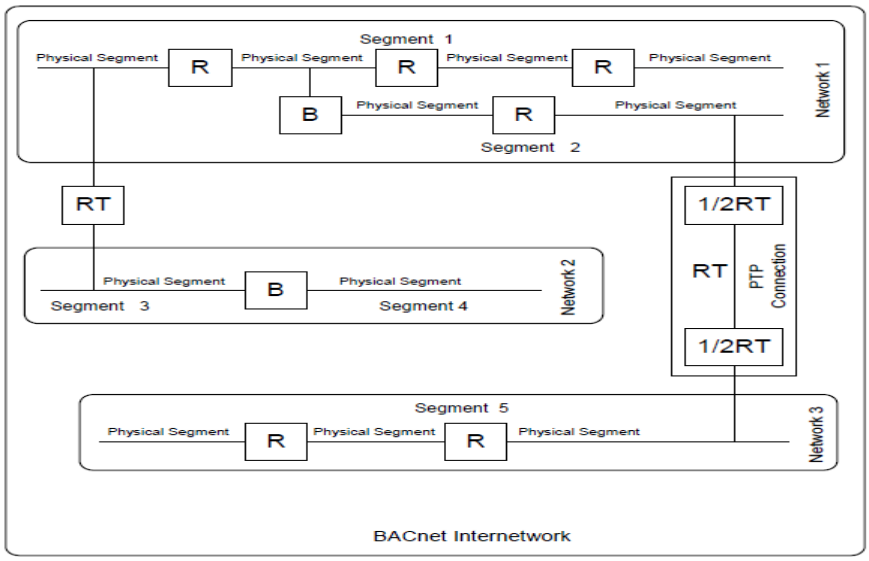
Existen 2 variantes del protocolo: Bacnet IP que comunica sobre UDP y Bacnet MSTP que comunica sobre RS485. Bacnet está concebido fundamentalmente como un protocolo para el nivel más elevado de comunicación en los sistemas de control de edificios y no para el nivel de campo, aunque Bacnet MSTP realiza esa tarea.

Siete opciones están definidas para la capa física y de enlace que permiten la utilización de protocolos existentes o del protocolo MSTP sobre RS485. Como vemos, la definición del estándar recoge la opción de LON y de Zigbee para los niveles de campo.



En términos de topología, cada dispositivo está conectado a un segmento físico. Un segmento Bacnet consiste en uno o varios interconectados por repetidores. Una red Bacnet consiste en uno o

varios segmentos interconectados por bridges que permiten conectar segmentos con distintas capas físicas y de enlace. Múltiples redes se interconectan a través de routers. En Bacnet existe una única ruta entre dos dispositivos para el envío de un mensaje.



B = Bridge
R = Repeater
RT = Router
1/2RT = Half Router

La capa de transporte es la responsable de garantizar la comunicación par a par entre dos dispositivos, la segmentación, el flujo de control y la recuperación de errores. Gran parte de estas funciones son similares a las de la capa de enlace. Dado que Bacnet soporta distintas redes en los niveles inferiores, el protocolo debe suministrar los servicios para a par y la recuperación de errores. Esto se hace en la capa de aplicación a través de retransmisiones y temporizaciones. La segmentación es también necesaria debido a que algunos servicios del protocolo permiten el envío de grandes cantidades de información. La segmentación la realiza también la capa de aplicación. Las secuencias de control para reensamblar la información segmentada se realiza así mismo en el nivel de aplicación en los procedimientos de segmentación.

La capa de sesión es utilizada para establecer y manejar largos diálogos entre los participantes en la comunicación. La mayoría de transacciones en Bacnet son cortas y no precisan los servicios de la capa de sesión por lo que está capa no está implementada para no penalizar la mayor parte de las transacciones.

La capa de presentación suministra un procedimiento para negociar la sintaxis a utilizar en la transmisión de los mensajes. Si solo se permite una sintaxis las funciones de la capa de presentación se reducen a un esquema de codificación para representar los datos de aplicación. Bacnet usa un sistema de codificación fijo que se incluye en la capa de aplicación, haciendo innecesaria la capa de presentación.

Las funciones suministradas por la capa de red incluyen la traslación de direcciones globales a locales, el rutado de mensajes a través de varias redes, resolver las diferencias de los distintos niveles inferiores en cuanto al tamaño de los mensajes, secuenciación, control de flujo, control de errores etc.

Para obviar las funciones de segmentación y reensamblado, Bacnet pone la restricción a las comunicaciones con los routers de que la máxima longitud de los mensajes debe ser la máxima

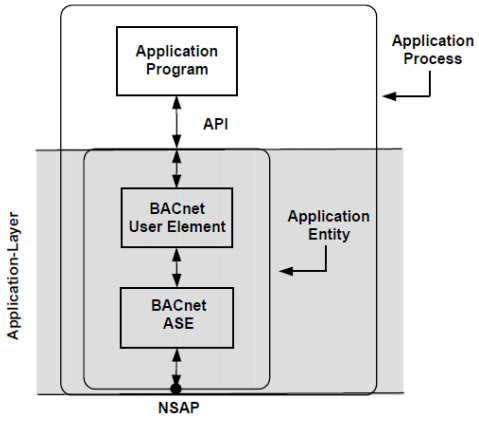
soportada en el nivel de enlace por cada capa de enlace. De esta forma el tamaño máximo del mensaje dependerá de la capa de enlace utilizada.

Tecnología de enlace de datos	Máxima longitud NPDU
ISO 8802-3 Ethernet, definido en cláusula 7	1497 bytes
ARCNET, definido en cláusula 8	501 bytes
MS/TP definido en cláusula 9	501 bytes
Punto a punto definido en cláusula 10	501 bytes
LonTalk definido en cláusula 11	228 bytes
Bacnet/IP definido en anexo J	1497 bytes
Zigbee definido en anexo O	501 bytes

El segundo byte de la trama en la capa de red es un byte de control que especifica los formatos de las direcciones origen y destino y el MSB con valor 1 especifica que la trama contiene un mensaje de red y no de datos. Se habilita de esta forma un campo de tipo de mensaje que especifica el mensaje concreto de red que se transmite. Los tipos de mensajes permitidos son:

X'00': Who-Is-Router-To-Network
X'01': I-Am-Router-To-Network
X'02': I-Could-Be-Router-To-Network
X'03': Reject-Message-To-Network
X'04': Router-Busy-To-Network
X'05': Router-Available-To-Network
X'06': Initialize-Routing-Table
X'07': Initialize-Routing-Table-Ack
X'08': Establish-Connection-To-Network
X'09': Disconnect-Connection-To-Network
X'0A': Challenge-Request
X'0B': Security-Payload
X'0C': Security-Response
X'0D': Request-Key-Update
X'0E': Update-Key-Set
X'0F': Update-Distribution-Key
X'10': Request-Master-Key
X'11': Set-Master-Key
X'12': What-Is-Network-Number
X'13': Network-Number-Is
X'14' to X'7F': Reserved for use by ASHRAE
X'80' to X'FF': Available for vendor proprietary messages

La capa de aplicación suministra los servicios de comunicación requeridos por la aplicación. Un proceso de aplicación es la funcionalidad dentro de un sistema que ejecuta el procesado de la información necesario para esa aplicación concreta. Parte de este procesado no depende de las comunicaciones y es por tanto externo a Bacnet. La parte de la aplicación que está dentro de la capa de aplicación se denomina entidad de aplicación. La comunicación entre el programa de aplicación y la entidad de aplicación se realiza a través de una API que no está definida en el documento de protocolo de Bacnet.



La entidad de aplicación está dividida en dos partes: Bacnet User Element y Bacnet Application Service Element (ASE).

Bacnet user element se ocupa de la interface con el programa de aplicación (API). Es además responsable de mantener información de cada transacción incluyendo sus identificadores, el tipo de servicio necesario, temporizadores etc.

BAcnet ASE representa el conjunto de funciones y servicios de aplicación para las alarmas y eventos, servicios de acceso a ficheros, servicios de acceso a objetos, servicios de gestión remota del dispositivo y servicios de terminal virtual.

Desde un punto de vista funcional, Bacnet es un protocolo creado para la automatización de los sistemas de climatización en edificios usando el concepto de interoperabilidad, pero no excluye el control de otros tipos de instalaciones como los sistemas contraincendios, seguridad, iluminación, ascensores etc.

2.3.1 Dispositivos Bacnet

Un dispositivo Bacnet está formado en hardware por los elementos sensores o actuadores específicos del dispositivo y por un microcontrolador que realiza las tareas de comunicación y de funcionamiento propio del dispositivo.

Un dispositivo puede ser un controlador, una pasarela o un dispositivo de interfaz de usuario. Cada dispositivo contiene un objeto dispositivo que contiene informaciones del dispositivo incluyendo sus identificador e instancia. La instancia del dispositivo debe ser única en la red. Además cada dispositivo contiene una colección de informaciones acerca de las entradas y salidas del dispositivo que monitoriza y controla.

2.3.2 Objetos

Toda la información de un dispositivo es modelada en términos de objetos de información, que representan algún componente importante del dispositivo o alguna información importante del dispositivo que puede ser de interés a otro dispositivo Bacnet. Los objetos pueden representar datos individuales o múltiples y pueden representar informaciones reales o virtuales del dispositivo.

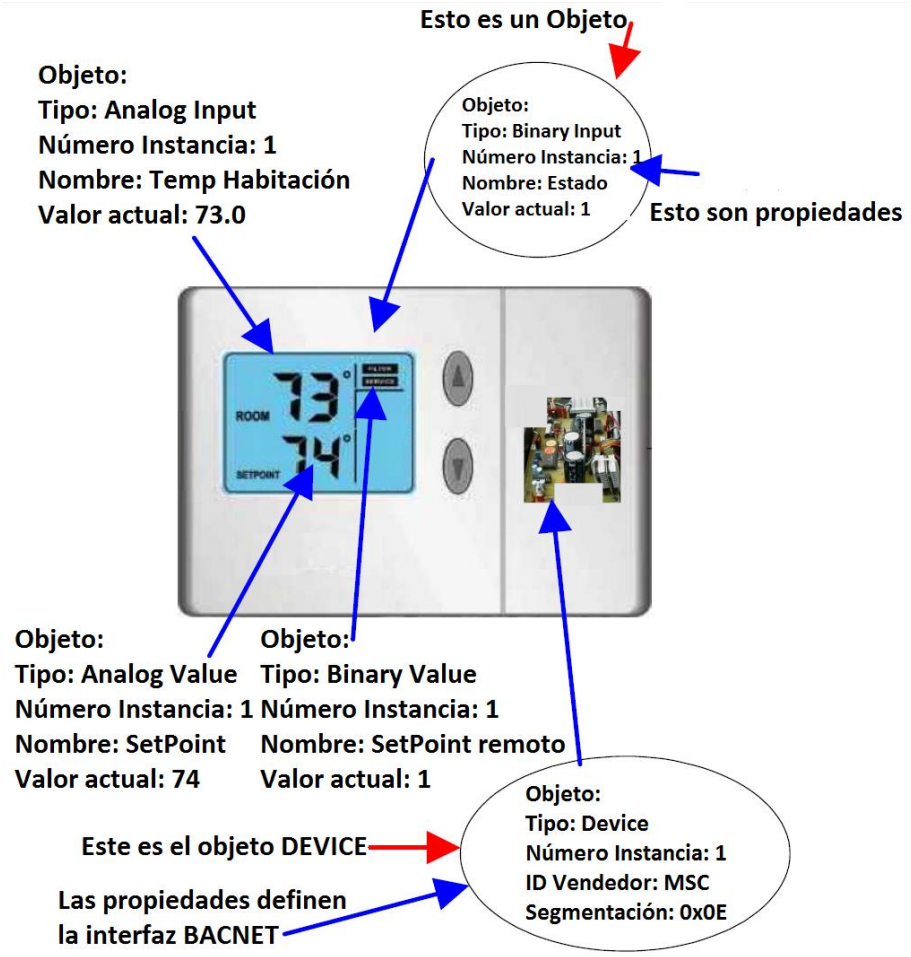
Bacnet define 54 tipos de objetos distintos estandarizados. La implementación de un dispositivo puede hacer uso de una combinación de estos objetos para representar informaciones del dispositivo y de su lógica de control. Bacnet permite también la definición de objetos no estandarizados como

definibles por el usuario. Cada objeto se identifica a través de código de 32 bits y contiene una colección de propiedades que definen al objeto.

2.3.3 *Propiedades*

Una propiedad contiene una información acerca de un objeto Bacnet y consta de un identificador y un valor. El identificador es un valor numérico único para la propiedad dentro del objeto. Cada propiedad puede ser de solo lectura o de lectura-escritura.

El propósito de una propiedad es permitir a otros dispositivos acceder a la información del objeto para leerla y modificar su valor si está permitido. Algunas propiedades de los objetos son obligatorias para los objetos del mismo tipo y otras son opcionales. Las tablas de descripción de las propiedades de los objetos contienen una columna que especifica esta cualidad.



2.3.4 *Servicios*

Los servicios son requerimientos que un dispositivo Bacnet envía a otro para intercambiar información o realizar alguna acción.

Se agrupan en 5 categorías de funcionalidad:

- Acceso a los objetos: leer, escribir, crear, borrar

- Gestión de los dispositivos: descubrir, sincronización temporal, inicializar, copia de seguridad y restauración de datos
- Alarmas y eventos: alarmas y cambios de estado
- Transferencia de ficheros: datos históricos y transferencia del programa
- Terminal virtual: interface hombre-máquina a través de menús

Los servicios definen la forma de la petición y los datos necesarios para obtener la respuesta.

2.3.5 Áreas de interoperabilidad

Usando los conceptos de dispositivos, objetos, propiedades y servicios, Bacnet proporciona capacidades funcionales denominadas "*Áreas de Interoperabilidad*". Hay 5 áreas de interoperabilidad:

- Datos compartidos
- Históricos
- Planificación
- Alarmas y eventos
- Gestión de dispositivos y red

Datos compartidos es el intercambio de información entre dispositivos. Se aplica el concepto cliente servidor para identificar donde se encuentran las informaciones. Las peticiones que un cliente puede realizar a un servidor son ReadProperty y WritePropety.

Cuando es necesario intercambiar muchas informaciones entre los dispositivos, se usa un mecanismo denominado Change Of Value (COV) que evita tener que solicitar las informaciones una a una y hace que el servidor envíe solamente los valores que han sufrido cambio, según los umbrales definidos, desde la última petición.

Históricos es un procedimiento que permite a los dispositivos almacenar datos históricos que otro dispositivo puede recoger a través del uso de ReadProperty y COV. Este proceso es más eficiente que el de centralización de los datos de históricos en el servidor permitiendo que los datos históricos estén distribuidos entre los dispositivos. Si los dispositivos permiten COV se usará este procedimiento para la comunicación de los datos, y ReadProperty en caso contrario.

Algunos tipos de datos históricos están disponibles solo para los objetos del mismo dispositivo y se denominan históricos internos, otros están disponibles para el resto de dispositivos y se denominan históricos externos.

La planificación permite definir acciones de control para una fecha y hora determinadas. Los objetos calendario indican los días hábiles para las acciones. Se pueden especificar como días concretos o intervalos de fechas y pueden especificar también días de la semana y excepciones. La idea de los objetos de planificación es la de valores de tiempo entendidos como parejas (tiempo, valor).

La gestión de alarmas y eventos define el intercambio de información en base a límites de alarma predefinidos y a eventos. El disparo de estas alarmas y eventos puede requerir la intervención de un operador para aceptar su reconocimiento. El proceso de detección se denomina intrínseco cuando la detección del suceso se realiza en el propio objeto y algorítmica cuando se debe ir solicitando los valores de otro objeto y aplicar un procedimiento o reglas para detectar el suceso. Para la

especificación de los sucesos, Bacnet usa una enumeración que se denomina *event state* con los valores NORMAL, OFFNORMAL y FAULT.

La gestión de dispositivos y de la red, permite a Bacnet descubrir los dispositivos de la red, sus objetos y sus propiedades, restablecer las comunicaciones, reiniciar un dispositivo y cambiar su aplicación. También permite el *binding* que es proceso para establecer conexiones entre dispositivos. Este proceso se puede realizar de una forma estática o dinámica en base a la instancia del objeto o incluso a través del nombre del objeto.

Con objeto de dar a conocer la interoperabilidad implementada en un dispositivo, el fabricante debe suministrar un fichero de información denominado PICS (Protocol Implementation and Conformance Statement) que especifica en detalle las áreas de interoperabilidad implementadas en el dispositivo.

Un elemento fundamental en los PICS son los BIBBs (Bacnet Interoperability Buildings Blocks) que definen conjuntos y grupos de funcionalidades que pueden ser fácilmente comparadas entre dispositivos para determinar qué características Bacnet son interoperables entre ellos.

Los BIBBS están agrupados en las siguientes categorías:

1. Data Sharing
    a. Read/write property
    b. Read/write multiple properties
    c. COV (Change of Value)
    d. Unsubscribed COV
2. Trending
    a. Viewing and modifying trends – internal
    b. Viewing and modifying trends – external
    c. Automated trend retrieval
3. Scheduling
    a. Scheduling – internal
    b. Scheduling - external
4. Alarm and Event Management
    a. Alarm and event notification – internal
    b. Alarm and event notification – external
    c. Alarm acknowledgement
    d. Life safety alarm
5. Device Management
    a. Device binding - discovery and connection
    b. Object binding - discovery and connection
    c. Device communication control
    d. Private transfer of message
    e. Text message

f. Time synchronization
g. UTC time synchronization
h. Reinitialize device and restart notification
i. Backup and restore device database
j. List manipulation
k. Object creation and deletion
l. Virtual terminal

6. Network Management
   a. Device connection establishment
   b. Router configuration

Cada BIBB está definido con una letra A o B que identifica la implementación en el cliente o servidor. La letra A indica lado del Cliente que es el que inicia la función, la letra B indica Servidor que es quien ejecuta la función. De esta forma si en los PICS aparece por ejemplo DS-RP-A indica que este dispositivo implementa DataSharing con la propiedad ReadProperty como un cliente.

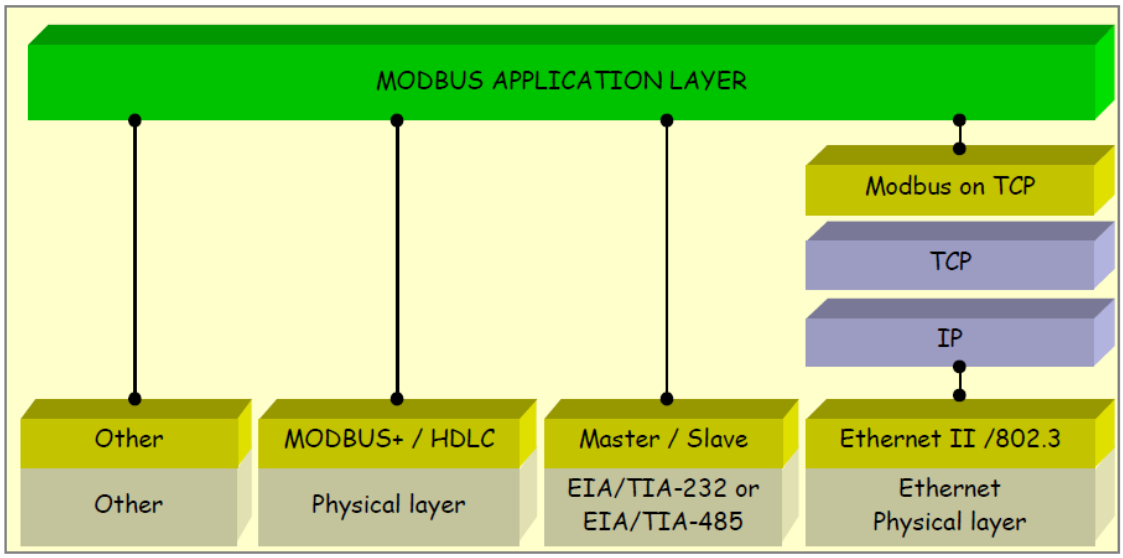
En el ANEXO 1 se muestra un ejemplo de fichero PIC de especificación de un equipo Bacnet.

2.4 MODBUS

El protocolo Modbus fue creado en 1979 por la empresa MODICON actualmente perteneciente al grupo Schneider Electric.

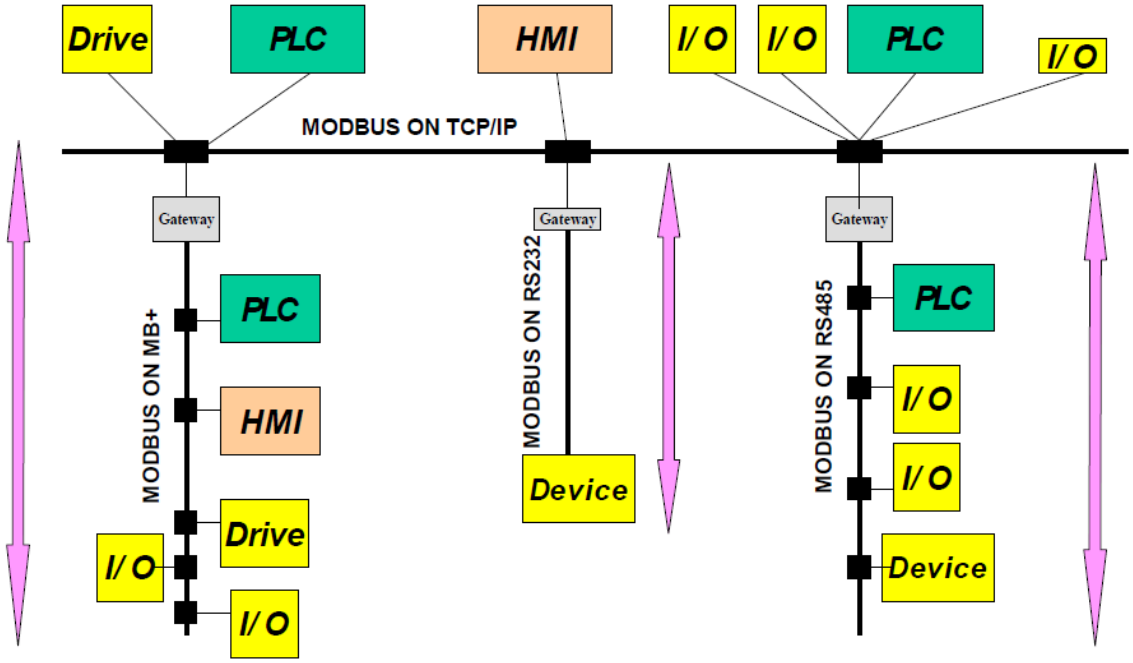
2.4.1 ESTRUCTURA DEL PROTOCOLO

Es un protocolo Maestro-Esclavo implementado sobre comunicaciones serie RS232 y RS485 con el nombre de Modbus RTU y sobre redes IP con el nombre de Modbus TCP/IP.

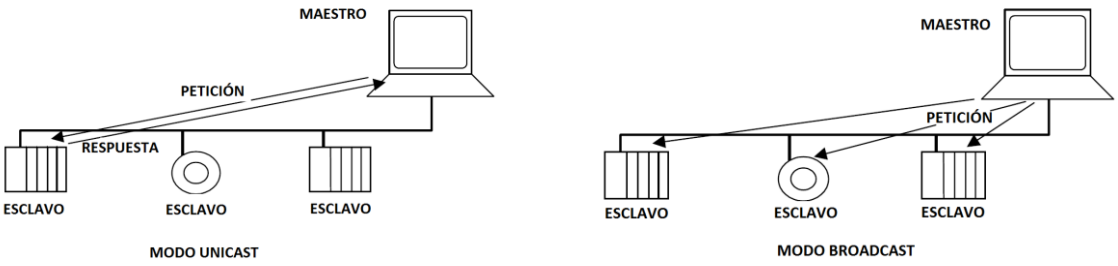


Es un protocolo muy popular debido a que no precisa Hardware especial, está bien documentado y es muy eficiente en dispositivos que tienen muchos puntos de control.

A través de un canal RS485 se pueden conectar hasta 247 dispositivos cada uno de ellos identificado por una dirección de 8 bits. Si se precisan más dispositivos se puede usar un maestro con múltiples canales RS485 o bien usar Gateway IP/RS485.

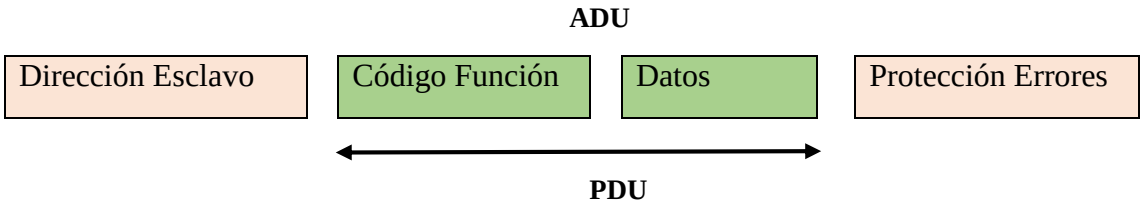


Las comunicaciones se realizan normalmente en unicast, siempre iniciadas desde el maestro y dirigidas a un solo esclavo.



También son posibles los mensajes dirigidos a todos los esclavos (broadcast) a través de la dirección 0. Estos comandos deben ser forzosamente de escritura. Ningún esclavo debe responder a un mensaje con la dirección 0.

Las comunicaciones en el protocolo Modbus se establecen en el nivel de aplicación a través del ADU (application data unit). La parte del mensaje PDU (protocol data unit) es común a todos los medios y. el ADU contiene campos específicos a RTU y a IP. La protección de errores para el protocolo Modbus RTU es CRC16. Se describe el algoritmo de cálculo en [5].



El código de función tiene el tamaño de 8 bits de los cuales los 7 bits de menor peso representan el código de función (0-127) y el bit msb con valor 1 se utiliza como una bandera para indicar errores.

Cada código de función indica el acceso a un conjunto distinto de datos del esclavo. Algunas funciones usan también un código de subfunción para aumentar las posibilidades de la función.

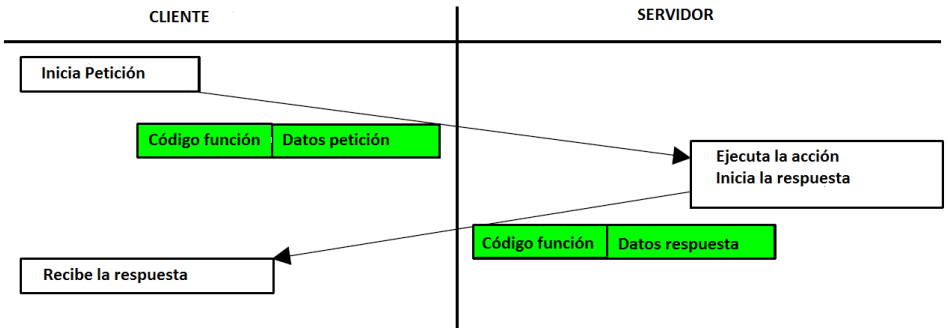
El campo de datos depende de si la comunicación es petición o respuesta y del código de función utilizado.

El tamaño máximo de los mensajes es de 256 bytes sobre RS485. Ya que La dirección ocupa 1 byte y el campo de protección de errores ocupa 2 bytes nos quedan 253 bytes útiles para la PDU. Sobre IP la PDU tiene un tamaño máximo de 249 bytes.

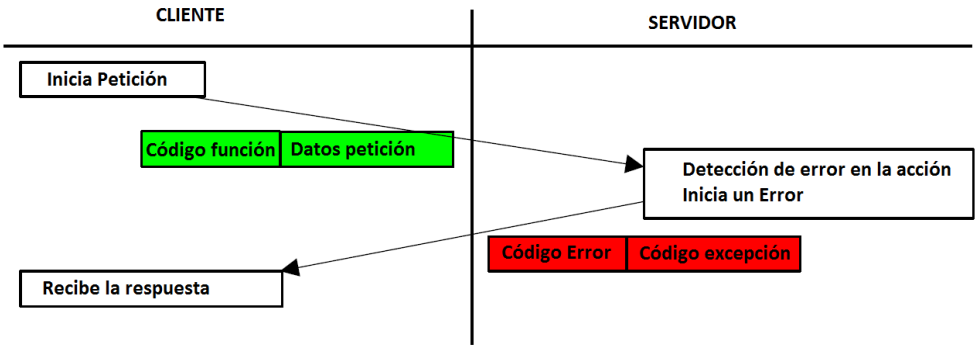
El protocolo usa tres tipos distintos de PDU:

- Modbus Request PDU, mb_req_pdu
- Modbus Response PDU, mb_rsp_pdu
- Modbus Exception Response PDU, mb_excep_rsp_pdu

En el funcionamiento normal se realiza una petición por el maestro (cliente) mb_req_pdu y el esclavo (servidor) envía la respuesta mb_rsp_pdu. El código de función en la respuesta es el mismo que el usado en la petición.



Si se produce un error en la petición del maestro, el esclavo responde con mb_excep_rsp_pdu usando el mismo código de función pero marcando el bit msb a 1 para indicar la excepción y se devuelve un código de error descriptivo del error detectado.



En la transmisión de datos mayores de 1 byte se usa el formato big-Endian enviando primero el byte de mayor peso MSB. Este criterio es válido para el campo de datos pero no para la protección de errores en la que se usa un CRC16 enviando primero el byte de menor de peso LSB.

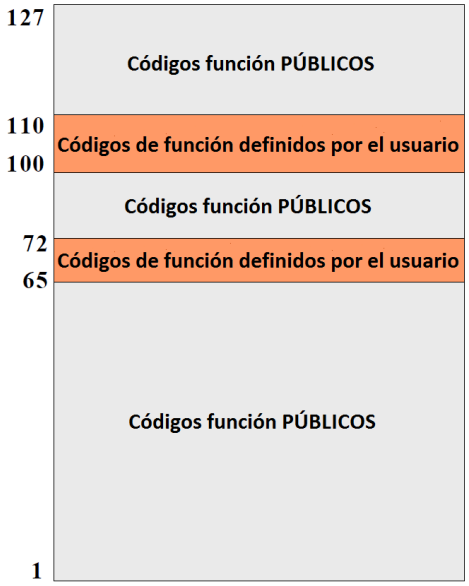
El modelo de datos en Modbus está basado en las cuatro tablas de datos primarios siguientes:

Tablas primarias	Tipo de Objeto	Tipo de acceso	Comentarios
Discretes input	bit	Solo lectura	Suministrado por un sistema I/O
Coils	bit	Lectura-Escritura	Puede ser modificado por un programa de aplicación
Input Registers	Palabra 16 bits	Solo lectura	Suministrado por un sistema I/O
Holding Registers	Palabra 16 bits	Lectura-Escritura	Puede ser modificado por un programa de aplicación

Solo están soportados datos de 1 bit y de 16 bits. Cada tabla usa 16 bits para direccionar a sus elementos con lo que tenemos 63536 elementos en cada tabla. El direccionamiento de los elementos de las tablas se hace con los valores 0 a 65535.

Hay que prestar atención a que algunos maestros numeran los registros con los valores 1 a 65536 por lo que hay que realizar el desplazamiento de una posición al construir los PDUs correspondientes.

Los 127 posibles códigos de función se dividen en públicos y definidos por el usuario.



Solo algunos códigos de funciones públicas están definidos en el estándar, quedando el resto libres para ampliaciones futuras.

Las funciones públicas definidas en el documento del estándar se muestran en la tabla que figura a continuación.

Cada esclavo implementará solo las funciones que le corresponden en función de los tipos de entradas y salidas disponibles y del criterio del fabricante.

				Function Codes	
				code	*Sub code*
Data Access	**Bit access**	Physical Discrete Inputs	Read Input Discrete	**02**	
		Internal Bits Or Physical coils	Read Coils	**01**	
			Write Single Coil	**05**	
			Write Multiple Coils	**15**	
	16 bits access	Physical Input Registers	Read Input Register	**04**	
		Internal Registers Or Physical Output Registers	Read Multiple Registers	**03**	
			Write Single Register	**06**	
			Write Multiple Registers	**16**	
			Read/Write Multiple Registers	**23**	
			Mask Write Register	**22**	
	File record access		Read File record	**20**	**6**
			Write File record	**21**	**6**
Encapsulated Interface			Read Device Identification	**43**	**14**

Como vemos en la tabla, las funciones se dividen en Data Access que permiten el acceso a los datos y Encapsulated Interface que permite leer la identificación del dispositivo.

La descripción de cada función precisa especificar el formato de la petición, la respuesta y las excepciones. En el ANEXO 2 se detallan todas las funciones del protocolo.

2.4.2 PROTOCOLO MODBUS TCP/IP

El protocolo Modbus TCP/IP se utiliza para comunicar los dispositivos a través de una red Ethernet usando la estructura cliente/servidor TCP/IP.

Los esclavos son los servidores TCP/IP y el maestro se conecta como cliente TCP/IP. Por defecto se utiliza el puerto 502.

Un dispositivo Modbus TCP/IP puede ser un controlador con sus entradas y salidas o una Gateway que permita conectar a la red IP un conjunto de controladores Modbus RTU sobre RS485.

Aunque el PDU de la comunicación es el mismo que se ha descrito anteriormente, hay dos cambios en el ADU. Por un lado se suprime la protección de errores ya que la trama IP tiene sus propias protecciones y además se sustituye el campo inicial de dirección del esclavo por una cabecera MBAP HEADER de 7 bytes de longitud.

CAMPOS	LONG	DESCRIPCIÓN	CLIENTE	SERVIDOR
Identificador transacción	2 bytes	Identificador de una transacción Modbus de petición/respuesta	Iniciado por el cliente	Copiado por el servidor desde la petición recibida
Identificador protocolo	2 bytes	0=protocolo MODBUS	Iniciado por el cliente	Copiado por el servidor desde la petición recibida
Longitud	2 bytes	Número de bytes que siguen	Iniciado por el cliente (petición)	Iniciado por el servidor (respuesta)
Identificador de la unidad	1 byte	Identificación de un esclavo remoto conectado a una línea serie	Iniciado por el cliente	Copiado por el servidor desde la petición recibida

El campo Identificador de la transacción se utiliza para emparejar los mensajes de petición y respuesta. El maestro usa este campo para identificar la petición y el esclavo responde con el mismo identificador. Habitualmente este campo se usa como un contador y el maestro incrementa su valor después de cada petición.

El campo Identificar del protocolo tiene el valor fijo 0 que representa que es el protocolo Modbus TCP/IP. Otros valores pueden representar nuevas versiones del protocolo.

El campo Longitud contiene el valor del número de bytes siguientes hasta el fin de la trama. Contiene al campo identificador de la unidad y la PDU.

El campo Identificador de la unidad se usa por las Gateways para indicar la dirección del esclavo dentro de su segmento secundario (RS485) con el que se quiere comunicar.

Los códigos de función son los mismos descritos anteriormente para el protocolo Modbus RTU.

2.4.3 COMUNICACIÓN MULTIMAESTRO

La estructura básica de comunicación Modbus es una estructura Maestro-Esclavo con un único maestro sobre un canal de comunicación RS485. Las estructuras multimaestro usando mecanismos de paso de testigo o de reparto del tiempo bus entre los maestros no están definidas en el protocolo.

Estos procesos son complejos ya que si un maestro usa un paso de testigo a otro maestro y este a su vez a un tercer maestro, la cadena de paso de testigo entre maestros debe ser consistente. Esto quiere

decir que si un maestro se desconecta, los otros maestros tienen que ser capaces de reconocer el problema y reconstruir dinámicamente la cadena de maestros para el paso de testigo. Es necesario limitar también el tiempo de intervención de cada maestro para permitir la intervención de todos los maestros con un reparto de tiempo ecualizado a las necesidades de cada maestro.

Todas estas cuestiones hacen que el sistema se convierta en complejo y además el rendimiento del bus se resentirá enormemente obteniéndose unas tasas efectivas de comunicación mucho más bajas.

Si los maestros son externos a la subred, la estructura multimaestro puede tener soluciones más razonables. Si consideramos una topología de subredes en RS485 cada una de ellas con una pasarela Modbus RTU a Modbus TCP/IP, las pasarelas pueden realizar la gestión de las peticiones de varios maestros desde el lado TCP encolando las peticiones a la subred RS485. Cada maestro Modbus se conecta como cliente a la pasarela, que al ser multicliente gestiona las peticiones y respuestas al bus.

3 Normalización de los tipos de datos MODBUS

3 NORMALIZACIÓN DE LOS TIPOS DE DATOS MODBUS

Modbus no define tipos de datos, solo datos de 1 bit (COILS y HR) y datos de 16 bits (inputs registers y holding registers).

3.1 EL PROBLEMA DE LOS TIPOS DE DATOS MODBUS

Se deja en manos de cada desarrollador de aplicaciones la capacidad de definir e interpretar los datos tal como son (bits y registros de 16 bits) o bien usando múltiplos de estos tipos básicos para manejar datos más complejos con formatos y tamaños definidos por los usuarios.

Puesto que uno de los aspectos fundamentales en los sistemas de control es la interoperabilidad, es de vital importancia normalizar los tipos de datos con la condición de incluir las funcionalidades actuales de los equipos comerciales. Esto nos permitirá por una parte gestionar los equipos actuales y además disponer de una norma, hasta ahora inexistente en Modbus, para la representación de los datos.

Veamos algunos ejemplos de tipos de datos utilizados en los equipos comerciales.

ANALIZADOR DE REDES CVM-C10 DE CIRCUTOR

Parámetros de medida:

Parámetro	Símbolo	Instantáneo	Máximo	Mínimo	Unidades
Tensión fase L1	V 1	00-01	106-107	164-165	V x 10
Corriente L1	A 1	02-03	108-109	166-167	mA
Potencia Activa L1	kW 1	04-05	10A-10B	168-169	W
Potencia Inductiva L1	kvarL1	06-07	10C-10D	16A-16B	var
Potencia Capacitiva L1	kvarC1	08-09	10E-10F	16C-16D	var
Potencia Aparente L1	kVA 1	0A-0B	110-111	16E-16F	VA

Factor de potencia L1	PF 1	0C-0D	112-113	170-171	x 100
Cos φ L1	Cos φ 1	0E-0F	114-115	172-173	x 100

Salidas digitales:

Variable	Dirección	Valor por defecto
Estado de las salidas digitales	4E21	-

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	**0**	**0**	**0**	**Salida 4** **0:** OFF **1:** ON	**Salida 3** **0:** OFF **1:** ON	**Salida 2** **0:** OFF **1:** ON	**Salida 1** **0:** OFF **1:** ON

Datos de configuración:

Máxima demanda			
Variable de configuración	**Dirección**	**Margen válido de datos**	**Valor por defecto**
Periodo de integración	274C	**1 - 60** minutos	15

Órdenes (funciones de bit):

Parámetros	Dirección	Margen válido de datos
Borrado de energías	834	**FF00**
Borrado de máximos y mínimos	838	**FF00**
Inicialización de la máxima demanda	839	**FF00**
Borrado de los contadores de horas (Todas las tarifas)	83D	**FF00**
Borrado del valor máximo de la máxima demanda	83F	**FF00**
Borrado de energías, máxima demanda y máximos y mínimos	848	**FF00**

El dispositivo CVM-C10 es un analizador de redes trifásico con comunicaciones Modbus RTU. Las informaciones de medida de los parámetros están disponibles en registros de 16 bits accesibles a través de las funciones 3 y 4 para lectura y por la función 16 para escritura.

Se utilizan 32 bits para cada parámetro, ocupando 2 registros de 16 bits consecutivos. Como vemos la tabla documenta el nombre del parámetro, su símbolo y sus unidades.

Si observamos la columna de unidades, podemos ver que los parámetros Tensión de Fase L1, Factor de Potencia L1 y Cos φ L1, están afectados por un factor multiplicativo x10 para la tensión de fase y x100 para los otros dos parámetros.

El procedimiento seguido por este fabricante, que por otra parte es bastante frecuente, ha sido el de suministrar un dato con cifras decimales (1 para la Tensión de Fase y 2 para el Factor de Potencia y Cos φ) como un entero. Para ello se ha encapsulado el dato en formato entero largo multiplicándolo por un factor constante y se transmite entonces como un entero largo y no como un dato en punto flotante.

Evidentemente esto obliga al maestro, que es quien realiza la petición de los datos, a conocer de antemano los factores asociados para restituir el dato recibido a su valor original realizando la función inversa a la realizada por el esclavo.

En el bloque de salidas digitales, se ha utilizado el registro de 16 bits 4E21 para contener la información individual del estado de las salidas digitales. Esto obliga al maestro a realizar las operaciones correspondientes de bit sobre el dato recibido de 16 bits. Algunas veces, los fabricantes repiten las informaciones binarias en los mapas de 1 bits y en los de registros de 16 bits.

Los registros de bit y de 16 bits se pueden usar también para modificar el comportamiento del dispositivo en base a la configuración de sus parámetros. El registro de 16 bits 274C se utiliza en este caso para programar el valor del periodo de integración. El registro de 1 bit 834 se utiliza para el borrado de las energías cuando se escribe el valor ON 0xFF00.

CONTADOR DE ENERGÍA SERIE iEM3100 SCHENEIDER

Utiliza la función 3 para la lectura de registros y la función 16 para la escritura.

Los tipos de información contenidos en los registros de 16 bits están representados en la tabla siguiente:

Tipo	Descripción
Uint16	Entero sin signo de 16 bits
Int16	Entero con signo de 16 bits
Uint32	Entero sin signo de 32 bits (2 regs 16 bits)
UTF8	Campo de 8 bits codificación de caracteres multibyte para Unicode
Int64	Entero con signo 64 bits (4 regs 16 bits)
Float32	Valor de 32 bits Representación estándar IEEE para números flotantes (de precisión simple) (2 regs 16 bits)
DateTime	Estructura fecha y hora (4 regs 16 bits)
Bitmap	Reg 16 bits con significado por bits

Estructura DateTime:

<table>
<tr><th rowspan="2">Palabra</th><th colspan="16">Bits</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>2</td><td colspan="8">Reservado(0)</td><td>0</td><td colspan="7">Año(0..127)</td></tr>
<tr><td>2</td><td colspan="5">0</td><td colspan="3">Mes(1..12)</td><td colspan="4">WD(0)</td><td colspan="4">Día(1..31)</td></tr>
<tr><td>3</td><td>SU</td><td colspan="3">0</td><td colspan="4">Hora(0..23)</td><td>IV</td><td>0</td><td colspan="6">Minuto(0..59)</td></tr>
<tr><td>4</td><td colspan="15">Milisegundo(0..59999)</td></tr>
</table>

Ejemplos del mapa de registros:

Dirección	Tamaño	Tipo	Unidades	Descripción
3000	2	Float32	A	Corriente Fase 1
3204	4	Int64	Wh	Exportación energía activa total
3252	4	DateTime	-	Fecha y hora reinicio de energía
45001	1	Bitmap	-	Alarma sobrecarga: 0x0000 desactivada, 0x0100 activada

Vemos en esta implementación del protocolo el uso de tipos de datos habituales como enteros con o sin signo de distinta longitud, datos float de simple precisión, datos de fecha y hora y nuevamente datos de bits en registros de 16 bits que reciben aquí el nombre de Bitmaps. No se especifica en el manual el orden de los bytes en los datos que ocupan más de 1 registro.

3.2 PROPUESTA DE TIPOS DE DATOS PARA MODBUS

En esta fase de definición de los tipos de datos, el objetivo marcado es la concreción de los tipos base para contener las informaciones, de manera similar a los subtipos de KNX o los tipos de datos de LON o los tipos IEC.

Debemos incluir enteros de distintas longitudes con y sin signo, datos no enteros, cadenas de caracteres etc. Todo ello adaptado al estándar Modbus y a nuevas implementaciones.

Los datos enteros se clasifican en primer lugar por su tamaño (8, 16, 32 y 64 bits). Para cada longitud, los datos pueden ser con signo y sin signo y en función del orden de los bytes en B (Big Endian) para indicar que la parte alta se representa primero (como en el estándar Modbus) o L (Little Endian) para indicar que la parte baja se representa primero (como en IEC).

Consideremos los siguientes datos de 16, 32 y 64 bits representados en hexadecimal y sus correspondientes bytes numerados desde 1 al 8:

16 bits: 0xA1B3 B_1=0xA1 B_2=0xB3

32 bits: 0xA1B3457C B_1=0xA1 B_2=0xB3 B_3=0x45 B_4=0x7C

64 bits: 0xA1B3457C125A824D B_1=0xA1 B_2=0xB3 B_3=0x45 B_4=0x7C B_5=0x12 B_6=0x5A B_7=0x82 B_8=0x4D

El orden de los bytes en las representaciones Big-endian y Little-endian es:

Big-endian

16 bits: B_1B_2

32 bits: $B_1B_2B_3B_4$

64 bits: $B_1B_2B_3B_4B_5B_6B_7B_8$

Little-endian

16 bits: B_2B_1

32 bits: $B_4B_3B_2B_1$

64 bits: $B_8B_7B_6B_5B_4B_3B_2B_1$

Se considera también la representación BCD con sus correspondientes representaciones B y L.

ID	Tipo Dato	Descripción
1	Bool	Dato booleano. Para su uso en COILS y entradas binarias. Rango {0,1}
2	UInt8	Entero 8 bits sin signo
3	Int16B	Entero con signo de 16 bits. ***big-endian***
4	UInt16B	Entero sin signo de 16 bits. ***big-endian***
5	Int16L	Entero con signo de 16 bits. ***little-endian***
6	UInt16L	Entero sin signo de 16 bits. ***little-endian***
7	UInt16DB	Entero BCD de 16 bits (0..9999). ***big-endian***
8	UInt16DL	Entero BCD de 16 bits (0..9999). ***little-endian***
9	Int32B	Entero con signo de 32 bits. ***big-endian***
20	UInt32B	Entero sin signo de 32 bits. ***big-endian***
11	Int32L	Entero con signo de 32 bits. ***little-endian***
12	UInt32L	Entero sin signo de 32 bits. ***little-endian***.
13	UInt32DB	Entero BCD de 16 bits (0..99999999). ***big-endian***
14	UInt32DL	Entero BCD de 16 bits (0..99999999). ***little-endian***
15	Int64B	Entero con signo de 64 bits. ***big-endian***
16	UInt64B	Entero sin signo de 64 bits. ***big-endian***.
17	Int64L	Entero con signo de 64 bits. ***little-endian***
18	UInt64L	Entero sin signo de 64 bits. ***little-endian***.
19	UInt64DB	Entero BCD de 64 bits (0..9999999999999999). ***big-endian***
20	UInt64DL	Entero BCD de 64 bits (0..9999999999999999). ***little-endian***
21	Float32B	Float IEEE754 simple precision. ***big-endian***
22	Float32L	Float IEEE754 simple precision. ***little-endian***
23	String	Cadena de caracteres. Cada carácter ocupa 8 bits. Necesita un número par de caracteres. Las cadenas se considerarán terminadas en NULL.
24	Fecha	Dato fecha de 32 bits. Cada dato tiene un tamaño de 8 bits. Al valor del año hay que sumarle 2000. El formato es año, mes y día, día de la semana.
25	Hora	Dato hora de 32 bits. Cada dato tiene un tamaño de 8 bits. El formato es hora, minuto, segundo. El campo hora ocupa la parte alta del primer

		registro de 16 bits. La parte baja del segundo registro de 16 bits no se usa.
26	Fecha y Hora	Longitud de 48 bits (3 registros de 16 bits). Los campos son todos de 8 bits y se representan en el formato año, mes, día, hora minuto, segundo. Año ocupa la parte alta del primer registro de 16 bits y a su valor hay que sumarle 2000.
27	Intervalo Fechas	Dato 48 bits. Usa 24 bits para la fecha inicial y 24 bits para la fecha final. Cada fecha usa 1 byte para el año, 1 byte para el mes y 1 byte para el día

Detalle formato Fecha

CAMPO	DESCRIPCIÓN	POSICIÓN
Año	Año - 2000	Byte alto primer registro 16 bits
Mes	Mes 1-12	Byte bajo primer registro 16 bits
Día	Día 1-31	Byte alto segundo registro 16 bits
Día semana	1 a 7	D=1,L=2,M=3,X=4,J=5,V=6,S=7

Detalle formato Hora

CAMPO	DESCRIPCIÓN	POSICIÓN
Hora	Hora 0-23	Byte alto primer registro 16 bits
Minuto	Minuto 0-59	Byte bajo primer registro 16 bits
Segundo	Segundo 0-59	Byte alto segundo registro 16 bits
No usado		

Detalle Fecha y Hora

CAMPO	DESCRIPCIÓN	POSICIÓN
Año	Año - 2000	Byte alto primer registro 16 bits
Mes	Mes 1-12	Byte bajo primer registro 16 bits
Día	Día 1-31	Byte alto segundo registro 16 bits
Hora	Hora 0-23	Byte bajo segundo registro 16 bits
Minuto	Minuto 0-59	Byte alto tercer registro 16 bits
Segundo	Segundo 0-59	Byte bajo tercer registro 16 bits

Detalle Intervalo de fechas

CAMPO	DESCRIPCIÓN	POSICIÓN
Año Inicial	Año - 2000	Byte alto primer registro 16 bits
Mes Inicial	Mes 1-12	Byte bajo primer registro 16 bits
Día Inicial	Día 1-31	Byte alto segundo registro 16 bits
Año Final	Año - 2000	Byte bajo segundo registro 16 bits
Mes Final	Mes 1-12	Byte alto tercer registro 16 bits
Día Final	Día 1-31	Byte bajo tercer registro 16 bits

El identificador del tipo de datos es un valor entre 1 y 27 que se puede representar en un campo de 8 bits. No obstante vamos a considerar que además de los datos simples se pueden usar arrays de estos tipos de datos.

Para poder definir un dato tipo array seguiremos el siguiente procedimiento:

- Usaremos un campo de 16 bits para el tipo de datos
- Los 8 bits de menor peso se corresponden con el tipo de dato definido anteriormente
- Los 8 bits de mayor peso son la dimensión del array.
- Si el dato es simple los 8 bits de mayor peso tiene valor 0.

En los datos enteros de 32 y 64 bits solo se consideran las representaciones Big-endian y Little-endian pero no las representaciones mixtas entre ellas.

Las informaciones binarias en registros de 16 bits o más, están perfectamente representadas con estos tipos de datos. No obstante se deberá conocer la distribución de los bits para poder extraer las informaciones contenidas en dichos registros.

Los tipos de datos definidos anteriormente no precisan de información adicional en los registros de datos y por tanto son compatibles con las implementaciones realizadas en todos los aparatos Modbus. Solo indican al maestro la forma en que están contenidas las informaciones en los registros.

Al igual que ocurre en KNX, Bacnet y LON, el tipo de dato no siempre suministra toda el conocimiento necesario para gestionar la información. Por ejemplo si el dato está afectado por un factor multiplicativo o por una relación lineal deberemos conocer esos datos para obtener los valores correctos.

Otras informaciones como las unidades, los rangos de las variables, umbrales de alarma etc. pueden ser complementarios y de gran utilidad para la gestión y visualización de las informaciones y deben definirse adicionalmente a los valores de los propios datos.

4 CONFIGURACIÓN DE UNA RED MODBUS

4 CONFIGURACIÓN DE UNA RED MODBUS

4.1 CONOCIMIENTO DE LOS ESCLAVOS INSTALADOS EN LA RED

Los equipos Modbus necesitan una dirección de valor 1 a 247 para comunicarse con el maestro. En algunos dispositivos esta dirección está cableada por hardware a través de puentes o microinterruptores. En otros dispositivos la dirección se establece a través de programas de configuración usando puertos auxiliares como USB para esta comunicación.

En redes de más de 247 dispositivos se utilizan varios puertos de comunicación serie en el maestro, cada uno de ellos con capacidad para controlar otros 247 dispositivos o se usan pasarelas IP que a través del protocolo Modbus TCP/IP permiten crear redes de cualquier tamaño.

Cada grupo de 247 dispositivos controlados por un puerto RS485 o una pasarela TCP/IP forman una subred. Los dispositivos de dos subredes no están conectados entre sí y cada dispositivo no puede ver las comunicaciones de la otra subred. Las comunicaciones y las configuraciones de las redes Modbus deben hacerse por tanto de manera independiente por cada subred.

Normalmente el maestro dispone de un fichero de configuración o base de datos donde están identificados los esclavos conectados en cada subred. Es posible comprobar la coherencia de esta base de datos con la instalación de campo a través de un mecanismo de búsqueda de los esclavos con dirección configurada.

En Modbus estándar, la única forma de conocer los dispositivos conectados a un maestro es realizar una petición a cada dirección Modbus de 1 a 247. Si la petición a una dirección tiene respuesta, significa que el dispositivo existe. Si no se produce respuesta, el dispositivo no existe o no está disponible en ese momento.

La petición se puede realizar con cualquier función Modbus, ya que si el dispositivo dispone de la información solicitada (por ejemplo COILS) se producirá una respuesta correcta y en caso contrario se producirá una respuesta de excepción. En cualquiera de los dos casos anteriores se confirmará la existencia del esclavo con esa dirección.

Si la red está mal configurada y existe más de un esclavo con la misma dirección, se producirá probablemente una colisión y la respuesta no se recibirá correctamente en el maestro. Sin embargo el maestro puede determinar que se ha producido la colisión ya que se producirá actividad en el puerto de comunicaciones en el que posiblemente se reciba algún byte válido y se producirán además errores de trama y de paridad. Este conocimiento nos permitirá crear soluciones para resolver el problema de la dualidad de direcciones. Por ejemplo una solución sencilla puede ser enviar a los esclavos con la dirección repetida un mensaje para borrar su dirección y proceder posteriormente a asignarles una nueva dirección a los esclavos que lo necesiten.

El tiempo mínimo necesario para realizar esta tarea de identificación de los esclavos presentes en el bus es función de la velocidad de comunicación.

Si consideramos una velocidad de 9600 b/s y una petición de la función 1 (estado de COILS) para 1 a 8 COILS, el tiempo usado para la petición del maestro es de 13 ms donde hemos considerado el tiempo total de 126,5 bits correspondientes a 8 caracteres de la petición y 3.5 t_{car} para la detección de fin de trama. Cada carácter tiene un total de 11 bits tal como define el estándar Modbus.

El tiempo necesario para la respuesta es de 11ms que se corresponden con los 104,5 bits del mensaje de respuesta (6 caracteres de respuesta + 3.5 t_{car}).

El tiempo mínimo total entre la petición y respuesta es de 24ms que supone un total de 5928 ms para el total de los 247 esclavos.

Es cierto que este tiempo puede aumentar, ya que en el caso de que un esclavo no exista, el maestro esperará el tiempo de timeout predefinido. Por otra parte el esclavo no responderá instantáneamente y posiblemente retarde su respuesta algún ms.

También es cierto que hemos elegido la velocidad de comunicación más baja definida en el estándar y que simplemente eligiendo la velocidad inmediatamente superior de 19200 b/s el tiempo se reduce a la mitad.

Las buenas prácticas de instalación consistentes simplemente en asignar direcciones consecutivas a los esclavos permiten reducir drásticamente los tiempos de búsqueda. Por ejemplo si solo están configurados 10 esclavos, la petición al esclavo 11 fallará y terminará el ciclo con lo que el número de esclavos interrogados será de 11 y el tiempo de búsqueda será de 24*11 = 264 ms + tiempo de timeout de la última petición.

Ya que la comprobación de los esclavos se realiza en la fase de arranque del maestro y no cíclicamente, no parece interesante buscar mecanismos más sofisticados para mejorar los tiempos de obtención de esta información del bus.

Para mantener actualizada la información de la red, las comunicaciones periódicas con los esclavos configurados nos indicarán los problemas concretos de comunicación con estos dispositivos lo que permitirá determinar su desconexión o apagado.

4.2 ASIGNAMIENTO DE LAS DIRECCIONES DE LOS ESCLAVOS EN LAS SUBREDES

En sistemas modernos de control es importante realizar la asignación de direcciones de los esclavos desde el ordenador central de forma individual o automática. Su interés es mayor aún en la configuración de un equipo principal con sub-esclavos. Esta funcionalidad no está contenida en el protocolo original y por tanto se propone como una extensión del mismo.

Dado que se propone una ampliación del protocolo, es posible que en una instalación concreta parte de los esclavos instalados no conozcan los procedimientos ampliados y sólo permitan la asignación de la dirección por los procedimientos clásicos. Por ello en los procesos de asignación de dirección no se podrán utilizar mensajes que confundan a los esclavos que no conozcan estos procedimientos.

Teniendo en cuenta los procedimientos utilizados en los protocolos LON y KNX, sabemos que en KNX se debe utilizar un pulsador para identificar al dispositivo que va a recibir una dirección física durante la instalación. Es posible además a través del programa ETS identificar el dispositivo que tiene una dirección física concreta haciendo parpadear su led de programación frontal. En LON sabemos que cada dispositivo tiene un identificador único de 48 bits (NEURON ID) y que a través de este identificador es posible asignar una posición al dispositivo en la topología de la red. Del mismo modo que en KNX podemos hacer parpadear el led del pinservice frontal desde el equipo central para su identificación en campo.

Se propone en este trabajo que cada dispositivo Modbus con protocolo extendido, disponga de un identificador único de 48 bits que denominaremos ModbusID, un pulsador de programación y un led de programación frontal.

Cuando se pulsa el botón de programación de un esclavo, este enciende su led de programación y espera la recepción de la función de programación correspondiente. Si no se recibe el mensaje del maestro en un tiempo preestablecido, se restaura la actividad del esclavo a su funcionamiento normal.

Los 48 bits del ModbusID se estructuran con la siguiente información:

- 12 bits de identificación de fabricante
- 4 bits de identificación del tipo de aparato
- 6 bits de identificación del modelo
- 26 bits de identificación del número de aparato

Los 4 bits de identificación del aparato clasifican al equipo por la función realizada en el sistema de control. Los dos bits de mayor peso indican el área de utilización y los dos de menor peso el tipo de dispositivo.

B_3	B_2		B_1	B_0	
0	0	Equipo para climatización	0	0	Sensor
0	1	Equipo para Iluminación	0	1	Actuador
1	0	Equipo E/S general	1	0	Controlador
1	1	Otros	1	1	Módulo de sistema

Los módulos de sistema incluyen las pasarelas, los repetidores y otros dispositivos para la instalación como por ejemplo fuentes de alimentación con diagnóstico.

Los 26 bits de identificación del dispositivo actúan como un número de serie, lo que permite 67108864 aparatos de cada modelo.

Los procedimientos relacionados con el asignamiento de direcciones deben permitir:

- El asignamiento de la dirección del esclavo a través del bus de comunicaciones
- El borrado de la dirección individual asignada a un esclavo
- El encendido del led de programación desde el ordenador central para su identificación
- El Asignamiento automático de las direcciones a los esclavos que no tengan asignada dirección
- La resolución de conflictos de dirección debida a direcciones repetidas

Para realizar esta tarea necesitamos definir nuevas funciones Modbus específicas para esta tarea.

El mapa de funciones modbus contiene hasta 127 funciones, gran parte de ellas públicas, muchas de las cuales están sin utilizar y algunas zonas están reservadas para funciones específicas de fabricantes.

Para evitar posibles problemas futuros con los códigos de función es preferible usar códigos en las bandas a definir por el usuario, es decir 65-72 y 100-110.

Las nuevas funciones que definimos para la gestión de las direcciones son:

- Función 100: solicita a los esclavos su identificador de 48 bits
- Función 101: asigna una dirección a un esclavo en base a su ModbusID
- Función 102: identifica a un esclavo con su led de programación

Los mensajes intercambiados entre el maestro y el esclavo deben tener el formato de los mensajes Modbus convencionales.

Algunas de las funciones deben enviarse simultáneamente a todos los esclavos de la subred y pueden requerir respuesta de los esclavos. La dirección 0 es broadcast y con ella todos los esclavos recibirán los mensajes, no obstante si necesitamos una respuesta de los esclavos no debemos usar esta dirección en los mensajes ya que según el documento del estándar los esclavos no deben responder a las peticiones a esta dirección broadcast 0.

Puesto que las direcciones 1 a 247 son válidas como direcciones, nos quedan libres la direcciones 247 a 255 que según el estándar están reservadas. La dirección 255 se suele usar para comunicar con las pasarelas en redes que las utilizan. Así que haremos uso de las direcciones 248 a 254 para los procesos broadcast del protocolo ampliado que precisan respuesta de los esclavos.

4.2.1 FUNCIÓN 100: PETICIÓN DE IDENTIFICADORES

Esta función se utilizará para solicitar los identificadores ModbusID a los esclavos.

El comportamiento de la función dependerá de la dirección del esclavo indicada en la petición:

- Si la dirección es una dirección válida (1-247), se devolverá el identificador de dicho esclavo.
- Si la dirección es broadcast 254 se iniciará el proceso automático de identificación de los esclavos. Los esclavos que tengan una dirección asignada no responderán a este mensaje. Este proceso se describe en el punto 3.2.x.
- Si la dirección es broadcast 253 devolverá el identificador del esclavo que tenga su pulsador de programación pulsado. Solo deberá haber un esclavo con el pulsador activado ya que si no se producirá un conflicto de comunicaciones.

Petición de identificadores de 48 bits

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo
100	Código de función
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

La respuesta contiene los 48 bits del identificador ModbusID del esclavo que responde al mensaje.

El valor de la dirección del esclavo en la respuesta dependerá de la dirección del esclavo usada en el mensaje de petición:

- Si la dirección del esclavo en la petición era de 1 a 247, su valor se devuelve también en el mensaje de respuesta.
- Si la dirección del esclavo en la petición era 253 o 254, se devuelve 0 en el campo de dirección del esclavo con objeto de que el resto de los esclavos no lo interpreten como una nueva petición del maestro.

Respuesta a petición de identificadores 48 bits

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo
100	Código de función
ID6	48 bits de identificación del esclavo. Primero se envían los 8 bits de mayor peso
ID5	
ID4	
ID3	
ID2	

ID1	
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

4.2.2 FUNCIÓN 101: ASIGNACIÓN DE LA DIRECCIÓN DEL ESCLAVO

Esta función permite programar una dirección de 8 bits al esclavo indicado. La nueva dirección debe ser válida y comprendida en el rango 1 a 247.

Si el valor indicado de la nueva dirección tiene el valor 0, se procederá al borrado de la dirección del esclavo que quedará sin dirección.

El modo de operación es función del valor de la dirección del esclavo al que se envía la petición:

- Si la dirección de 8 bits del esclavo es un valor válido de dirección (1 a 247), se ignorará el valor del ModbusID y se producirá una reprogramación de la dirección del esclavo al nuevo valor indicado en el mensaje.
- Si la dirección del esclavo es una dirección broadcast 254, y el valor del ModbusID de 48 bits es distinto de cero, se programará la nueva dirección al esclavo cuyo ModbusID coincida con el enviado.
- Si la dirección del esclavo es una dirección broadcast 254 y el campo de los 48 bits del ModbusID es todo ceros, se realizará la programación de la dirección al esclavo que tenga pulsado el botón de programación.

Petición asignación de dirección del esclavo

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo
101	Código de función
ID6	48 bits de identificación del esclavo. Primero se envían los 8 bits de mayor peso
ID5	
ID4	
ID3	
ID2	
ID1	
DIR	Nueva dirección del esclavo (1 a 247). Si el valor es 0 se borra la dirección del esclavo y este queda desprogramado.
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

La respuesta será similar al mensaje de petición. El campo ModbusID tendrá siempre el valor del esclavo que responde a la petición. El campo de nueva dirección del esclavo tendrá el valor de la nueva dirección que ha tomado el esclavo.

El valor del campo de dirección del esclavo devuelto en la respuesta dependerá del valor del mismo campo en la petición realizada por el maestro:

- Si la dirección de 8 bits del esclavo era un valor válido de dirección (1 a 247), se devolverá el mismo valor en la respuesta.
- Si la dirección del esclavo era una dirección broadcast 254, el valor en la respuesta debe ser 0 para que no se interprete por el resto de los esclavos como una nueva petición del maestro.

Respuesta asignación de dirección del esclavo

BYTES	DESCRIPCIÓN
254	Dirección del esclavo
101	Código de función
ID6	48 bits de identificación del esclavo. Primero se envían los 8 bits de mayor peso
ID5	
ID4	
ID3	
ID2	
ID1	
DIR	Nueva dirección del esclavo
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

4.2.3 FUNCIÓN 102: ACTIVAR LED DE PROGRAMACIÓN

Esta función se utilizará para identificar a un esclavo a través del parpadeo de su led de programación. Es muy útil cuando se tienen instalados varios dispositivos iguales en un cuadro eléctrico y no se sabe con certeza quién es cada dispositivo. El tiempo de intermitencia del led es función exclusiva de la programación del esclavo.

El comportamiento de la función dependerá del valor del esclavo direccionado:

- Si la dirección del esclavo es una dirección válida (1 a 247), se usará este valor para identificar al esclavo y se ignorará el valor del campo MosbusID.
- Si la dirección del esclavo es broadcast con valor 254 se utilizarán los 48 bits del ModbusID para la identificación del esclavo.

De esta forma es posible identificar visualmente a los esclavos que tienen su dirección asignada y a los que no estén configurados.

Petición

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo
102	Código de función
ID6	48 bits de identificación del esclavo. Primero se envían los 8 bits de mayor peso
ID5	
ID4	
ID3	
ID2	
ID1	
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

El mensaje de respuesta es similar al de petición con las siguientes salvedades:

- El ModbusID será siempre el del esclavo que responda al mensaje.
- Si la dirección del esclavo era 254 en la petición, el campo de dirección del esclavo será 0 en la respuesta para que no se interprete por otros esclavos como una nueva petición del maestro.

Respuesta

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo
102	Código de función
ID6	48 bits de identificación del esclavo. Primero se envían los 8 bits de mayor peso
ID5	
ID4	
ID3	
ID2	
ID1	
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

4.2.4 PROCEDIMIENTO DE OBTENCIÓN DE LOS IDENTIFICADORES DE LOS ESCLAVOS DE UNA SUBRED Y ASIGNAMIENTO AUTOMÁTICO DE SUS DIRECCIONES

La solución más eficiente es la más simple, que consiste en disponer de un código de barras o algún identificador similar en cada dispositivo que pueda ser capturado por el sistema de control. En los equipos LON es frecuente que cada dispositivo se suministre con dos pegatinas de código de barras, una fija y otra despegable para llevarla al ordenador de control que realizará después la configuración. El técnico de campo adhiere esa pegatina sobre un cuaderno donde figura el tipo de dispositivo y su ubicación para correlacionarlo después en el sistema de supervisión con los planos de la instalación.

Desde el ordenador de control y con la ayuda de un operario, es posible encontrar los identificadores de los dispositivos usando la función 100 y pulsando previamente el pulsador de programación del dispositivo. Esto implica que hay que ir cuadro por cuadro y dispositivo por dispositivo para realizar esta tarea, pero tiene la ventaja de que correlacionamos directamente el dispositivo con su ubicación en el plano y con su función de control.

Si los dispositivos tienen asignada ya dirección se sobreentiende de que se dispone ya del identificador del dispositivo, en caso contrario llamando a la función 100 con la dirección del esclavo obtendremos su identificador.

Si tenemos una red entera sin conocer los identificadores de los dispositivos y sin tener asignadas sus direcciones en el bus, es mejor solución configurar automáticamente todo el sistema y posteriormente identificar los dispositivos de campo a través de su función de control (por ejemplo encendiendo una luz desde el ordenador de control). Una vez identificados los dispositivos, se etiquetarán conveniente en los planos de control y se tendrá la información correcta de los puntos de control para usarlos en horarios u otras funciones.

El proceso de obtención de los identificadores y asignamiento de direcciones se realiza con el siguiente procedimiento:

1. El maestro inicia un ciclo de reconocimiento de los esclavos con dirección asignada, para averiguar las direcciones ocupadas y las libres en la subred, usando el procedimiento descrito en el apartado 3.1. Si fuera necesario obtener los identificadores de estos dispositivos se utilizará la función 100 con su número de esclavo.
2. El maestro enviará un mensaje al bus para que respondan los esclavos que no tienen dirección asignada usando la dirección broadcast 254. Cada esclavo usará un procedimiento CSMA/CD generando internamente una temporización T para obtener un slot temporal en el que determine si el bus está libre y pueda entonces enviar su respuesta. Este tiempo T se obtiene como el producto de un valor aleatorio entre 1 y 50 y el valor de la duración de 1,1 caracteres que es función de la velocidad de transmisión. Si un dispositivo escucha un mensaje de respuesta de otro esclavo, anula su respuesta y espera a una nueva petición del esclavo. A la finalización de este paso y si había esclavos sin configurar y no se han producido colisiones, el maestro habrá recibido la identificación de un esclavo con dirección no asignada. Si el maestro detecta una colisión en la comunicación repite el envío del mensaje.
3. Si se recibe una respuesta correcta, el maestro procede a asignar una dirección libre al esclavo que ha enviado su identificación usando la función 101. Una vez asignada la dirección el maestro repite el procedimiento volviendo al paso 2.

4. Si no hay respuesta de los esclavos y no se han detectado errores de colisiones, el proceso termina y todos los esclavos han quedado configurados

Con objeto de determinar con precisión el instante de comienzo de transmisión de un dispositivo en la red, es necesario usar una señal de interrupción asociada al pin de recepción. De esta forma cualquier dispositivo conectado a la red recibirá el bit de inicio del dispositivo que comienza la transmisión y determinará con precisión el instante de ocupación del bus.

5 CLASES Y PROPIEDADES

5 CLASES Y PROPIEDADES

Se definen a continuación las clases implementadas en el protocolo ampliado. Se ha considerado un conjunto mínimo funcional de clases, que lógicamente es susceptible de ampliación en futuras revisiones.

Un aspecto diferenciador de estas clases para el protocolo Modbus es su relación con los conjuntos de valores asociados a las funciones clásicas de Modbus 1 a 6. Independientemente de que podamos acceder a los valores de los objetos a través de nuevas funciones, debemos mantener el acceso a través de las funciones clásicas. Por ello cada objeto deberá tener las propiedades necesarias para identificar claramente el mapa de registros asociado a la información de su valor y el número de registros usados para contenerla.

Como sabemos el tipo de datos asociado a los registros no está especificado en el protocolo Modbus y tendremos que comunicarlo al maestro a través de una nueva función.

Todas las propiedades de los objetos permiten los servicios BuscarPropiedades, LeerPropiedad y LeerValor. El servicio LeerPropiedades es general al esclavo y tiene el mismo comportamiento para todas las propiedades. Será la instancia de la clase “dispositivo” la que establezca si el servicio está disponible para el esclavo. La propiedad EscribirValor estará permitida o no en función de cada propiedad particular por lo que es necesario especificarlo en la descripción de cada propiedad.

Para indicar los mapas de registros, usaremos a partir de ahora la nomenclatura siguiente:

- COIL para coils
- DI para discrete inputs
- HR para holding registers
- IR para input registers

5.1 DISPOSITIVO

Esta clase identifica a los dispositivos esclavos y suministra información acerca de sus características generales. Cada esclavo debe tener una instancia de esta clase.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	ID 48 bits	UInt16B[3]	x
3	Nombre del dispositivo	String	x
4	Número de instancia de la clase	Uint16B	x
5	Descripción del dispositivo	String	
6	Localización del dispositivo	String	
7	Clases implementadas	UInt64B	x
8	Número de objetos en el esclavo	UInt16B	x
9	Propiedades Múltiples habilitadas	Bool	x
10	Fabricante	String	x
11	Código de producto	String	x
12	Revisión	String	x
13	URL del vendedor	String	
14	Nombre del producto	String	
15	Nombre del modelo	String	
16	Nombre de la aplicación	String	
17	Versión de firmware	String	x
18	Versión del protocolo	String	x
19	Reset	Bool	x
20	Offline	Bool	x
21	Online	Bool	x

Identificador del objeto

Es un campo que identifica de manera única a este objeto respecto del resto de los objetos dentro del esclavo. Este campo siempre toma el valor 1 ya que es el primer objeto del esclavo. El campo es obligatorio y tiene asociado un tipo de datos UInt16B.

El servicio EscribirValor no está permitido.

ID 48 bits

Contiene el código universal de 48 bits del dispositivo. Los 48 bits se guardan en 3 registros correlativos de 16 bits.

El servicio EscribirValor no está permitido.

Nombre del dispositivo

Es una cadena de caracteres corta que identifica al dispositivo con un texto significativo. El tamaño máximo de la cadena es de 16 caracteres y es NULL terminado. El campo es obligatorio y se usará en las aplicaciones de visualización al objeto.

El servicio EscribirValor está permitido.

Número de instancia de la clase

Es un identificador que distingue a un objeto de una clase de otro. Su valor se utiliza internamente para indexar a la zona de memoria de almacenamiento de los datos del objeto. El campo es obligatorio y el tipo de dato asociado es UInt16B lo que permite 65535 instancias de una clase. Solo debe existir una instancia de esta clase.

El servicio EscribirValor no está permitido.

Descripción del dispositivo

Es una cadena de caracteres que contiene información adicional del dispositivo. El tamaño máximo es de 40 caracteres y debe Null terminada. Su uso es opcional.

El servicio EscribirValor está permitido.

Localización del dispositivo

Es una cadena de caracteres que contiene información sobre la ubicación del dispositivo en la instalación. Puede describir la planta, armario eléctrico etc. donde esté ubicado el dispositivo. El tamaño máximo es de 40 caracteres y debe Null terminada. Su uso es opcional.

El servicio EscribirValor está permitido.

Clases implementadas

Este campo indica cuales son los tipos de clases implementadas en el dispositivo. El tipo de dato asociado es UInt64B de 64 bits. Cada bit está asociado a un tipo de clase de manera que si el bit correspondiente tiene el valor 1 la clase está implementada y si vale 0 no lo está. Esto nos permite manejar hasta 64 clases distintas en esta versión del protocolo. En versiones posteriores se podría aumentar el tamaño del campo si fuera necesario. El uso de este campo es obligatorio.

El servicio EscribirValor no está permitido.

Número de objetos en el esclavo

Este campo contiene el número total de objetos gestionados por el esclavo para todas las clases. El tipo de dato asociado es UInt16B. El uso de este campo es obligatorio.

El servicio EscribirValor no está permitido.

Propiedades Múltiples habilitadas

Este campo especifica si el dispositivo permite la petición de varias propiedades en un mensaje. El tipo de datos asociado es Bool y su uso es obligatorio.

El servicio EscribirValor no está permitido.

Fabricante

Contiene el nombre del fabricante en una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Basic indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es obligatorio

El servicio EscribirValor no está permitido.

Código de producto

Contiene el código de fabricante del producto expresado como una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Basic indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es obligatorio

El servicio EscribirValor no está permitido.

Revisión

Contiene el valor de la revisión del producto expresado como una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Basic indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es obligatorio

El servicio EscribirValor no está permitido.

URL del vendedor

Contiene la URL del vendedor expresada como una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Regular indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es opcional

El servicio EscribirValor no está permitido.

Nombre del producto

Contiene el nombre del producto expresado como una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Regular indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es opcional

El servicio EscribirValor no está permitido.

Nombre del modelo

Contiene el nombre del modelo del dispositivo expresado como una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Regular indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es opcional

El servicio EscribirValor no está permitido.

Nombre de la aplicación

Contiene el nombre de la aplicación expresado como una cadena de caracteres Null terminada. Este campo se corresponde con uno de los campos de la categoría Regular indicados en la función Modbus 43.

El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es opcional

El servicio EscribirValor no está permitido.

Versión de firmware

Contiene la versión de firmware de la aplicación expresada como una cadena de caracteres Null terminada. El tamaño máximo de la cadena es de 20 caracteres. El uso del campo es obligatorio.

El servicio EscribirValor no está permitido.

Versión del protocolo

Contiene la información de la versión del protocolo Modbus implementada, expresada en una cadena de caracteres de longitud 6 Null terminada. El valor por defecto para esta versión de Modbus ampliado es “2.0”. El uso del campo es obligatorio.

El servicio EscribirValor no está permitido.

Reset

Esta propiedad se usa como una señal de control para ejecutar el reset del esclavo. El tipo de dato es Bool y la escritura de un valor 1 provoca el Reset del dispositivo. El uso del campo es obligatorio.

El servicio EscribirValor está permitido.

Offline

Esta propiedad se usa una señal de control para ejecutar la orden Offline del esclavo. El tipo de dato es Bool y la escritura de un valor 1 provoca la entrada del dispositivo en el estado Offline, en el cual el dispositivo funcionará localmente pero no atenderá a las comunicaciones. La restitución del dispositivo al estado normal se puede hacer a través de las propiedades Reset y Online. El uso del campo es obligatorio.

El servicio EscribirValor está permitido.

Online

Esta propiedad se usa una señal de control para ejecutar la orden Online del esclavo. El tipo de dato es Bool y la escritura de un valor 1 provoca la entrada del dispositivo en el estado Online, en el cual el dispositivo funcionará normalmente y atenderá a las comunicaciones. El uso del campo es obligatorio.

El servicio EscribirValor está permitido.

5.2 ENTRADA BINARIA

Una entrada binaria es un objeto asociado a una entrada digital o de estado del esclavo y a cuya información se accede a través de la función Modbus 2.

Una entrada binaria tiene dos estados diferenciados asociados a los valores binarios 0 y 1. Su interpretación varía en función del uso asignado a la entrada binaria en el esclavo. Valores como ON/OFF, VERDADERO/FALSO, HABILITADO/NO HABILITADO describen el significado asociado a los valores 0 y1 de la entrada digital.

Un texto asociado a los valores 0 y 1 del objeto nos permitirá obtener una información clara del significado del estado del objeto y su polaridad nos permitirá invertir el comportamiento normal de la entrada. Por ejemplo si para nosotros el valor 1 corresponde a OFF y el 0 a ON lo podremos expresar en esos textos asociados a los valores de los estados modificando además la polaridad para invertir el sentido establecido por defecto a los valores binarios del objeto.

Se permite en Modbus que el valor de una entrada digital esté en el mapa de registros de las entradas discretas ocupando un registro, pero además puede estar en un Holding register o Input register ocupando un bit o todo el registro.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Valor actual	Bool	x
6	Calidad del dato	Uint8	x
7	Polaridad	Bool	x
8	Texto para valor Activo	String	
9	Texto para valor Inactivo	String	
10	Mapa de registros	Uint8	x
11	Número de registro	Uint16B	x
12	Número de bit del registro	Uint8	
13	Valor registro para Activo	Uint16B	
14	Valor registro para Inactivo	Uint16B	
15	Tiempo mínimo estable para cambio estado	Uint16B	
16	Contador de cambios de estado	Uint32B	
17	Fecha inicio contador cambios de estado	Fecha y Hora	
18	Borrado contador cambios estado	Bool	
19	Fecha del último cambio estado	Fecha y Hora	

Identificador del objeto

Es un campo que identifica de manera única a un objeto dentro del esclavo. Este identificador se crea en el esclavo en fase de diseño de su aplicación. El campo es obligatorio y tiene asociado un tipo de datos UInt16B.

El servicio EscribirValor no está permitido.

Nombre del objeto

Es una cadena de caracteres corta que identifica al objeto con un texto significativo. El tamaño máximo de la cadena es de 16 caracteres y es NULL terminado. El campo es obligatorio y se usará en las aplicaciones de visualización del objeto.

El servicio EscribirValor está permitido.

Número de instancia de la clase

Es un identificador que distingue a un objeto de la clase de otro. Su valor se utiliza internamente para indexar a la zona de memoria de almacenamiento de los datos del objeto. El campo es obligatorio y el tipo de dato asociado es UInt16B lo que permite 65535 instancias de una clase.

El servicio EscribirValor no está permitido.

Descripción del objeto

Es una cadena de caracteres que contiene información adicional del objeto. El tamaño máximo es de 40 caracteres y debe Null terminada. Su uso es opcional.

El servicio EscribirValor está permitido.

Valor actual

Contiene el valor binario actual del objeto. El tipo de datos usado es Bool que nos permite distinguir los valores 0 y 1 pero no el significado físico asociado a los valores. Su uso es obligatorio.

El servicio EscribirValor no está permitido.

Calidad del dato

Este campo indica si la información asociada al valor actual es fiable o no. El tipo de datos asociado es UInt8 y su uso es obligatorio. Los valores posibles de calidad del dato son:

VALOR	DESCRIPCIÓN
0	Dispositivo Inactivo
1	Fallo de hardware
2	Dato válido

El servicio EscribirValor no está permitido.

Polaridad

Este campo, de tipo bool, indica cómo se interpretan los valores 0 y 1 del campo valor actual del objeto. Los valores permitidos de la polaridad son NORMAL=0 e INVERTIDA=1. Si la polaridad es NORMAL entonces el valor 0 de la entrada binaria se corresponde con el estado Inactivo de la entrada binaria y el valor 1 con el estado Activo. Si la polaridad es INVERTIDA entonces el valor 0 de la entrada binaria se corresponde con el estado Activo y si el valor 1 con el estafo Inactivo. El campo es obligatorio y su valor por defecto es NORMAL.

El servicio EscribirValor está permitido.

Texto para valor Inactivo

Este campo es una cadena de longitud máxima 16 caracteres NULL terminada que informa sobre el significado del valor Inactivo del valor actual del objeto. El texto debe ser descriptivo del tipo de información binaria contenida en el objeto. Por ejemplo valores típicos son OFF, APAGADO, ABIERTO etc. El campo es opcional y su valor por defecto es OFF.

El servicio EscribirValor está permitido.

Texto para valor Activo

Este campo es una cadena de longitud máxima 16 caracteres NULL terminada que informa sobre el significado del valor Activo del valor actual del objeto. El texto debe ser descriptivo del tipo de información binaria contenida en el objeto. Por ejemplo valores típicos son ON, ENCENDIDO, CERRADO, ACTIVO etc. El campo es opcional y su valor por defecto es ON.

El servicio EscribirValor está permitido.

Mapa de registros

Especifica el mapa de registros Modbus donde está contenida la información de la entrada binaria. Los valores posibles se corresponden con la función Modbus que permite el acceso a esos registros, así COILS=1, HR=2, HR=3, IR=4. El tipo de datos del campo es UInt8 y es obligatorio.

El servicio EscribirValor no está permitido.

Número de registro

Especifica el número de registro del mapa de registros indicado en el campo anterior donde está contenido el dato. El tipo de dato es UInt16B y es obligatorio.

El servicio EscribirValor no está permitido.

Número de bit de registro

El valor binario del objeto puede estar contenido en registros binarios del tipo coils o discrete inputs, pero también puede estar en un registro de 16 bits tipo holding register o input register ocupando 1 bit o los 16 bits del registro.

Este campo no es obligatorio y solo existe cuando el dato está almacenado de la forma descrita. El tipo de datos es UInt8 y su valor indica el bit ocupado (0-15).

El servicio EscribirValor no está permitido.

Valor registro para Inactivo

Si el valor binario está almacenado en un registro de 16 bits y los valores 0 y 1 están almacenados como valores de 16 bits, este campo contiene el valor de 16 bits asociado al valor binario Inactivo.

El campo es opcional ya que solo existirá en el caso indicado. El tipo de dato asociado es UInt16B.

El servicio EscribirValor no está permitido.

Valor registro para Activo

Si el valor binario está almacenado en un registro de 16 bits y los valores 0 y 1 están almacenados como valores de 16 bits este campo contiene el valor asociado al valor binario Activo.

El campo es opcional ya que solo existirá en el caso indicado. El tipo de dato asociado es UInt16B.

El servicio EscribirValor no está permitido.

Tiempo mínimo estable para cambio estado

Los valores de la entrada binaria pueden sufrir cambios de muy corta duración debido fundamentalmente a los rebotes de los contactos de los elementos de control. Este tiempo establece el tiempo mínimo que la señal de entrada debe estar estable para considerar que se ha producido un cambio de estado en la entrada. El uso de este parámetro permite eliminar falsos cambios de estado actuando como un filtro.

El tipo de dato asociado es UInt16B y su valor representa el número de milisegundos que la entrada debe permanecer estable para considerar que se ha producido un cambio de estado. El uso de esta propiedad es opcional.

El servicio EscribirValor está permitido.

Contador de cambios de estado

Este parámetro actúa como un contador de cambios de la entrada. Cualquier cambio 0->1 y 1->0 provoca el incremento del valor del contador. El tipo de datos asociado es UInt32B y su uso es opcional. Es posible escribir el valor de la propiedad para inicializar su valor.

El servicio EscribirValor está permitido.

Fecha inicio contador cambios de estado

Esta propiedad contiene la información de la fecha y hora desde la que se ha comenzado a contabilizar los cambios de estado. El tipo de datos asociado es FechaHora y el uso de la propiedad es opcional.

El servicio EscribirValor no está permitido.

Borrado contador cambios estado

Este parámetro permite inicializar el contador de cambios de estado. Para inicializarlo escribimos un 1 en su campo valor. Las acciones que se desencadenan son la puesta a cero del contador y la escritura del valor de la fecha y hora de inicio a la fecha y hora actual del RTC. El tipo de datos es Bool y el campo es opcional. Si el campo contador de cambios de estadoes usado, esta parámetro debe también usarse.

El servicio EscribirValor está permitido.

Fecha del último cambio estado

Este campo contiene el valor de la fecha y hora del último cambio producido en la entrada. El tipo de datos asociado es FechaHora y su uso es opcional.

El servicio EscribirValor no está permitido.

5.3 SALIDA BINARIA

Un objeto de la clase salida binaria está conectado a una salida física del esclavo como por ejemplo un relé, contactor, telerruptor o una salida a colector abierto o bien puede tratarse de una señal

virtual utilizada como una orden de control. El acceso a la lectura con las funciones Modbus es a través de la función 1 o 3 para lectura y la función 5 o 6 para escritura.

Se puede también modificar el valor de una salida binaria a través de un registro de 16 bits (Holding register o Input register). En este caso la salida puede estar representada por 1 bit en el registro u ocupar todo el registro con dos valores predeterminados para los valores Activo y Inactivo.

En una instalación de calidad se suele usar un contacto auxiliar de la salida para poder leer su estado real. Esto permite detectar fallos en el propio relé o el circuito de conmutación del mismo. Esta señal es una entrada digital de estado que se gestionará a través de un objeto de la clase ENTRADA BINARIA.

El protocolo Modbus permite a través de la función 1 leer el estado de la salida binaria tal como la ha programado el esclavo. Esto es útil por ejemplo en una inicialización del maestro para sincronizar los valores de las salidas digitales con los valores del maestro. Sin embargo hay que notar que el valor real de la salida puede ser distinto si se ha producido un error en la electrónica del esclavo.

El elemento de conmutación de salida puede ser un telerruptor, en cuyo caso la actuación de la electrónica de la salida binaria es a través de pulsos. La electrónica del esclavo tendrá que procesar el valor del estado actual y el nuevo valor solicitado para generar en caso necesario el pulso adecuado.

Si existe señal de estado, puede ser interesante también que el esclavo compare el valor programado de la salida binaria con el estado real para detectar anomalías y generar una señal de aviso.

Otro aspecto importante y ampliamente usado en las salidas binarias es la temporización. Esta permite realizar la conmutación ON y OFF con retrasos independientes o bien establecer una acción ON con una duración predeterminada tras la cual se conmuta al estado OFF.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Valor actual	Bool	x
6	Calidad del dato	Uint8	x
7	Polaridad	Bool	x
8	Texto para valor Inactivo	String	
9	Texto para valor Activo	String	
10	Mapa de registros	Uint8	x
11	Número de registro	Uint16B	x
12	Número de bit del registro	Uint8	
13	Valor registro para Inactivo	Uint16B	
14	Valor registro para Activo	Uint16B	

15	Tiempo mínimo estable antes de cambio	Uint16B	
16	Contador cambios de estado	Uint32B	
17	Fecha inicio contador cambios de estado	Fecha y Hora	
18	Borrado contador cambios de estado	Bool	
19	Fecha último cambio de estado	Fecha y Hora	
20	ID de la entrada Binaria de estado asociada	Uint16B	
21	Alarma Estado	Bool	
22	Retraso ON	Uint16B	
23	Retraso OFF	Uint16B	
24	Duración pulso ON-OFF	Uint16B	
25	Persistencia	Bool	
26	ID de señal de habilitación	Uint16B	
27	Valor estado para habilitación	Bool	
28	ID de señal de bloqueo	Uint16B	
29	Valor estado bloqueo	Bool	

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Valor actual

Ver la descripción en la clase Entradas Binarias

Calidad del dato

Ver la descripción en la clase Entradas Binarias

Polaridad

Ver la descripción en la clase Entradas Binarias

Texto para valor Inactivo

Ver la descripción en la clase Entradas Binarias

Texto para valor Activo

Ver la descripción en la clase Entradas Binarias

Mapa de registros

Ver la descripción en la clase Entradas Binarias

Número de registro

Ver la descripción en la clase Entradas Binarias

Número de bit de registro

Ver la descripción en la clase Entradas Binarias

Valor registro para Inactivo

Ver la descripción en la clase Entradas Binarias

Valor registro para Activo

Ver la descripción en la clase Entradas Binarias

Tiempo mínimo estable antes del cambio

Normalmente las salidas binarias actúan sobre un elemento electromecánico como un relé o contactor. Estos elementos no permiten una alta velocidad de conmutación y además su vida útil se mide por el número de conmutaciones realizadas. Para proteger al elemento electromecánico, se impedirán conmutaciones continuas de estos elementos de forma que este parámetro indicará el tiempo mínimo que la salida debe permanecer sin cambios antes de aplicar el siguiente cambio en la salida. Si se produce una orden de conmutación antes de que haya transcurrido este tiempo, la salida no cambiará inmediatamente pero si lo hará una vez transcurrido el tiempo mínimo indicado al valor de la última orden recibida.

El valor de este parámetro está expresado en milisegundos y el tipo de datos asociado es UInt16B.

El servicio EscribirValor está permitido.

Contador cambios de estado

Ver la descripción en la clase Entradas Binarias

Fecha inicio contador cambios de estado

Ver la descripción en la clase Entradas Binarias

Borrado contador cambios de estado

Ver la descripción en la clase Entradas Binarias

Fecha del último cambio de estado

Ver la descripción en la clase Entradas Binarias

ID de la entrada Binaria de estado asociada

Si la salida binaria tiene asociada una entrada binaria de estado para supervisar su funcionamiento, el número de la instancia del objeto Entrada Binaria se indicará en este parámetro. Esta información la usará el esclavo para supervisar el correcto funcionamiento del esclavo. El tipo de dato asociado es UInt16B. El uso de la propiedad es opcional.

El servicio EscribirValor está permitido.

Alarma Estado

Si el esclavo tiene entrada de estado asociada, este supervisará la congruencia entre la orden enviada a la salida binaria y el estado leído desde la entrada binaria de estado asociada. Si se produce una disconformidad entre ambos valores el esclavo indicará esta circunstancia a través de este parámetro señalando una alarma de funcionamiento. El tipo de dato asociado a este parámetro es Bool y su uso es opcional.

El servicio EscribirValor no está permitido.

Retraso ON

En algunas aplicaciones se precisa que la conmutación ON u OFF de una salida binaria no sea instantánea sino temporizada. Este parámetro indica el retraso a aplicar a la conmutación OFF->ON de la salida binaria. El tipo de dato asociado es UInt16B y su valor representa segundos de espera. El parámetro es opcional.

El servicio EscribirValor está permitido.

Retraso OFF

De manera similar al retraso en ON, este parámetro indica el retraso a aplicar a la conmutación ON->OFF de la salida binaria. El tipo de dato asociado es UInt16B y su valor representa segundos de espera. El parámetro es opcional.

El servicio EscribirValor está permitido.

Duración pulso ON-OFF

En algunas ocasiones el uso deseado de la salida binaria es para realizar una operación de conmutación temporizada en la que se realiza una conmutación OFF->ON seguida de un tiempo de espera y después una conmutación ON->OFF. Un ejemplo clásico es el de una luz de escalera. Este parámetro indica el tiempo que permanecerá la salida ON, transcurrido el cual la salida conmutará a OFF. El tipo de dato asociado es UInt16B y representa los segundos que la salida permanecerá en ON. Su uso es opcional.

El servicio EscribirValor está permitido.

Persistencia

Si la salida binaria actúa sobre un relé y su estado actual es ON, al producirse un reinicio del esclavo o una pérdida de alimentación el estado de la salida pasará a OFF hasta que el maestro modifique nuevamente el valor de la salida binaria. Si esta situación no es admisible, indicaremos con este parámetro a la salida que su valor es persistente, lo que significa que cada vez que se produzca un cambio en la salida, su valor se almacenará en una memoria no volátil y cuando se produzca un reinicio del esclavo, este rescatará el último valor conocido de la salida de la memoria no volátil y procederá automáticamente a su actualización.

El tipo de dato asociado es Bool y su uso es opcional. El valor 1 indica usar persistencia.

El servicio EscribirValor está permitido.

ID de señal de habilitación

En algunas aplicaciones de control, se precisa que el valor ON de la salida solo pueda alcanzarse si una señal de habilitación está activada. Se considera que el valor asociado a esta señal de habilitación es Bool.

Este parámetro indica el número de instancia de una señal binaria que realizará el control de habilitación. Ya que la señal es Bool y debe ser posible modificar su valor, se tratará de una salida binaria. Hay que recordar que una salida binaria tiene asociado un registro en los mapas de COILS o de HR pero la señal puede ser virtual y no tener asociada ninguna salida física en el dispositivo.

El tipo de datos asociado a este parámetro es UInt16B y su uso es opcional.

El servicio EscribirValor está permitido.

Valor estado para habilitación

Este parámetro especifica el valor de la señal de habilitación para el cual la salida binaria está habilitada. Su valor es BOOL así que un 1 en este campo indica que la habilitación se produce por el valor Activo de la entrada de habilitación. El uso de este parámetro es opcional.

El servicio EscribirValor está permitido.

ID de señal de bloqueo

En algunas aplicaciones de control, se precisa que el cambio de valor de la salida no esté permitido si una señal de bloqueo está activada. Se considera que el valor asociado a esta señal de habilitación es Bool.

Este parámetro indica el número de instancia de una señal binaria que realizará el control de bloqueo. Ya que la señal es Bool y debe ser posible modificar su valor, debe tratarse de una salida binaria. El tipo de datos asociado a este parámetro es UInt16B y su uso es opcional.

El servicio EscribirValor está permitido.

Valor estado para bloqueo

Este parámetro especifica el valor de la señal de bloqueo para el cual la salida binaria no puede modificarse. Su valor es BOOL así que un 1 en este campo indica que el bloqueo se produce por el valor Activo de la entrada binaria. El uso de este parámetro es opcional.

El servicio EscribirValor está permitido.

5.4 CONTADOR

Un contador se encarga de contar los pulsos de una entrada digital y convertirlos a una magnitud física para la medida de energía eléctrica, gas natural, energía calorífica, agua etc.

Las unidades de medida y el factor de conversión a aplicar dependen del dispositivo físico que genera los pulsos y por tanto es necesario configurar el objeto para obtener los valores correctos de medida.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	X
2	Nombre del objeto	String	X
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Valor actual	Uint32B	x

6	Calidad del dato	Uint8	x
7	Polaridad pulso	Bool	x
8	Duración mínima del pulso	Uint32B	
9	Valor actual en unidades de ingeniería	UInt32B	
10	Unidades de medida	String	
11	Número de registro para lectura valor actual	Uint16B	x
12	Número de registro para lectura valor ingeniería	Uint16B	
13	Multiplicador	Uint16B	
14	Divisor	Uint16B	

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Valor actual

Contiene el valor del número de pulsos contabilizados. Este valor se puede inicializar a uno determinado para su sincronización con un contador con display o también se puede inicializar a cero. El almacenamiento de la información se realiza en el mapa de HR ya que es posible escribir en el registro. El tipo de datos asociado es UInt32B. Este parámetro es de uso obligatorio.

El servicio EscribirValor está permitido.

Calidad del dato

Ver la descripción en la clase Entradas Binarias

Polaridad pulso

Especifica si el pulso se considera con un flanco 0->1 para el inicio y 1->0 para el final (pulsa en alto) o bien el inicio es el flanco 1->0 y el fin del pulso es el flanco 0->1 (pulso en bajo). Para el pulso en alto este parámetro tiene valor 0 y para el pulso en bajo valor 1. El tipo del dato asociado es Bool y el parámetro es obligatorio.

El servicio EscribirValor está permitido.

Duración mínima del pulso

Este parámetro especifica el tiempo mínimo de duración del pulso. El objeto de este parámetro es filtrar pulsos de ruido de pequeña duración que no provienen del medidor. El tipo de datos asociado es UInt32B y se expresa en milisegundos. El uso de este parámetro es opcional.

El servicio EscribirValor está permitido.

Valor actual en unidades de ingeniería

Este parámetro mide la información de la cuenta de pulsos en unidades de ingeniería y traducida por tanto a información final como por ejemplo m^3, KW etc. El valor se obtiene a partir del valor del contador afectándolo por una operación de multiplicación y división de factores constantes. El tipo de datos asociado es UInt32B. El uso del parámetro es opcional.

El servicio EscribirValor no está permitido.

Unidades de medida

Contiene una cadena de caracteres, descriptiva de las unidades de ingeniería con las que se expresa el parámetro anterior. El tipo de datos es String NULL terminado de longitud 10 y su uso es opcional.

El servicio EscribirValor está permitido.

Número de registro para lectura del valor actual

Este parámetro indica la dirección del primer registro de 16 bits que contiene la información del valor actual del contador. Ya que el valor de cuenta es de 32 bits, hacen falta dos registros consecutivos para almacenar la información. El tipo de datos asociado es UInt16B y su uso es obligatorio.

El servicio EscribirValor no está permitido.

Número de registro para lectura valor ingeniería

Este parámetro indica la dirección del primer registro de 16 bits que contiene la información del valor de ingeniería del contador. Ya que el valor de cuenta es de 32 bits, hacen falta dos registros consecutivos para almacenar la información. El tipo de datos asociado es UInt16B y su uso es opcional.

El servicio EscribirValor no está permitido.

Multiplicador

La conversión de unidades de cuenta a unidades de ingeniería se hace a través de la operación Unidades de ingeniería = valor actual *multiplicador/divisor. Este parámetro contiene el dato del numerador de la fracción de conversión. El tipo de datos es UInt16B y su uso es opcional.

El servicio EscribirValor está permitido.

Divisor

Este parámetro contiene el dato del denominador de la fracción de conversión. El tipo de datos es UInt16B y su uso es opcional.

El servicio EscribirValor está permitido.

5.5 ENTRADA ANALÓGICA

Una entrada analógica representa el valor de una señal analógica de entrada del esclavo conectada a un dispositivo de campo como un sensor. El valor de medida se puede representar como un dato float, como un entero, o bien como un entero sobre el que se ha efectuado una operación de multiplicación o división.

El acceso a esta información se realiza a través de registros de 16 bits en los mapas de IR o holging register usando las funciones 3 o 4 Modbus. El formato de representación del dato se especifica a través del campo tipo de dato.

El valor del objeto en el campo Valor actual se representa siempre como float.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Valor actual	Float	x
6	Calidad del dato	Uint8	x
7	Unidades de medida	String	
8	Tipo de dato	Uint16B	x
9	Mapa de registros	Uint8	x
10	Número de registro	Uint16B	x
11	Máscara	Uint64B	
12	Multiplicador para UI	Uint16B	
13	Offset para UI	Uint16B	
14	Periodo de conversión	Uint16B	
15	Multiplicado por	Uint16B	
16	Dividido por	Uint16B	

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Valor actual

Contiene el valor de la entrada analógica en unidades de ingeniería. El tipo de datos asociado es Float. Su uso es obligatorio.

El servicio EscribirValor no está permitido.

Calidad del dato

Ver la descripción en la clase Entradas Binarias

Unidades de medida

Contiene el texto de las unidades de ingeniería en una cadena de caracteres NULL terminada. El tamaño máximo es 6 caracteres incluido el Null. Su uso es opcional.

El servicio EscribirValor está permitido.

Tipo de dato

El dato de la entrada analógica está contenido en registros de 16 bits usando distintos formatos de datos. Este campo especifica el tipo de dato con el que está representada la información en los mapas de registros HR o IR.

El tipo de datos asociado a este campo es UInt16B. Su uso es obligatorio.

El servicio EscribirValor no está permitido.

Mapa de registros

Este campo especifica cual es el mapa de registros Modbus que contiene la información. El valor 3 indica HR y el valor 4 IR.

El tipo de dato asociado a este campo es UInt8. Su uso es obligatorio.

El servicio EscribirValor no está permitido.

Número de registro

Indica la dirección del primer registro de 16 bits del mapa de registros anterior que contiene la información del valor de la entrada analógica.

El tipo de datos asociado a este campo es UInt16B. Su uso es obligatorio.

El servicio EscribirValor no está permitido.

Máscara

Es una máscara de 64 bits que especifica los bits significativos para este dato. Los bits con valor 1 indican bit usado. Esta máscara permite que en un registro se guarde más de una información. De esta forma, la máscara permite usar solo los bits válidos y realizar además la operación de desplazamiento de bits necesaria para obtener el valor correcto.

El tipo de datos asociado a este campo es UInt64B. Su uso es obligatorio.

El servicio EscribirValor no está permitido.

Multiplicador para UI

El valor de la entrada analógica se puede expresar en unidades de ingeniería usando la expresión lineal Valor actual = offset + valor entrada x multiplicador.

El tipo de datos asociado a este campo es UInt16B. Su uso es opcional.

El servicio EscribirValor está permitido.

Offset para UI

Contiene el valor offset de la expresión anterior para la conversión del dato a unidades de ingeniería.

El tipo de datos asociado a este campo es UInt16B. Su uso es opcional.

El servicio EscribirValor está permitido.

Periodo de conversión

Indica el intervalo de tiempo entre conversiones del ADC para obtener un nuevo valor analógico. Se expresa en milisegundos.

El tipo de datos asociado a este campo es UInt16B. Su uso es opcional.

El servicio EscribirValor está permitido.

Multiplicado por

Como vimos en los ejemplos de equipos Modbus, algunas veces los valores de medida son multiplicados o divididos por un factor de 10 para expresar datos no enteros en registros enteros. Este parámetro nos indica el factor multiplicativo que se ha asociado al dato del registro.

El tipo de datos asociado a este campo es UInt16B. Su uso es opcional.

El servicio EscribirValor no está permitido.

Dividido por

Este parámetro nos indica el factor divisor que se ha asociado al dato del registro.

El tipo de datos asociado a este campo es UInt16B. Su uso es opcional.

El servicio EscribirValor no está permitido.

5.6 SALIDA ANALÓGICA

Los objetos de esta clase están asociados a una salida analógica del controlador o a una salida virtual usada como señal de control. La información del valor de la salida, está contenida en el mapa de los HR y se accede a ella a través de la función 6 Modbus para escritura y con la función 3 para lectura.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	

5	Valor actual	Float	x
6	Calidad del dato	Uint8	x
7	Unidades de medida	String	
8	Tiempo mínimo entre cambios	Uint16B	
9	Tipo de dato	Uint16B	x
10	Número de registro HR inicial asociado	Uint16B	x
11	Multiplicador para UI	Uint16B	x
12	Offset para UI	Uint16B	x
13	Multiplicado por	Uint16B	
14	Dividido por	Uint16B	

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Valor actual

Contiene el valor de la salida analógica en unidades de ingeniería. El tipo de datos asociado es Float. Su uso es obligatorio.

El servicio EscribirValor está permitido.

Calidad del dato

Ver la descripción en la clase Entradas Binarias

Unidades de medida

Contiene el texto de las unidades de ingeniería en una cadena de caracteres NULL terminada. El tamaño máximo es de 6 caracteres incluido el Null. Su uso es opcional.

El servicio EscribirValor está permitido.

Tiempo mínimo entre cambios

Este parámetro especifica el tiempo mínimo que se debe mantener el valor del dato de la salida analógica antes de un permitir un nuevo cambio y se expresa en milisegundos. El tipo de datos asociado es UInt16B y su uso es opcional.

El servicio EscribirValor está permitido.

Tipo de dato

Ver la descripción en la clase Entradas Analógicas.

Número de registro HR inicial asociado

Puesto que se trata de una salida analógica, el mapa de registros asociado es HR. Este campo indica el primer registro HR donde está localiza la información del dato.

El tipo de dato asociado es UInt16B y su uso es obligatorio.

El servicio EscribirValor no está permitido.

Multiplicador para UI

Ver la descripción en la clase Entradas Analógicas.

Offset para UI

Ver la descripción en la clase Entradas Analógicas.

Multiplicado por

Ver la descripción en la clase Entradas Analógicas.

Dividido por

Ver la descripción en la clase Entradas Analógicas.

5.7 RELOJ, CALENDARIOS Y HORARIOS

Los horarios son una parte importante del control que permite a un esclavo realizar actuaciones en fechas y horas concretas.

El control de horarios se puede realizar de dos formas:

- Controlado por el maestro
- Autónomo

El sistema controlado por el maestro no precisa de recursos específicos en el esclavo como el reloj de tiempo real. La estructura de funcionamiento es muy simple, consiste en que cada dispositivo tiene propiedades adicionales en sus objetos susceptibles de producir actuaciones como las salidas binarias y las analógicas. Estas nuevas propiedades contienen el valor deseado para la actuación y lo ejecutan al recibir una orden desde el maestro.

Por ejemplo, en la clase salidas binarias se pueden crear 10 propiedades de horario (horario 1 a horario 10), cada una de ellas con un tipo de dato asociado Bool. De la misma manera podemos tener las mismas propiedades en la clase salida analógica pero con datos de tipo Float.

El maestro contendrá una tabla de acciones donde vinculará las fechas y horas a los valores predefinidos de los horarios (1-10). Cuando se tenga una acción programada a una hora determinada, entonces el maestro la enviará a todos los esclavos simultáneamente a través de un mensaje broadcast que contendrá por ejemplo la orden horario 1. Cuando un esclavo reciba este mensaje buscará la propiedad horario 1 de cada uno de sus objetos y establecerá el valor de salida programado previamente para este horario.

Aunque este procedimiento es válido y puede ser aplicado en sistemas Modbus, tiene algunos inconvenientes.

1. Requiere que las comunicaciones estén activas. Si fallan las comunicaciones o el esclavo está aislado no se ejecutarán los horarios.
2. La comunicación tiene que ser broadcast y en Modbus este tipo de comunicación no es con reconocimiento por lo que no tenemos certeza de que el esclavo reciba el mensaje. Si el mensaje no es broadcast, el maestro deberá tener una lista de cuales de los esclavos usan cada uno de los horarios y desencadenaría una comunicación Modbus para cada uno de ellos. Además de la complejidad del sistema, las comunicaciones pueden ser intensivas si el número de esclavos es elevado.

Por las razones anteriores se propone un sistema de horarios autónomo que requiere unos recursos físicos en el esclavo y una estructura de clases adecuadas para esta gestión.

Lo primero que debemos tener para poder ejecutar los horarios es un reloj de tiempo real RTC que nos indique la fecha y hora. Este dispositivo es un periférico del microcontrolador, o periférico independiente, que estará alimentado por una batería auxiliar de forma que incluso con el esclavo apagado se mantenga en funcionamiento.

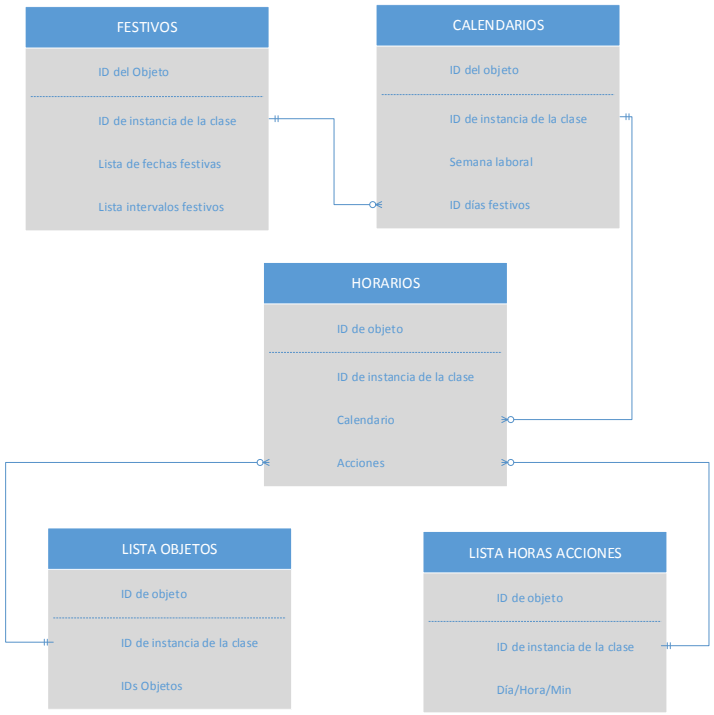
Crearemos una clase RTC que nos permitirá fundamentalmente la puesta en hora y consulta de la fecha y hora.

La ejecución de los horarios de control se realiza fundamentalmente a través de calendarios que nos marcan los intervalos de fecha de validez y la estructura de la semana laboral. La clase CALENDARIO nos permite la creación de los calendarios laborales necesarios.

Disponemos al cabo del año de días festivos y de intervalos festivos que son un conjunto de días consecutivos festivos. La clase FESTIVOS nos permitirá la creación de distintas instancias de días festivos que podremos aplicar a los calendarios. Por ejemplo podemos tener una lista de festivos nacionales, otra de autonómicos y otra de locales y vincular las tres listas de festivos a un mismo calendario. Cuando las fechas festivas sean varias consecutivas se usará el concepto de intervalo festivo. Aunque la definición de los festivos se podría hacer directamente en los calendarios, se ha considerado su definición de forma independiente ya que podemos usar varios calendarios simultáneamente y sería una repetición la definición de la lista de festivos en cada uno de ellos.

El concepto de horarios consiste en la definición de eventos a realizar por el esclavo. Por ejemplo el encendido de una luz, subir una persiana, regular una luz etc. Estos eventos precisan de la especificación del día de la semana y de la hora en que deseamos que se produzca la actuación y de la lista de puntos de control sobre los que queremos que se ejecute la orden. Es importante notar que cada punto de control permite actuaciones distintas, unos son binarios como un apagado/encendido y otros tienen un campo de variación amplio como por ejemplo en una regulación 0-100%.

La clase HORARIOS es la clase principal que permitirá la definición de todos estos eventos y se vinculará a los objetos anteriores.



5.7.1 DÍAS FESTIVOS

Se pueden crear varias listas de festivos. Cada una será una nueva instancia de su clase.

El número total de listas permitidas será dependiente de la implementación concreta en el esclavo y de su memoria disponible. Se ha considerado un vector de tamaño N para la lista de fechas y otro de dimensión M para los intervalos de fechas festivas.

El número de festivos depende de cada año pero en general un rango entre 10 y 14 es suficiente para todos los casos. Los rangos de festivos se aplican a periodos concretos como navidades, semana santa etc. En un centro educativo un periodo festivo clásico sería el de las vacaciones de verano. En general con un máximo de 5 periodos tendremos suficiente para describir estos intervalos.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Número de fechas festivas	UInt8	x
6	Lista Fechas	Fecha[N]	x
7	Número de intervalos festivos	UInt8	x
8	Lista Intervalos	Intervalo Fechas[M]	x

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Número de fechas festivas

Las fechas festivas se especifican por fecha concreta o por intervalos de fechas. Para simplificar la estructura de la clase se han considerado listas de estos elementos. Este campo indica la dimensión de la lista de fechas que sigue. El tipo de datos es UInt8 y su uso es obligatorio.

El servicio EscribirValor está permitido.

Lista Fechas

Este campo es un vector de dimension N que contiene las fechas festivas individuales. Cada elemento del vector es de tipo Fecha. No se admiten caracteres especiales en la representación de la fecha. El valor del año puede ser cero lo que indica que la fecha se repite todos los años durante la existencia del calendario vinculado. Su uso es obligatorio.

El servicio EscribirValor está permitido.

Número de intervalos festivos

Cuando hay más de un día festivo correlativo, en lugar de especificar cada uno individualmente se utiliza un intervalo de fechas que especifica el primer y el último día festivo del intervalo. Para simplificar la estructura de la clase se han considerado listas de estos intervalos. Este campo indica la dimensión de la lista de intervalos de fechas que sigue. El tipo es datos es UInt8 y su uso es obligatorio.

El servicio EscribirValor está permitido.

Lista Intervalos

Este campo es un vector de dimension N que contiene los intervalos de fechas festivas. Cada elemento del vector es de tipo Intervalo Fechas. No se admiten caracteres especiales en la representación de las fecha. Su uso es obligatorio.

El servicio EscribirValor está permitido.

5.7.2 CALENDARIOS

Cada instancia del calendario permite definir un rango de fechas que indican la validez del mismo.

La semana laboral se establece identificando que días de la semana son válidos para el calendario. Se usará un byte como una máscara para marcar los días válidos de la semana. En una oficina, los días válidos de la semana suelen ser de lunes a viernes, pero en un centro comercial suelen ser de lunes a

sábado o incluso los 7 días de la semana. En edificios como hospitales y hoteles son válidos todos los días.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Fecha Inicial	Fecha	
6	Fecha Final	Fecha	
7	Semana Laboral	UInt8	
8	Número de elementos de la lista	UInt8	x
9	ID lista días festivos	Uint16B[N]	

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Fecha Inicial

Si el calendario tiene restringida su validez en el tiempo, es necesario especificar la fecha inicial y la final para las cuales el calendario debe permanecer operativo. Este campo indica la fecha inicial a partir de la cual el calendario es operativo. El tipo de dato asociado es Fecha y su uso es opcional.

El servicio EscribirValor está permitido.

Fecha Final

Este campo indica la fecha final de validez del calendario. A partir de esta fecha el calendario dejará de ser operativo. El tipo de dato asociado es Fecha y su uso es opcional.

El servicio EscribirValor está permitido.

Semana Laboral

Este campo especifica que días de la semana deben ser considerados válidos por el calendario. Según el uso de las instalaciones podrán ser válidos todos los días o puede ser que los fines de semana no este ocupado edificio.

Para indicar la validez de los días usaremos para cada día de la semana un bit de un byte para especificar su validez. Si el bit tiene valor 1 el día señalado es hábil, si tiene valor 0 el día no es hábil.

La representación de los bits en el byte es la siguiente:

Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
-	Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado

El tipo de dato asociado es UInt8 y su uso es opcional.

El servicio EscribirValor está permitido.

Número de elementos de la lista

Los días festivos están representados por instancias de la clase Días Festivos. Esto permite tener distintas listas de fechas festivas reutilizables en distintos calendarios y evita su repetición.

Este campo indica cuantas listas de días festivos usará este calendario. El tipo de datos asociado es UInt8 y su uso es obligatorio.

El servicio EscribirValor está permitido.

ID lista días festivos

Las listas de días festivos se especifican a través del número de instancia de la clase días festivos de un objeto de ese tipo. Esta lista es un vector que contiene todas estas instancias, una por cada lista de festivos que usará el calendario.

El tipo de datos asociado es un vector de UInt16B y su uso es opcional.

El servicio EscribirValor está permitido.

5.7.3 LISTA HORAS ACCIONES

La ejecución de las acciones por horarios, precisa especificar cuándo y que acciones se deben realizar.

Normalmente se indican las fechas y horas de las acciones respecto a los días de la semana domingo a sábado y la hora concreta de la acción. En la mayoría de los casos se indica además la lista de objetos sobre los que se debe ejecutar la acción y el valor asociado a la acción, por ejemplo ON/OFF, regular al 20%.

Se propone separar la lista de objetos de las horas de ejecución de manera que se puedan reutilizar la lista de objetos en distintas acciones sin necesidad de repetirla.

Esta clase define solo la lista de horas de las acciones.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x

2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Número de elementos N de la lista	UInt8	x
6	Día/Hora/Min	UInt16B[N]	x

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Número de elementos N de la lista

Las acciones que se realizan a través del calendario se especifican indicando el día y la hora de la acción, el objeto sobre el que se debe realizar la opción y el valor que hay que transmitir al objeto.

Se han separado la información del día y la hora para permitir reutilizar estas informaciones en varios calendarios y evitar repeticiones. Este campo indica el número de informaciones que contiene la lista siguiente. El tipo de datos del campo es UInt8 y su uso es obligatorio

El servicio EscribirValor está permitido.

Día/Hora/Min

Este campo contiene la lista de horas de las acciones representada como elementos Día/Hora/Min. El valor Día es un valor de 1 a 7 que indica el día de la semana. Su valor se corresponde con la posición del bit ocupado por ese día en el campo semana laboral.

El tipo de datos del campo es un vector de UInt16B y su uso es obligatorio. El formato que ocupan las informaciones en los 16 bits es el siguiente:

it15	it14	it13	it12	it11	it10	it9	it 8	it7	it6	it5	it4	it3	it2	it1	it0
		día			hora					min					

Si el campo de hora tiene el valor 24, los bits 15 y 14 tendrán un significado especial haciendo uso de los valores de la hora que amanece y de la hora que anochece del RTC. El campo minutos indicará entonces los minutos de adelanto o de atraso que modifican la hora del amanecer o del anochecer.

El significado de estos bits es:

Bit 15	Bit 14	Descripción
0	0	Se considera la hora del amanecer. El campo min se resta.
0	1	Se considera la hora del amanecer. El campo min se suma.
1	0	Se considera la hora del anochecer. El campo min se resta.
1	1	Se considera la hora del anochecer. El campo min se suma.

Esta consideración de la hora del amanecer o anochecer es muy necesaria, ya que sus valores cambian de un día a otro. Hay acciones que para favorecer el ahorro energético se deben realizar en esas ocasiones. Por ejemplo las luces exteriores de un edificio se deben encender al anochecer y apagarse al amanecer. El campo minutos en este caso nos permite retocar estos valores horarios para mejorar la funcionalidad de las acciones. Por ejemplo podemos apagar las luces exteriores 15 minutos después del amanecer para esperar a tener un nivel de luz ambiente adecuado y podemos encender la luces exteriores 10 minutos antes del anochecer para evitar tener una exterior insuficiente.

El servicio EscribirValor está permitido.

5.7.4 LISTA OBJETOS

Cada acción que se ejecute a las horas programadas se realiza sobre un grupo homogéneo de objetos. No es posible, por ejemplo dar una orden de regulación a un actuador binario. Estos objetos se deben clasificar en función del tipo de dato utilizado y del valor deseado. Por ejemplo si tenemos un grupo de objetos cuya acción es todo/nada como es el caso de una salida binaria, agruparemos todos los objetos que queremos que actúen simultáneamente con el mismo valor en una lista. Esto simplifica notablemente la gestión horaria ya que la misma lista de objetos se usará tanto para la conmutación ON como para la conmutación OFF.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Número de elementos N de la lista	UInt8	x
6	ID_OBJETOS	UInt16B[N]	x

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Número de elementos N de la lista

Los objetos sobre los que se realizarán las acciones de control se especificarán a través de una lista que contendrá los identificadores globales de los objetos. Este campo nos indica el número de elementos de esa lista de objetos. El tipo de datos del campo es UInt8 y su uso es obligatorio

El servicio EscribirValor está permitido.

ID_OBJETOS

Este campo contiene la lista de identificadores globales de los objetos sobre los que se realizarán las acciones de control. Cada elemento de la lista es de tipo UInt16B. El uso de este campo es obligatorio.

El servicio EscribirValor está permitido.

5.7.5 HORARIOS

En la definición de los eventos, se suelen crear espacios para su definición para cada día de la semana. Desde el punto de vista de la implementación esto obliga a listas separadas para cada día de la semana así que se ha considerado mejor opción especificar el día de la semana dentro de cada evento. Esto permite un mejor aprovechamiento de la memoria del esclavo.

Si el número de acciones a realizar es elevado, se pueden crear varios horarios todos ellos activos y basados en el mismo calendario que se ejecutarán simultáneamente y se comportarán como un único horario.

Cada acción de control consta de las siguientes informaciones: ID lista de objetos, Listado horas, Tipo de dato y Valor. El campo ID lista de objetos es una instancia de la clase Lista Objetos, Listado horas es una instancia de la clase ListaHorasAcciones, el tipo de dato indica el tipo asociado al dato que sigue y Valor contiene el valor de control de la acción. El tamaño del Valor es variable, se considera para su almacenamiento un tamaño de 8 bytes que el tamaño mayor de los tipos de datos numéricos.

La información de una acción de control debe ser completa, así que los cuatro datos forman una unidad de información que llamaremos ACCIÓN.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Activo	UInt8	x
6	Calendario asociado	UInt16B	x
7	Número de elementos de la lista de excepciones	UInt8	x

8	Lista días excepciones	Fecha[M]	
9	Número de acciones N	UInt8	x
10	Lista de acciones de control	ACCIÓN[N]	x

Aunque el tipo de dato de las propiedades Valor X es variable, se reservará siempre el tamaño del mayor valor posible es decir 8 bytes (4 registros de 16 bits).

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Activo

Este campo indica si el calendario está activo y las acciones de control deben ejecutarse, o por el contrario si no debemos ejecutar esas acciones. El tipo de datos asociado es Bool y el valor 1 indica calendario activo. Su uso es obligatorio.

El servicio EscribirValor está permitido.

Calendario asociado

Este parámetro nos indica el calendario que debemos seguir para la ejecución de las acciones. El calendario se especifica por su número de instancia dentro de su clase. El tipo de datos asociado es UInt16B y su uso es obligatorio.

El servicio EscribirValor está permitido.

Número de elementos de la lista de excepciones

Indica el tamaño de la lista de días de excepciones que sigue expresado en número de elementos. El tipo de datos asociado es UInt8 y su uso es obligatorio.

El servicio EscribirValor está permitido.

Lista días excepciones

Este campo contiene una lista de fechas individuales que se considerarán como excepciones a las fechas de funcionamiento expresadas en el calendario. Durante estos días de excepciones no se ejecutarán las acciones programadas. Cada elemento es de tipo Fecha y su uso es opcional.

El servicio EscribirValor está permitido.

Número de acciones

Indica el número de acciones que se quieren ejecutar. Cada acción requiere la siguiente información:

- Identificador de la lista objetos
- Identificador de la Lista de horas
- Tipo dato
- Valor

Cuando se ejecutan las acciones, unas de otras se distinguirán por las horas a las que queremos que se ejecute la acción, pero incluso a las mismas horas unos objetos precisarán una orden de control distinta en función de las necesidades del control y de las características del objeto a controlar (binario, analógico etc.). El tipo de datos asociado es UInt8 y su uso es obligatorio.

El servicio EscribirValor está permitido.

Lista Acciones de control

Contiene una lista con las acciones de control deseadas. Cada elemento de la lista es de tipo ACCIÓN y su tamaño es bytes es de 14 bytes. Su uso es obligatorio.

El servicio EscribirValor está permitido.

5.7.6 RTC

La clase RTC gestionará el reloj de tiempo real del esclavo. Existirá una sola instancia de esta clase que estará asociada al periférico del esclavo que realiza esta función.

El campo valor contendrá la información de la fecha y la hora y los servicios de LeerValor y EscribirValor permitirán la lectura y la puesta en hora respectivamente.

ID_PROP	NOMBRE	TIPO DATO	OBLIGATORIA
1	Identificador del objeto	Uint16B	x
2	Nombre del objeto	String	x
3	Número de instancia de la clase	Uint16B	x
4	Descripción del objeto	String	
5	Valor	Fecha-Hora	x
6	ValorUTC	Fecha-Hora	x
7	Longitud	Float	
8	Latitud	Float	
9	Offset UTC	Int16B	
10	Hora amanecer	Hora	
9	Hora anochecer	Hora	
10	Cambio de hora automático inv/verano	Bool	

Identificador del objeto

Ver la descripción en la clase Entradas Binarias

Nombre del objeto

Ver la descripción en la clase Entradas Binarias

Número de instancia de la clase

Ver la descripción en la clase Entradas Binarias

Descripción del objeto

Ver la descripción en la clase Entradas Binarias

Valor

Contiene la información de la fecha y la hora del reloj RTC. Este valor de hora se puede leer y se puede también cambiar para realizar la puesta en hora correcta del RTC. El tipo de dato es Fecha-Hora y su uso es obligatorio.

El servicio EscribirValor está permitido.

ValorUTC

Contiene la información de la fecha y hora referenciada al meridiano cero. Se usa a los efectos de sincronización de fecha y hora de dispositivos en distintas ubicaciones geográficas. El tipo de dato es Fecha-Hora y su uso es obligatorio.

El RTC trabaja siempre internamente con este formato. Si se define el campo OffsetUTC, el RTC calculará la fecha y hora en formato local que estará contenida en el campo Valor. Si no está definida OffsetUTC, Valor y ValorUTC coincidirán.

El servicio EscribirValor está permitido.

Longitud

Contiene el valor de la Longitud geográfica expresada como un dato Float. Esta coordenada se debe corresponder con las de la instalación realizada. Su uso es opcional.

El servicio EscribirValor está permitido.

Latitud

Contiene el valor de la Latitud geográfica expresada como un dato Float. Esta coordenada se debe corresponder con las de la instalación realizada. Su uso es opcional.

El servicio EscribirValor está permitido.

Offset UTC

Indica el offset en minutos entre la hora local y la hora de referencia del meridiano universal. Los minutos se consideran positivos al oeste y negativos al este y su tango es de -780 a +780 minutos

Hora amanecer

Este campo contiene la hora a la que amanece el día actual. Este valor se calcula automáticamente por el esclavo y se podrá usar después en los horarios de control. El tipo de dato es Hora y su uso es opcional.

Solo es posible el cálculo de esta hora si previamente se han especificado los campos de Longitud y Latitud anteriores.

El servicio EscribirValor no está permitido.

Hora anochecer

Este campo contiene la hora a la que anochece el día actual. Este valor se calcula automáticamente por el esclavo y se podrá usar después en los horarios de control. El tipo de dato es Hora y su uso es opcional.

Solo es posible el cálculo de esta hora si previamente se han especificado los campos de Longitud y Latitud anteriores.

El servicio EscribirValor no está permitido.

Cambio de hora automático inv/verano

Esta propiedad indica si se debe realizar automáticamente el cambio de hora de verano a invierno y de invierno a verano. El tipo de datos es Bool y su uso es opcional. Un valor 1 en esta propiedad significa que se debe realizar el cambio automático. El servicio EscribirValor no está permitido.

6 OBTENCIÓN DE LOS RECURSOS DE LOS ESCLAVOS

6 OBTENCIÓN DE LOS RECURSOS DE LOS ESCLAVOS

Las informaciones que un esclavo debe notificar al maestro en base a que este obtenga las informaciones sobre sus recursos son:

- Informaciones generales del dispositivo
- Informaciones del número de objetos implementados y sus identificadores
- Informaciones de detalle de cada instancia u objeto

Tenemos dos posibilidades para la identificación de un objeto perteneciente a un dispositivo esclavo.

La primera es hacerlo a través de un doble valor clase-instancia. El número de instancia dentro de una clase debe ser único pero puede existir el mismo valor en otra clase. Los números de instancia no tienen que ser obligatoriamente consecutivos facilitando así al desarrollador criterios numéricos de asignación de instancias en función por ejemplo del uso del objeto.

Un segundo criterio de identificación de un objeto es a través de un identificador único del objeto, sin distinción de clases. Dentro de las propiedades del objeto se especificará la clase a la que pertenece y un número de instancia que permite diferenciar a una instancia de otra dentro de una clase.

Independientemente del uso de cualquiera de las dos notaciones, ambas válidas, el problema principal que debemos resolver es el conocimiento de cuáles son los objetos que están disponibles en un dispositivo.

Si usamos la identificación del objeto por un código único, la estrategia de intercambio de información entre el maestro y el esclavo podría ser la siguiente:

Maestro->pide identificación del primer objeto

Esclavo-> responde con el número del primer objeto disponible

Bucle

Maestro->pide identificación del siguiente objeto

Esclavo-> responde con el número del siguiente objeto disponible

Mientras objeto disponible

El primer objeto disponible en un esclavo es el objeto "dispositivo", por lo que siempre debe haber al menos un objeto en la respuesta del esclavo.

El maestro guardará la lista de objetos y posteriormente interrogará individualmente a cada objeto para recoger las informaciones del detalle de cada uno.

Si usamos el criterio de identificación a partir de la clase-instancia, deberemos conocer en primer lugar las clases implementadas en el dispositivo.

Esto se podría hacer, creando en el objeto dispositivo un atributo tipo lista con la identificación de cada una de las clases implementadas en el esclavo.

Otra posibilidad consiste en hacer al esclavo una serie de peticiones para que nos devuelva los identificadores de las clases implementadas. Teniendo en cuenta que la información del propio esclavo debe residir en una clase dispositivo el flujo de información maestro esclavo sería:

Maestro->pide identificación de la primera clase

Esclavo-> responde con el número de la primera clase disponible

Bucle

Maestro->pide identificación de la primera instancia de esa clase

Esclavo-> responde con el número de la primera instancia disponible

Bucle

Maestro->pide identificación de la siguiente instancia

Esclavo-> responde con el identificador de la siguiente instancia disponible

Mientras instancia disponible

Maestro->pide identificación de la siguiente clase

Esclavo-> responde con el identificador de la siguiente clase disponible

Mientras clase disponible

El primer procedimiento es más simple y la información clase-instancia está contenida en las propiedades del propio objeto.

La función 91 no es válida entonces para el intercambio de información y procederemos a crear las funciones alternativas necesarias.

6.1 FUNCIÓN 103: OBTENCIÓN DE LOS IDENTIFICADORES DE LOS OBJETOS

Creamos la función 103 para que el maestro solicite a los esclavos los identificadores de los objetos que contiene. La función invoca a un esclavo por su dirección Modbus de forma que previamente hemos tenido que asignar las direcciones a los esclavos. El formato de la nueva función 103 para la petición y para la respuesta se describe a continuación.

Petición de identificadores de los objetos

BYTES	DESCRIPCIÓN
Xx	Dirección del esclavo interrogado
103	Código de función de petición de objetos
ID_OBJ	Objeto solicitado.
ID_OBJ	
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

El campo ID_OBJ de identificación del objeto es de 16 bits (2 bytes) y usa el formato big-endian clásico de Modbus. El esclavo será el que nos envíe dichos códigos.

El primer objeto disponible en un esclavo es siempre la instancia de la clase "dispositivo" que corresponde a cada esclavo. Por convenio los objetos se numeran empezando por el 1, así que siempre el objeto 1 de un esclavo es la instancia de la clase dispositivo.

Si utilizamos un código en el campo ID_OBJ para indicar la solicitud del primer objeto (PRIMER_OBJETO) y después otro valor para indicar la solicitud del siguiente objeto (SIGUIENTE_OBJETO), se puede producir un problema de sincronización en la comunicación en determinadas circunstancias.

Maestro-> pide PRIMER_OBJETO

Esclavo-> responde con el identificador del primer objeto (**objeto 1**)

Maestro-> pide SIGUIENTE_OBJETO

Esclavo-> responde con el identificador del siguiente objeto (**objeto 2**)

Si se produce un error de recepción de este mensaje en el maestro, este intentará repetir la petición pero puesto que el esclavo había recibido correctamente la petición anterior el flujo de mensajes será ahora:

Maestro-> pide SIGUIENTE_OBJETO

Esclavo-> responde con el identificador del siguiente objeto (**objeto 3**)

Como vemos se ha perdido la información correspondiente al objeto 2.

Para resolver este problema modificaremos ligeramente el concepto del campo ID_OBJ de forma que el maestro indicará en la petición el código del último objeto conocido y el esclavo enviará en la respuesta el código del siguiente objeto disponible.

Puesto que el código del primer objeto es el 1, cuando el esclavo reciba en la petición el código 0, significará que se está solicitando el primero objeto del esclavo.

El flujo de petición respuesta anterior se modifica ahora como se indica a continuación.

Maestro-> pide siguiente a **objeto 0**

Esclavo-> responde con el identificador del primer objeto (**objeto 1**)

Maestro-> pide siguiente a **objeto 1**

Esclavo-> responde con el identificador del **objeto 2**

Se produce el error de comunicación y el maestro repite la petición.

Maestro-> pide siguiente a **objeto 1**

Esclavo-> responde con el identificador del **objeto 2**

Ahora la respuesta recibida por el maestro es correcta y no se ha perdido ningún objeto.

Cuando no exista el objeto siguiente al solicitado por ser el último disponible en el esclavo, este devolverá el valor 0 en la respuesta.

Respuesta de identificadores de los objetos

BYTES	DESCRIPCIÓN
Xx	Dirección del esclavo interrogado
103	Código de función de petición de objetos
ID_OBJ	Código del objeto devuelto.
ID_OBJ	
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

Como vemos el proceso de identificación de los códigos de los objetos funcionará correctamente pero plantea todavía un problema de eficiencia en las comunicaciones ya que necesitamos una petición-respuesta para la identificación de cada objeto. Este hecho contrasta claramente con la filosofía del protocolo Modbus de peticiones múltiples mostrada en las funciones 1 a 4 de dicho protocolo.

Para optimizar el flujo de información para la obtención de los identificadores de los objetos, vamos a retocar la respuesta a la función 103 para permitir el envío de múltiples identificadores en cada respuesta del esclavo. La nueva estructura de la respuesta es ahora la siguiente:

BYTES	DESCRIPCIÓN
Xx	Dirección del esclavo interrogado
103	Código de función de petición de objetos
N	Número de bytes de los identificadores. N=número de identificadores * 2
PEND	Número de identificadores pendientes después de este mensaje
PEND	
ID_OBJ 1	Identificador del primer objeto devuelto
ID_OBJ 1	
….	
ID_OBJ N	Identificador del objeto N devuelto
ID_OBJ N	
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

Puesto que el tamaño de la trama está limitado a 256 bytes, el número máximo de identificadores de objetos que se pueden enviar en una respuesta es de 256-7/2 =124.

Cada identificador de objeto necesita 16 bits (2 bytes). El primer identificador recibido es el siguiente al que figura en la petición del maestro.

El campo PEND indica los identificadores pendientes de enviar una vez que se complete la transmisión del mensaje.

Las comunicaciones maestro-esclavo son totalmente consistentes en un entorno multimaestro, ya que la petición del maestro indica siempre el identificador de referencia a partir del cual se enviarán los resultados. Esto significa que si dos maestros realizan peticiones entremezcladas de la misma función y no sincronizadas, el esclavo responderá correctamente a cada una de las peticiones de los respectivos maestros.

Si la respuesta contiene 124 identificadores y PEND>0, significa que existen más identificadores pendientes de notificar y el maestro realizará otra petición enviando como código del identificador el del último recibido. Si no hay más identificadores disponibles, el valor PEND de la respuesta valdrá cero.

Si el esclavo detecta un error en la información suministrada en la petición del maestro responderá con un código de función de error 128+102=230. Al código de función le seguirá un byte que contiene el código del error detectado.

Los códigos válidos de error para la función 103 son:

Código	Descripción
1	Función no soportada
2	Identificador de objeto inexistente

6.2 FUNCIÓN 104: ACCESO A LAS PROPIEDADES DE LOS OBJETOS

Una vez obtenidos los identificadores de los objetos, el maestro puede interactuar con las propiedades del objeto a través de los servicios implementados.

Aunque un maestro pueda tener conocimiento de las propiedades de cada clase, puede ocurrir que una diferencia de versiones entre maestro y esclavo impida conocer una propiedad más reciente en la versión del esclavo. Para tener acceso a todas las propiedades, tendremos en cuenta en las funciones de acceso, tanto la identificación concreta de una propiedad por su identificador como la búsqueda secuencial de las propiedades.

Usaremos la nueva función 104 para acceder a la información de las propiedades de un objeto. La identificación del servicio indicará al esclavo la función que tiene que realizar y el tratamiento que tiene que hacer con los datos.

Usaremos dos bytes para especificar el servicio solicitado a la propiedad. Puesto que cada propiedad tendrá habilitados de manera individual los servicios, y estos deben ser conocidos por el maestro, limitaremos a 16 el número máximo de servicios lo que permitirá identificar los servicios habilitados de una propiedad con un registro de 16 bits en el que cada bit estará asociado a uno de los servicios. Si el bit tiene el valor 1 el servicio está disponible para la propiedad. Si el bit tiene valor 0 el servicio no está disponible.

Los códigos válidos de servicios son:

Código	Servicio	Descripción
0	BuscarPropiedades	Busca las propiedades disponibles
1	LeerPropiedad	Lee la información de la propiedad
2	LeerPropiedades	Lee la información de varias propiedades
3	LeerValor	Lee el campo VALOR de la propiedad
4	EscribirValor	Modifica el campo VALOR de la propiedad

Todas las propiedades deben permitir el servicio LeerPropiedad y LeerValor. El resto de los servicios son opcionales para cada propiedad y se notificará su disponibilidad al maestro dentro de la respuesta a la ejecución del servicio LeerPropiedad.

El servicio EscribirValor precisa de datos adicionales para enviar a la propiedad. El tamaño del dato enviado debe ser compatible con el tipo de dato definido para el atributo. Si el dato no es compatible se generará un error en la respuesta.

Los servicios equivalentes a Reset, Online y Offline se establecerán a través de la ejecución del servicio EscribirValor a propiedades creadas al efecto en la clase dispositivo.

La estructura de la petición y respuesta se muestra a continuación.

Petición función 104

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo interrogado
104	Código de función de acceso a las propiedades
ID_OBJ	Identificador del objeto direccionado
ID_OBJ	
ID_PROP	Identificador de la propiedad direccionada
ID_PROP	
S	Identificador del servicio de acceso a la propiedad
N	Número de bytes de datos que siguen sin incluir los dos bytes de CRC
TYPE	Tipo del dato que sigue
…	Datos si proceden
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

Respuesta función 104

BYTES	DESCRIPCIÓN
xx	Dirección del esclavo
104	Código de función de acceso a las propiedades
ID_OBJ	Objeto direccionado
ID_OBJ	
ID_PROP	Identificador de la propiedad
ID_PROP	
S	Identificador del servicio de acceso a la propiedad
N	Número de bytes del bloque de datos que sigue
…	Datos si proceden
CRC-LO	Byte bajo CRC
CRC-HIGH	Byte alto CRC

Los campos de datos contienen la información de petición del maestro y de respuesta del esclavo. Su tamaño máximo es de 246 bytes.

Cada propiedad se identifica por un código ID_PROP que es único dentro del objeto y de tamaño 16 bits. Los códigos válidos son del 1 al 65535. La propiedad 1 siempre existe en un objeto y contiene un identificador del objeto.

Cada mensaje de petición y de respuesta de la función 104 debe caber en una trama cuyo máximo tamaño es de 256 bytes. Esto hace que el tamaño máximo del campo de datos sea de 246 bytes en un mensaje.

El campo TYPE de la petición solo se incluye si le siguen los datos.

En caso de que los datos de petición del maestro no sean correctos, el esclavo responderá con un mensaje de error usando el código de función 128+103=231. Este procedimiento es similar al del resto de las funciones Modbus. Al byte del código de la función le sigue un byte que identifica el error detectado.

Los códigos de error permitidos son los siguientes:

Código	Descripción
1	Función no soportada
2	Identificador de objeto inexistente
3	Identificador de propiedad inexistente
4	Servicio no permitido
4	Formato de datos erróneo

El código 2 se enviará cuando el identificador del objeto no exista en el esclavo.

El código 3 se enviará cuando el identificador de la propiedad no exista para el objeto indicado en el esclavo.

El código 4 se devolverá cuando el servicio no esté permitido para esa propiedad.

El código 5 se enviará cuando al invocar al servicio EscribirValor, el dato que sigue no es compatible con el tipo de datos definido en la propiedad. Por ejemplo si el tipo de datos de la propiedad es float (4 bytes) y el tipo especificado para la escritura es un entero de 16 bits se devolverá el código de error 3.

Servicio BuscarPropiedades

Este servicio permite obtener del esclavo la lista de identificadores de propiedades implementadas. Esto nos permitirá construir todo el árbol de propiedades de un objeto sin ningún conocimiento previo del mismo. No se exige al objeto que los identificadores de las propiedades implementadas sean consecutivos. En función del número de propiedades, puede hacer falta usar este servicio repetidas veces para obtener la lista completa de identificadores. El campo ID_PROP indicará a partir de que propiedad se realizará la búsqueda de propiedades. Para la primera búsqueda usaremos el valor 0 en el campo ID_PROP para que el primer identificador de propiedad devuelvo sea el de la propiedad 1. En sucesivas peticiones se utilizará el identificador de la última propiedad devuelta en la respuesta. La petición es autoconsistente y compatible con una estructura multimaestro.

No existen datos adicionales en la petición por lo que el campo N tendrá el valor 0.

El bloque de datos de la respuesta contiene los identificadores de las propiedades del objeto usando 16 bits para cada identificador. Los dos primeros bytes indican el número de identificadores pendientes de transmitir después de este mensaje.

DATOS	
PEND	Identificadores pendientes de transmitir
PEND	
ID_PROP1	Identificador de la propiedad 1
ID_PROP1	
…	
ID_PROPN	Identificador de la propiedad N
ID_PROPN	

Servicio LeerPropiedad

Permite solicitar al esclavo la información asociada a una propiedad cuyo identificador es conocido y está contenido en el campo ID_PROP.

No existen datos adicionales en la petición por lo que el campo N tendrá el valor 0.

En la respuesta el campo de datos contiene la información de la propiedad. Cada propiedad del objeto consta de las siguientes informaciones:

Campo	**Tamaño del campo en bytes**	**Descripción**
ID	2	Identificador de la propiedad (dentro de la clase)
SERV	2	Servicios permitidos (máscara de bits)
TIPO	2	Tipo de dato asociado
VALOR	Máximo 226 bytes	Valor de la propiedad

Si el tipo de datos del campo VALOR es una cadena de caracteres, esta debe usar el terminador NULL en la misma. Aunque el tamaño máximo del campo VALOR es de 226 bytes, en muchos casos su tamaño será mucho menor y delimitado por el tipo de dato asociado. Por ejemplo para un entero de 16 bits se usarán solo dos bytes.

La propiedad 1 de cada objeto será siempre un identificador del objeto y el tipo de datos asociado será por tanto UInt16B.

En el caso de que el tipo de datos de la propiedad sea una cadena de caracteres, su longitud queda especificada por su terminador NULL y por el valor del campo N de la respuesta.

Servicio LeerPropiedades

Ya que el tamaño de la información de una propiedad puede ocupar solo unos pocos bytes, el servicio LeerPropiedades permitirá recoger la información de varias propiedades en una sola respuesta y mejorar así la eficiencia de las comunicaciones en la fase de configuración de la red.

La característica de lectura de múltiples propiedades debe ser del esclavo y por tanto debe estar habilitada de la misma manera para todas las propiedades de todos los objetos.

El campo ID_PROP indicará la primera propiedad de la que se solicita su información. Ya que el tamaño de la información de cada propiedad es variable, será el esclavo el que determine en función del tamaño de cada una, el número de propiedades cuya información devolverá en cada mensaje.

Este servicio se deberá usar repetidas veces hasta completar la recogida de información de todas la propiedades.

No existen datos adicionales en la petición por lo que el campo N tendrá el valor 0.

Con este servicio se leen todos los campos de las propiedades que quepan dentro de un mensaje. El campo N de la respuesta indicará el tamaño en bytes de los datos de todas las propiedades.

El campo de datos contiene las siguientes informaciones en la respuesta:

DATOS	**N Bytes**	
N1	1	Número de bytes ocupados por la propiedad
ID_PROP1	2	Identificador de la propiedad
SERV1	2	Servicios permitidos (máscara de bits)
TIPO1	2	Tipo de dato asociado
VALOR1	Max 226	Valor de la propiedad
…		
Nn	1	Número de bytes ocupados por la propiedad
ID_PROPn	2	Identificador de la propiedad
SERVn	2	Servicios permitidos (máscara de bits)
TIPOn	2	Tipo de dato asociado
VALORn	Max 226	Valor de la propiedad

Los datos de cada propiedad están precedidos de un campo Ni que indica el tamaño en bytes de cada propiedad. La suma total de los bytes de datos, incluidos los Ni no puede superar los 246 bytes.

Servicio LeerValor

Este servicio indica al esclavo que se quiere recibir la información solo del campo VALOR de la propiedad. Su uso está previsto fundamentalmente para poder recoger el valor de una información variable en el esclavo como por ejemplo una señal digital o analógica. Representa una alternativa al uso de las funciones 1, 2, 3 y 4 del protocolo Modbus, aunque solo es posible solicitar el valor de una única propiedad. Puede resultar interesante para el uso de datos complejos y para funciones de test durante la instalación.

Si el campo valor es demasiado grande y no cabe en el mensaje de respuesta, el maestro necesitará hacer varias peticiones que deben estar identificadas para evitar repeticiones. Para ello el campo N de la petición tomará el valor 1 para indicar que sigue un dato en el campo valor.

El campo valor tomará el valor 1 para indicar que es la primera petición y el valor devuelto por el campo MENSAJE de la respuesta anterior en la siguiente petición, si quedan datos por recibir.

Con este servicio se lee la información del tipo de datos de la propiedad y del campo VALOR. La información devuelta en el bloque de datos de la respuesta es la siguiente:

DATOS	N Bytes	
MENSAJE	2	Número de envío de datos
TIPO	2	Tipo de dato asociado
VALOR	Max 224	Valor de la propiedad

El campo mensaje contiene el número de orden de la respuesta. La parte alta de este campo toma el valor 1 para indicar que hay más datos pendientes y el valor 0 para indicar que no quedan datos para enviar.

Servicio EscribirValor

Este servicio permite modificar el campo VALOR de una propiedad. Parte de las propiedades de un objeto son de identificación y no permiten modificar el campo VALOR, sin embargo otras permiten modificar este campo para cambiar opciones de configuración o para modificar el estado o valor de las salidas del esclavo. Un ejemplo puede ser una salida analógica o la velocidad de las comunicaciones del puerto RS485.

La operación de escritura solo se puede realizar sobre una única propiedad y es equivalente a las funciones 5 y 6 de Modbus con la diferencia de que usa datos con tipo.

El campo N de la petición indicará al esclavo la longitud del bloque de datos que sigue. El bloque de datos contiene dos bytes que indican el tipo de dato adjunto seguido de los bytes que contienen el valor del dato. El esclavo comprobará la compatibilidad del tipo de dato con el de la propiedad y generará un error en caso de que sean distintos.

El campo de datos de la petición contiene:

DATOS	N Bytes	
MENSAJE	2	Número de envío de datos
TIPO	2	Tipo de dato asociado
VALOR	Max 224	Valor de la propiedad

El campo MENSAJE es un número de orden asociado al mensaje enviado por el maestro. Algunas propiedades precisan muchos datos en su campo valor que no caben en un solo mensaje, así que este campo indica el número de orden de la secuencia de datos.

El primer bloque de datos tiene un valor de 1 en el campo MENSAJE. Siempre que se reciba el número 1, el esclavo procederá al borrado del contenido valor y escribirá el nuevo valor que acaba de recibir. Si el valor de MENSAJE es distinto de 1, el esclavo añadirá los datos recibidos a los que ya tiene almacenados. Si se produce una repetición del mensaje, el esclavo responderá con normalidad pero ignorará los datos recibidos.

Puesto que el bloque de datos puede tener muchos bytes, en el mensaje de respuesta no se devuelve el campo VALOR sino solo el TIPO de datos asociado

El campo de datos de la respuesta es el mismo que el de la petición por lo que el mensaje de respuesta es idéntico al de petición.

DATOS	**N Bytes**	
TIPO	2	Tipo de dato asociado

7 VALIDACIÓN DE LAS EXTENSIONES DEL PROTOCOLO

7 VALIDACIÓN DE LAS EXTENSIONES DEL PROTOCOLO

El modelo de objetos presentado, precisa de su validación, para poder incorporarse en el diseño de dispositivos MODBUS. La implementación del nuevo protocolo en un esclavo requiere del análisis y la comprobación de los siguientes aspectos:

- Necesidades de almacenamiento de la información de cada objeto
- Cálculo de las necesidades de memoria de un esclavo MODBUS
- Ubicación de la memoria de los objetos en los mapas de registros y su acceso a través de las funciones clásicas de Modbus y de las funciones ampliadas

7.1 ALMACENAMIENTO DE LA INFORMACIÓN DE LOS OBJETOS

Cada objeto perteneciente a una clase necesita de un espacio de almacenamiento de memoria para cada una de sus propiedades. Es necesario por tanto analizar cada clase y cada una de sus propiedades para obtener la información de la memoria máxima requerida para su almacenamiento.

Esta memoria máxima asume que todas las propiedades de las clases se están utilizando en el objeto. Evidentemente es posible que en una implementación dada, algunas de las propiedades no se usen y que en consecuencia las necesidades de almacenamiento para el objeto se reduzcan.

Consideraremos en el estudio dos valores de memoria requeridos para el almacenamiento del objeto:

- La memoria mínima necesaria
- La memoria máxima necesaria

La memoria mínima es la memoria necesaria para que un objeto contenga toda la información clasificada como obligatoria y ninguna opcional.

Se tendrá que permitir posteriormente que el acceso a estas informaciones se realice a través de registros de 16 bits y por las funciones clásicas de Modbus. Por ello ajustaremos el tamaño de memoria de cada propiedad a múltiplos de 16 bits.

Procedemos con cada clase a ordenar su lista de propiedades en base al campo OBLIGATORIA. Añadimos también una quinta columna que contendrá el número de registros de 16 bits necesarios para almacenar solo el campo VALOR de la propiedad ya que el resto del almacenamiento requerido es una constante para todas las propiedades.

Recordamos que cada propiedad precisa para su almacenamiento de los siguientes campos:

- ID que usa 2 bytes
- Servicios disponibles que usa 2 byte
- Tipo de datos que usa 2 bytes
- Valor que es de longitud variable en función del tipo de dato usado.

Campo	Bytes	Registros
ID	2	1
Servicios	2	1
TipoDatos	2	1
VALOR	var	var

Usaremos entonces 3 registros de 16 bits por cada propiedad más los que requiera el campo VALOR que es función del tipo de datos almacenado.

7.1.1 CLASE DISPOSITIVO

La tabla ordenada y ampliada queda:

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros
1	Identificador del objeto	Uint16B	x	1
2	ID 48 bits	UInt16B[3]	x	3
3	Nombre del dispositivo	String	x	8
4	Número de instancia de la clase	Uint16B	x	1
7	Clases implementadas	UInt64B	x	4

8	Número de objetos en el esclavo	UInt16B	x	1	
9	Propiedades Múltiples habilitadas	Bool	x	1	
10	Fabricante	String	x	10	
11	Código de producto	String	x	10	
12	Revisión	String	x	10	
17	Versión de firmware	String	x	10	
18	Versión del protocolo	String	x	3	
19	Reset	Bool	x	1	
20	Offline	Bool	x	1	
21	Online	Bool	x	1	65
5	Descripción del dispositivo	String		20	
6	Localización del dispositivo	String		20	
13	URL del vendedor	String		10	
14	Nombre del producto	String		10	
15	Nombre del modelo	String		10	
16	Nombre de la aplicación	String		10	80

El número de registros de los campos obligatorios está marcado con color más claro y los de los campos opcionales en color más oscuro.

El total de registros necesarios para el almacenamiento de un objeto es de 65+15*3=110 para las propiedades obligatorias y 80+6*3=98 para las opcionales. Por tanto el máximo es de 208 registros.

7.1.2 ENTRADA BINARIA

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	x	1	
2	Nombre del objeto	String	x	8	
3	Número de instancia de la clase	Uint16B	x	1	
5	Valor actual	Bool	x	1	
6	Calidad del dato	Uint8	x	1	
7	Polaridad	Bool	x	1	
10	Mapa de registros	Uint8	x	1	
11	Número de registro	Uint16B	x	1	15
4	Descripción del objeto	String		20	
8	Texto para valor Activo	String		8	
9	Texto para valor Inactivo	String		8	
12	Número de bit del registro	Uint8		1	
13	Valor registro para Activo	Uint16B		1	
14	Valor registro para Inactivo	Uint16B		1	
15	Tiempo mínimo estable para cambio estado	Uint16B		1	
16	Contador de cambios de estado	Uint32B		2	
17	Fecha inicio contador cambios de estado	Fecha y Hora		3	
18	Borrado contador cambios estado	Bool		1	
19	Fecha del último cambio estado	Fecha y Hora		3	49

El total de registros necesarios para el almacenamiento de un objeto es de 15+8*3=39 para las propiedades obligatorias y 49+11*3=82 para las opcionales. Por tanto el máximo es de 121 registros.

7.1.3 SALIDA BINARIA

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	x	1	
2	Nombre del objeto	String	x	8	
3	Número de instancia de la clase	Uint16B	x	1	
5	Valor actual	Bool	x	1	
6	Calidad del dato	Uint8	x	1	
7	Polaridad	Bool	x	1	
10	Mapa de registros	Uint8	x	1	
11	Número de registro	Uint16B	x	1	15
4	Descripción del objeto	String		20	
8	Texto para valor Inactivo	String		8	
9	Texto para valor Activo	String		8	
12	Número de bit del registro	Uint8		1	
13	Valor registro para Inactivo	Uint16B		1	
14	Valor registro para Activo	Uint16B		1	
15	Tiempo mínimo estable antes de cambio	Uint16B		1	
16	Contador cambios de estado	Uint32B		1	
17	Fecha inicio contador cambios de estado	Fecha y Hora		3	
18	Borrado contador cambios de estado	Bool		1	
19	Fecha último cambio de estado	Fecha y Hora		3	
20	ID de la entrada Binaria de estado asociada	Uint16B		1	
21	Alarma Estado	Bool		1	
22	Retraso ON	Uint16B		1	
23	Retraso OFF	Uint16B		1	
24	Duración pulso ON-OFF	Uint16B		1	
25	Persistencia	Bool		1	
26	ID de señal de habilitación	Uint16B		1	
27	Valor estado para habilitación	Bool		1	
28	ID de señal de bloqueo	Uint16B		1	

29	Valor estado bloqueo	Bool		1	58

El total de registros necesarios para el almacenamiento de un objeto es de 15+8*3=39 para las propiedades obligatorias y 58+21*3=121 para las opcionales. Por tanto el máximo es de 160 registros.

7.1.4 CONTADOR

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	X	1	
2	Nombre del objeto	String	X	8	
3	Número de instancia de la clase	Uint16B	x	1	
5	Valor actual	Uint32B	x	2	
6	Calidad del dato	Uint8	x	1	
7	Polaridad pulso	Bool	x	1	
11	Número de registro para lectura valor actual	Uint16B	x	1	15
4	Descripción del objeto	String		20	
8	Duración mínima del pulso	Uint32B		2	
9	Valor actual en unidades de ingeniería	UInt32B		2	
10	Unidades de medida	String		3	
12	Número de registro para lectura valor ingeniería	Uint16B		1	
13	Multiplicador	Uint16B		1	
14	Divisor	Uint16B		1	30

El total de registros necesarios para el almacenamiento de un objeto es de 15+8*3=39 para las propiedades obligatorias y 30+7*3=51 para las opcionales. Por tanto el máximo es de 90 registros.

7.1.5 ENTRADA ANALÓGICA

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros
1	Identificador del objeto	Uint16B	x	1
2	Nombre del objeto	String	x	8
3	Número de instancia de la clase	Uint16B	x	1
5	Valor actual	Float	x	2
6	Calidad del dato	Uint8	x	1

8	Tipo de dato	Uint16B	x	1	
9	Mapa de registros	Uint8	x	1	
10	Número de registro	Uint16B	x	1	16
4	Descripción del objeto	String		20	
7	Unidades de medida	String		3	
11	Máscara	Uint64B		4	
12	Multiplicador para UI	Uint16B		1	
13	Offset para UI	Uint16B		1	
14	Periodo de conversión	Uint16B		1	
15	Multiplicado por	Uint16B		1	
16	Dividido por	Uint16B		1	32

El total de registros necesarios para el almacenamiento de un objeto es de 16+8*3=40 para las propiedades obligatorias y 32+8*3=56 para las opcionales. Por tanto el máximo es de 96 registros.

7.1.6 SALIDA ANALÓGICA

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	x	1	
2	Nombre del objeto	String	x	8	
3	Número de instancia de la clase	Uint16B	x	1	
5	Valor actual	Float	x	2	
6	Calidad del dato	Uint8	x	1	
9	Tipo de dato	Uint16B	x	1	
10	Número de registro HR inicial asociado	Uint16B	x	1	
11	Multiplicador para UI	Uint16B	x	1	
12	Offset para UI	Uint16B	x	1	17
4	Descripción del objeto	String		20	
7	Unidades de medida	String		3	
8	Tiempo mínimo entre cambios	Uint16B		1	
13	Multiplicado por	Uint16B		1	
14	Dividido por	Uint16B		1	26

El total de registros necesarios para el almacenamiento de un objeto es de 17+9*3=44 para las propiedades obligatorias y 26+5*3=41 para las opcionales. Por tanto el máximo es de 85 registros.

7.1.7 DÍAS FESTIVOS

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	x	1	
2	Nombre del objeto	String	x	8	
3	Número de instancia de la clase	Uint16B	x	1	
5	Número de fechas festivas N	UInt8	x	1	
6	Lista Fechas	Fecha[N]	x	2*N	
7	Número de intervalos festivos M	UInt8	x	1	
8	Lista Intervalos	Intervalo Fechas[M]	x	3*M	12+2*N+3M
4	Descripción del objeto	String		20	20

El total de registros necesarios para el almacenamiento de un objeto es de 12+7*3+2*N+3M=33+2*N+3M para las propiedades obligatorias y 20 para las opcionales. Por tanto el máximo es de 53+2*N+3*M registros.

7.1.8 CALENDARIOS

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	X	1	
2	Nombre del objeto	String	X	8	
3	Número de instancia de la clase	Uint16B	X	1	
8	Número de elementos de la lista N	UInt8	X	1	11
4	Descripción del objeto	String		20	
5	Fecha Inicial	Fecha		2	
6	Fecha Final	Fecha		2	
7	Semana Laboral	UInt8		1	
9	ID lista días festivos	Uint16B[N]		N	25+N

El total de registros necesarios para el almacenamiento de un objeto es de 11+4*3=23 para las propiedades obligatorias y 25+N+5*3=40+N para las opcionales. Por tanto el máximo es de 63+N registros.

7.1.9 LISTA HORAS ACCIONES

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	X	1	
2	Nombre del objeto	String	X	8	
3	Número de instancia de la clase	Uint16B	X	1	
5	Número de elementos N de la lista	UInt8	X	1	
6	Día/Hora/Min	UInt16B[N]	X	N	11+N
4	Descripción del objeto	String		20	20

El total de registros necesarios para el almacenamiento de un objeto es de 11+N+5*3=26+N para las propiedades obligatorias y 20 para las opcionales. Por tanto el máximo es de 46+N registros.

7.1.10 LISTA OBJETOS

ID_PROP	NOMBRE	TIPO DATO	OBLIG	REGISTROS	
1	Identificador del objeto	Uint16B	X	1	
2	Nombre del objeto	String	X	8	
3	Número de instancia de la clase	Uint16B	X	1	
5	Número de elementos de la lista N	UInt8	X	1	
6	ID_OBJETOS	UInt16B[N]	X	N	11+N
4	Descripción del objeto	String		20	20

El total de registros necesarios para el almacenamiento de un objeto es de 11+N+5*3=26+N para las propiedades obligatorias y 20 para las opcionales. Por tanto el máximo es de 46+N registros.

7.1.11 HORARIOS

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros
1	Identificador del objeto	Uint16B	x	1
2	Nombre del objeto	String	x	8
3	Número de instancia de la clase	Uint16B	x	1
5	Activo	UInt8	x	1
6	Calendario asociado	UInt16B	x	1
7	Número de elementos de la lista de excepciones M	UInt8	x	1

9	Número de acciones N	UInt8	x	1	
10	Lista de acciones de control	ACCIÓN[N]	x	14*N	14+14*N
4	Descripción del objeto	String		20	
8	Lista días excepciones	Fecha[M]		2*M	20+2*M

El total de registros necesarios para el almacenamiento de un objeto es de 14+14*N+8*3=38+14*N para las propiedades obligatorias y 20+2*M+2*3=26+2*M para las opcionales. Por tanto el máximo es de 64+14*N +2*M registros.

7.1.12 RTC

ID_PROP	NOMBRE	TIPO DATO	OBLIG	Registros	
1	Identificador del objeto	Uint16B	x	1	
2	Nombre del objeto	String	x	8	
3	Número de instancia de la clase	Uint16B	x	1	
5	Valor	Fecha-Hora	x	3	
6	ValorUTC	Fecha-Hora	x	3	16
4	Descripción del objeto	String		20	
7	Longitud	Float		2	
8	Latitud	Float		2	
9	Offset UTC	Int16B		1	
10	Hora amanecer	Hora		1	
9	Hora anochecer	Hora		1	
10	Cambio de hora automático inv/verano	Bool		1	28

El total de registros necesarios para el almacenamiento de un objeto es de 16+5*3=31 para las propiedades obligatorias y 28+7*3=49 para las opcionales. Por tanto el máximo es de 80 registros.

7.1.13 TABLA RESUMEN DE REGISTROS DE 16BITS USADOS

Se resume en la tabla siguiente el número de registros totales necesarios para la implementación de cada instancia de las clases.

CLASE	OBLIGATORIO	OPCIONAL	TOTAL
Dispositivo	110	98	208
Entrada Binaria	39	82	121

Salida Binaria	39	121	160
Contador	39	51	90
Entrada Analógica	40	56	96
Salida Analógica	44	41	85
Festivos	33+2*N+3*M	20	53+2*N+3*M
Calendarios	23	40+N	63+N
Lista Horas Acciones	26+N	20	46+N
Lista Objetos	26+N	20	46+N
Horarios	38+14*N	26+2*M	64+14*N+2*M
RTC	31	49	80

7.2 CÁLCULO DE LAS NECESIDADES DE MEMORIA DE UN ESCLAVO

Vamos a poner un ejemplo de esclavo Modbus que utilice diversos recursos para estudiar las necesidades completas de registros.

Además de los registros necesarios para la configuración cada entrada y salida utilizará además sus registros Modbus convencionales para permitir el acceso a los campos valor de los objetos. Estos registros estarán mapeados en los cuatro mapas de registros existentes.

Definición del esclavo

Suponemos un esclavo multifunción que contenga todos los tipos de recursos.

- 16 Entradas Binarias
- 16 Salidas Binarias
- 8 Entradas de Contador
- 16 Entradas Analógicas
- 16 Salidas Analógicas

En base a los datos de la tabla anterior, el número MÁXIMO de registros necesarios para la configuración de este esclavo sin contabilizar los horarios resulta:

CLASE	INSTANCIAS	REGISTROS	TOTAL
Dispositivo	1	208	208
Entradas Binarias	16	121	1936
Salidas Binarias	16	160	2560
Contadores	8	90	720

Entradas Analógicas	16	96	1536
Salidas Analógicas	16	85	1360
		TOTAL	**6010**

El número de registros necesarios para los horarios dependerá del número de elementos que consideremos para cada clase. Para poder hacer una estimación, haremos las siguientes suposiciones:

- La ciudad donde estamos realizando el control es Sevilla
- El año de estudio es 2015
- El controlador tiene 16 salidas binarias y 16 salidas analógicas
- El número de acciones de control por horarios se realizarán los 7 días de la semana
- El número de acciones de control para cada punto de control y cada día de la semana es de 4

La especificación de la ciudad es necesaria para saber los días festivos anuales y los intervalos de fechas festivos. Para analizarlo, usaremos el calendario laboral para el año 2015.

La relación de festivos es la siguiente:

- 1 Enero
- 6 Enero
- 28 Febrero
- 2 Abril
- 3 Abril
- 22 Abril
- 1 Mayo
- 4 Junio
- 15Agosto
- 12 Octubre
- 2 Noviembre
- 7 Diciembre
- 8 Diciembre
- 25 Diciembre

Analizando el conjunto de las fechas determinamos dos intervalos 2-3 abril y 7-8 diciembre. Por tanto resultan 10 festivos y dos intervalos festivos.

Festivos

El número de registros usados para la instancia de la clase festivos es de 53+2*N+3*M donde N=10 y M=2, por tanto regs=53+2*10+3*2=**79**

Listas de Objetos

Las acciones de control de horarios solo pueden realizarse sobre las salidas, tanto binarias como analógicas (32 en total), por eso solo hemos señalado esos recursos del controlador.

No conocemos el uso de los puntos de control y suponemos que las acciones de los horarios se realizarán sobre el conjunto de puntos del mismo tipo, digitales por un lado y analógicos por otro. Supongamos que las agrupaciones de objetos son de cuatro elementos. Tendremos entonces un total de 8 listas de objetos, cada una de ellas con 4 objetos.

Para una lista los registros usados son 46+N, siendo N el número de objetos de la lista que en nuestro caso es de 2. Como tenemos cuatro listas el resultado es regs=8*(46+4)=**400**.

Calendario

Usaremos un solo calendario que estará vinculado a la única instancia de festivos que hemos creado, así que regs=63+N donde N indica el número de instancias de festivos vinculados al calendario que en nuestro ejemplo vale 1. El resultado queda entonces regs=63+1=**64**.

Lista de Acciones

La lista de horas de las acciones necesita de 46+N registros donde N es el número de acciones horarias definidas. En el caso más desfavorable consideraremos que en cada una de las cuatro acciones diarias de control el valor de actuación es distinto, pero igual para cada día de la semana. Tenemos que considerar entonces 4 listas horarias cada una de ellas con 7 acciones, una por cada día de la semana. Tenemos entonces que regs=4*(46+7)=**212.**

Horarios

Por último tenemos que considerar los horarios que usarán 64+14*N+2*M registros, donde M es el número de excepciones y N es el número de acciones de control a desarrollar. Supondremos que las acciones de control se ejecutan de la misma manera los 7 días de la semana. Como hemos definido 16 grupos de objetos, cada uno de ellos usará sus propias acciones de control en cada una de las cuatro listas horarias. Tenemos por tanto 16*4=64 acciones de control.

Si no consideramos fechas de excepción nos queda finalmente que regs=64+14*64=**960**.

Resumimos a continuación los registros usados por la gestión de horarios.

Festivos	79
Listas de Objetos	400
Calendario	64
Lista de Acciones	312
Horarios	960
TOTAL	**1815**

En resumen el número total de registros necesarios para este ejemplo de controlador es de 6010+1815=7825.

Recordamos que cada uno de los mapas de registros Modbus permite 65535 registros, así que hemos usado solo el 11,94% del total de registros de un solo mapa.

En el anexo 4 se realiza la programación completa de este supuesto de calendario.

7.3 ACCESO A LOS OBJETOS DESDE LAS FUNCIONES MODBUS CLÁSICAS

El nuevo conjunto de funciones Modbus creadas, permite un acceso fácil a las informaciones de los objetos definidos en los dispositivos con el protocolo extendido.

Aunque cada objeto tiene en su espacio de memoria la información del campo valor de cada propiedad, es conveniente que, las informaciones de estos valores estén también situadas ocupando direcciones consecutivas en sus correspondientes mapas de memoria que son accesibles a través de las funciones clásicas de Modbus. Esto permite mantener las características del protocolo de acceso simultáneo a muchos registros en un único mensaje.

Por compatibilidad con los sistemas existentes, es conveniente no obstante, permitir el acceso a esta información utilizando las funciones del protocolo estándar. En concreto, usamos registros de 16 bits para lectura y escritura que determinan que el acceso a esta información se puede realizar a través de las funciones clásicas 3 para lectura y 16 para escritura usando el mapa de memoria de los holding registers.

La información sobre el contenido y posición de los registros debe hacerse a través de un documento informativo que muestre de forma individual los contenidos de cada uno de los registros utilizados.

Es posible autodocumentar la estructura de objetos del dispositivo con la creación de algunas reglas básicas y la modificación de la clase dispositivo:

- La instancia de la clase dispositivo se fija siempre a partir de la misma dirección en el mapa de los holding register. De esta forma siempre sabremos cómo acceder a la información del objeto que contiene la información del *dispositivo*.
- Se debe modificar la clase *dispositivo* para que contenga la información de cuantos objetos de cada clase contiene el dispositivo y la dirección del holding register del primer objeto de cada clase.
- El espacio de memoria destinado a cada objeto debe ser el máximo posible, es decir debe contener el espacio para los datos obligatorios y también de los opcionales. De esta forma podemos saltar de una a otra instancia de una clase usando un incremento de registros, que se corresponde con el tamaño del objeto en número de registros, siendo este un valor constante para todas las instancias de la misma clase.

8 CONCLUSIONES

8 CONCLUSIONES

Una vez finalizado el trabajo, podemos aportar las siguientes conclusiones:

- Se ha propuesto un conjunto de tipos de datos básico que permite la interoperabilidad de los dispositivos esclavos y sus maestros.
- Se ha propuesto un sistema de identificación de los dispositivos Modbus a través de un código de 48 bits definido como MODBUS_ID que permite su identificación única.
- Se ha propuesto la utilización de un botón y de un led de programación.
- Se han propuesto y desarrollado nuevas funciones Modbus con los códigos 100, 101 y 102 que permiten:
    - Asignar la dirección de un esclavo a través del bus de comunicaciones
    - El borrado de la dirección individual de un esclavo a través del bus
    - El encendido del led de programación desde el bus para la identificación de un esclavo
    - El Asignamiento automático de direcciones a los esclavos que no tengan asignada dirección
    - La resolución de conflictos de dirección debidos a direcciones repetidas
- Se ha definido un conjunto de clases de información sobre los recursos de los dispositivos, suficiente para la descripción del propio dispositivo, sus entradas y salidas físicas y su funcionalidad.
- Se ha definido un conjunto de clases, completas y relacionales, para permitir la gestión de horarios en los dispositivos esclavos.
- Se han definido nuevas funciones Modbus con los códigos 103 y 104, capaces de gestionar la nueva estructura de objetos creada y de permitir un acceso alternativo a los datos a través de la propiedad VALOR utilizando en este caso un tipo de dato normalizado.
- Se ha comprobado la validez del sistema de objetos propuesto, en cuanto a la necesidad de utilización de recursos de memoria asignados en los mapas de los holding registers.

ANEXO 1. MODELO PIC DEL PROTOCOLO BACNET

Honeywell

Class 500 BACNet PIC Statement

SPECIFICATION DATA

BACNET PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT

Date: May 2009
Vendor Name: FieldServer Technologies
Product Name: FieldServer
Product Model Number: FS-B40, FS-B20 or FS-B30 Series. SlotServer
Product Description: This software product will provide bi-directional communication between various RTU, DCS, SCADA and PLC using most common protocols and a BACnet system. The FieldServer can perform protocol conversion (as opposed to routing) between the different BACnet Data Link Layer options. This is arranged by way of static mappings.
Protocol Conversions: See FieldServer Technologies list of protocol drivers to determine available protocol conversions

1.1.1 BACNET STANDARIZED DEVICE PROFILE (ANNEX L) – [NOTE: FIELDSERVER IS A GATEWAY DEVICE]
X BACnet Application Specific Controller (B-ASC)

1.1.2 BACNET INTEROPERABILITY BUILDING BLOCKS SUPPORTED (ANNEX K):
X K.1.2 BIBB - Data Sharing - ReadProperty-B (DS-RP-B)
X K.5.2 BIBB - Device Management - Dynamic Device Binding-B (DM-DDB-B)
X K.5.3 BIBB - Device Management - Dynamic Device Binding-A (DM-DOB-A)

1.1.3 SEGMENTATION CAPABILITY: NONE

1.1.4 STANDARD OBJECT TYPES SUPPORTED
X Device Object
X Analog Input

1.1.5 UNSUPPORTED PROPERTIES AND RESTRICTIONS
1. Does not support BACnet CreateObject
2. Does not support BACnet DeleteObject
3. Does not support any optional properties
4. No additional writeable properties exist
5. No proprietary properties exist
6. No range restrictions exist
7. Client Driver can only read Present Value property

1.1.6 DATA LINK LAYER OPTIONS:
X BACnet IP, (Annex J)
X MS/TP master (Clause 9), baud rate up to 76.8 Kbps

1.1.7 DEVICE ADDRESS BINDING: NOT SUPPORTED

1.1.8 NETWORKING OPTIONS:
X BACnet/IP Broadcast Management Device (BBMD)
X Registrations by Foreign Devices

1.1.9 CHARACTER SETS SUPPORTED:
X ANSI X3.4.

1.1.10 UNSUPPORTED FUNCTIONS OF THE BACNET DRIVER -BACNET/IP AS A CLIENT:
- Registering the FieldServer as a foreign device on a remote BBMD
- Configuring a Connection for BBMD operation

Automation and Control Solutions
Honeywell International Inc.
1985 Douglas Drive North
Golden Valley, MN 55422

Honeywell Limited-Honeywell Limitée
35 Dynamic Drive
Toronto, Ontario M1V 4Z9
customer.honeywell.com

63-1360-01

ANEXO 2. DESCRIPCIÓN DE LAS FUNCIONES MODBUS

Describimos a continuación cada una de las funciones del protocolo, revisando primero la petición desde el maestro y a continuación la respuesta del esclavo.

Si el esclavo encuentra alguna incongruencia en la petición del maestro, devolverá una respuesta de excepción al maestro indicándole el problema detectado.

A2.1 Función 2: Read Input Discrete

Esta función se utiliza por el maestro para solicitar del esclavo los valores de sus entradas digitales.

La petición desde el maestro contiene el código de la función (0x02) y la identificación de las entradas digitales consecutivas solicitadas, especificadas indicando la primera de ellas y el número total de entradas solicitadas.

Function code	1 Byte	**0x02**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Inputs	2 Bytes	1 to 2000 (0x7D0)

Como vemos es posible solicitar los valores de hasta 2000 entradas binarias en una única petición al bus.

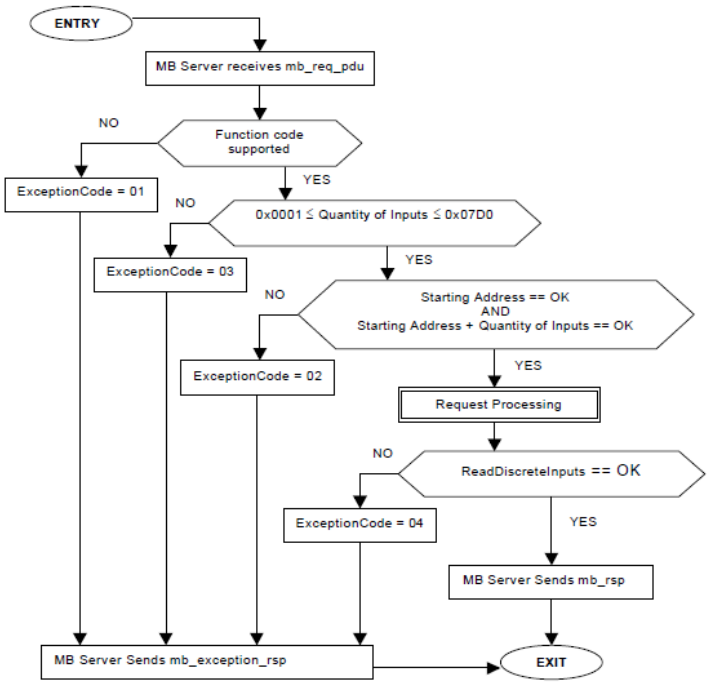
La respuesta desde el esclavo contiene el mismo código de función (0x02) y los datos binarios, usando un bit para representar el valor de cada entrada. Después del código de la función se emplea un byte para indicar el número de bytes de datos que siguen. Cada byte contiene el valor de 8 entradas binarias, menos el último que puede contener menos de 8 bits si el número de entradas solicitadas no es múltiplo de 8.

Function code	1 Byte	**0x02**
Byte count	1 Byte	**N***
Input Status	**N*** x 1 Byte	

Si se produce un error en la petición del maestro, por ejemplo se solicitan entradas que no dispone el esclavo, se responde con el código de error marcando a uno el msb del código de función y a continuación se identifica el error con un código entre 1 y 4.

Error code	1 Byte	**0x82**
Exception code	1 Byte	01 or 02 or 03 or 04

Para cada código de función se definen las situaciones de error y el código de excepción a enviar a través de un diagrama de flujo. Los códigos de respuesta de error 01 a 04 están indicados en el diagrama de flujo de la función.



A2.2 Función 1: Read Coils

Esta función es idéntica a la función 2 pero la información se refiere ahora a los estados de las salidas binarias.

A2.3 Función 5: Write Single Coil

Esta función se utiliza para escribir los valores ON/OFF a una única salida digital. Los valores ON y OFF se representan con 16 bits OFF=0x0000 y ON=0xFF00.

Petición

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

El campo output address indica la dirección del registro Coil sobre la que se realizará la escritura.

El campo Output Value señala el valor ON/OFF que se va a escribir en la salida binaria usando la codificación de 16 bits indicada antes.

Respuesta

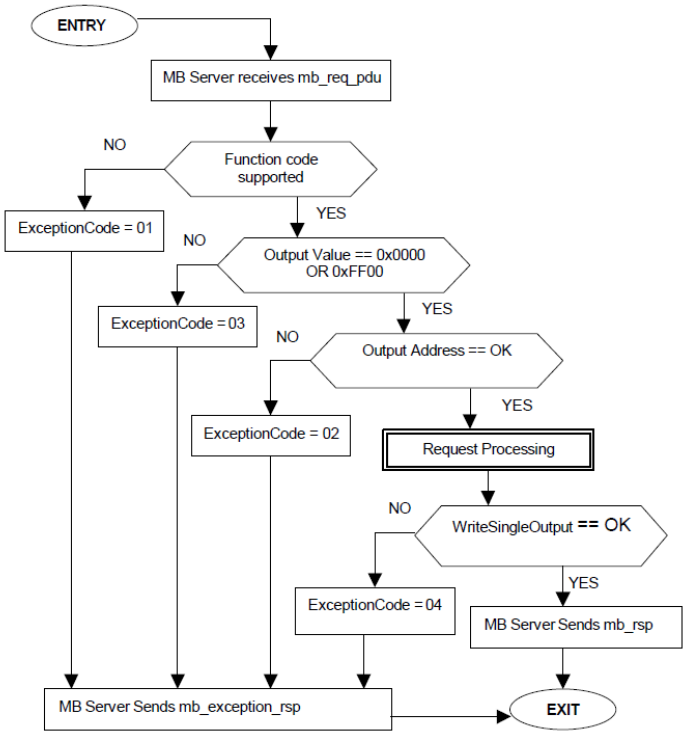
Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

La respuesta enviada por el esclavo es un eco del mensaje recibido del maestro

Respuesta de Error

Error code	1 Byte	**0x85**
Exception code	1 Byte	01 or 02 or 03 or 04

El diagrama de flujo de errores de la función es:



A2.4 Función 15: Write Multiple Coils

Esta función se utiliza para escribir múltiples salidas binarias en una sola petición.

Los valores ON/OFF se representan ahora como un 0 y un 1 ocupando un solo bit para su especificación.

El formato de los mensajes es el siguiente:

Petición

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0
Byte Count	1 Byte	**N***
Outputs Value	**N*** x 1 Byte	

El campo Starting Address indica la dirección del primer Coil que se quiere modificar.

El campo Quantity of Outputs indica cuantas salidas se van a modificar con este mensaje. Cada valor ocupa un bit de un byte, de manera que tendremos varios bytes de datos para especificar todos los bits.

Outputs Value son los bytes de datos para los valores de las salidas. El último byte solo estará completo si el número de salidas es un múltiplo de 8.

Respuesta

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0

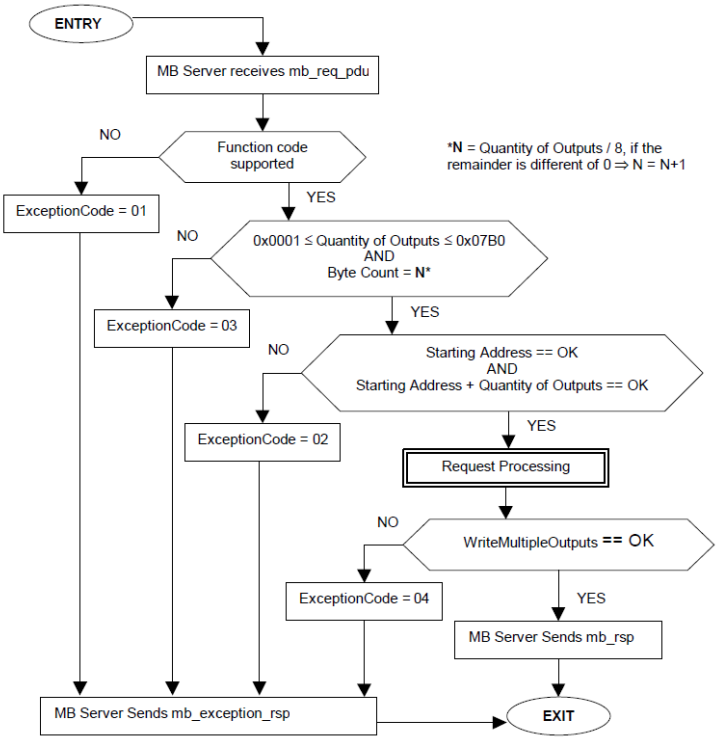
Puesto que la petición del maestro puede ocupar los 256 bytes del mensaje, la respuesta no es un eco de la petición ya que implicaría un tráfico innecesario y además se incrementaría la posibilidad de un error de comunicación. La respuesta del esclavo es un eco de solo los primeros 5 bytes recibidos en la petición del maestro.

Si la petición es errónea se devuelve la respuesta de error.

Error

Error code	1 Byte	**0x8F**
Exception code	1 Byte	01 or 02 or 03 or 04

Los códigos de error se detallan en el diagrama de flujo de errores de la función.



A2.5 Función 4: Read Input Register

Esta función se utiliza para leer valores de entradas de 16 bits. El significado de estos bits lo establece el fabricante del esclavo.

Es posible que los bits representen valores de entradas digitales o que representen valores de entradas analógicas representados a través de datos de 16 bits.

En una sola petición podemos solicitar los valores de hasta 125 registros de 16 bits.

El principal inconveniente de este sistema es que para saber cómo interpretar la información de los registros debemos hacer uso del manual de usuario del dispositivo.

No existen tipos de datos normalizados aunque podemos usar enteros y long con o sin signo y datos float de simple o doble precisión.

Si un dato necesita más de 16 bits se usan varios registros de 16 bits consecutivos. Por ejemplo un dato long de 32 bits o un float de 4 bytes usan dos registros consecutivos de 16 bits mientras que un dato float de doble precisión de 8 bytes precisa 4 registros de 16 bits consecutivos.

También se utilizan formatos especiales como por ejemplo multiplicar el dato por un factor constante para forzar el valor a un entero. Por ejemplo para representar una tensión de eléctrica con dos decimales, multiplicaremos el valor por 100 y lo representaremos como un entero de 16 bits, Así si la tensión medida es de 234,45 al multiplicarla por 100 se transmite el valor como 23445 que como vemos se representa con 16 bits en lugar de los 32 bits requeridos para un datos float de simple precisión. Es obligación del maestro realizar la operación contraria una vez recibido el dato para obtener el valor correcto de la medida.

Petición

Function code	1 Byte	**0x04**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Input Registers	2 Bytes	0x0001 to 0x007D

El campo Starting Address indica la dirección del primer Input Register solicitado.

El campo Quantity of Inputs Registers indica cuantos registros se solicitan en esta petición

Respuesta

Function code	1 Byte	**0x04**
Byte count	1 Byte	2 x **N***
Input Registers	**N*** x 2 Bytes	

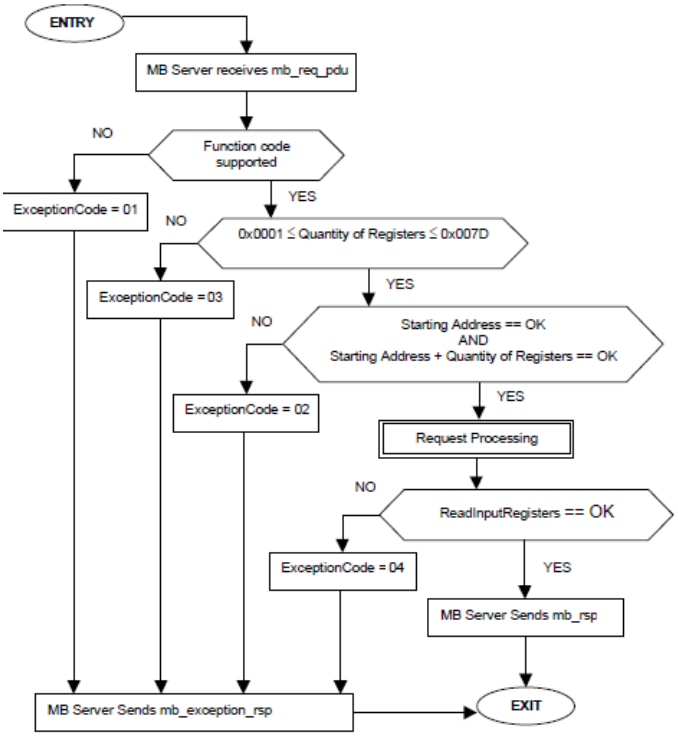
El esclavo devuelve 2 bytes de información por cada registro. Si N es el número de registros solicitados por el maestro, el campo Byte count especifica el tamaño de la respuesta en bytes no en registros y por tanto su valor es 2xN.

Al campo Byte count le siguen los 2xN bytes que contienen los valores de los registros. Los bytes se deben emparejar en grupos de dos para obtener las informaciones de 16 bits de los registros.

Si se detecta un error en la petición del maestro se devuelve la respuesta de error.

Error

Error code	1 Byte	**0x84**
Exception code	1 Byte	01 or 02 or 03 or 04



A2.6 Función 3: Read Holding Register

Es igual a la función 4 pero referida a los datos del mapa de HR.

A2.7 Función 6: Write Single Register

Con esta función podemos escribir los valores de un solo Holding Register de 16 bits. Este valor puede ser una salida física del controlador (salida analógica o PWM) cuyo valor podamos representar con 16 bits. Si el formato de la salida precisa más de 16 bits para representar su valor necesitaremos usar la función 16 que permite modificar los valores de varios HR consecutivos.

Petición

Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 or 0xFFFF

Ya que cada registro es de 16 bits, el campo Register Value que contiene el dato que queremos escribir en el registro ocupa 2 bytes.

Respuesta

Function code	1 Byte	**0x06**
Register Address	2 Bytes	0x0000 to 0xFFFF
Register Value	2 Bytes	0x0000 or 0xFFFF

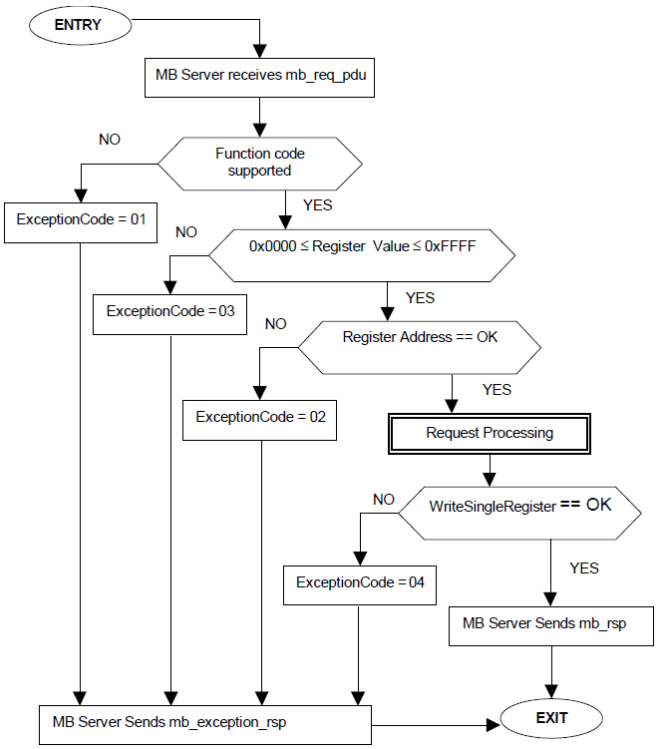
Ya que el tamaño del mensaje de petición es pequeño, la respuesta es un eco de la petición.

Si se detecta un error en la petición se responde con el mensaje de error.

Error

Error code	1 Byte	**0x86**
Exception code	1 Byte	01 or 02 or 03 or 04

El diagrama de flujo de errores de la función es:



A2.8 Función 16: Write Multiple Registers

Si queremos escribir en una petición varios Holding Registers consecutivos, usaremos esta función.

Petición

Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	0x0001 to 0x0078
Byte Count	1 Byte	2 x **N***
Registers Value	**N*** x 2 Bytes	value

El campo Quantity of Registers indica el número de registros que se quieren modificar en esta petición. Si el valor es N, el campo Byte Count expresa el número de bytes necesarios de datos que deben ir en la petición, su valor es 2xN. Registers Value son los 2xN bytes de datos que contienen la información a escribir en los registros.

Respuesta

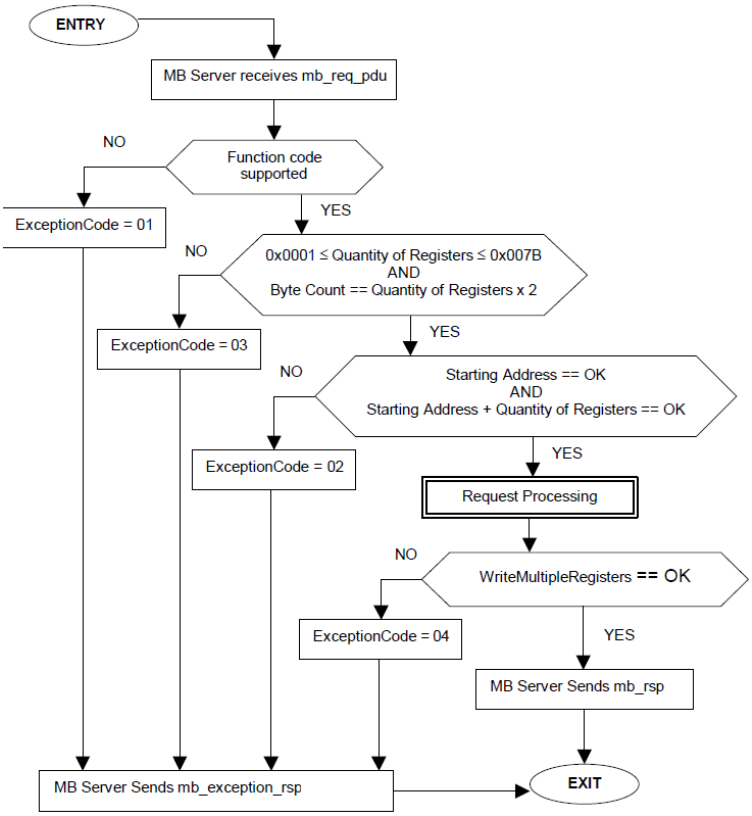
Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 123 (0x7B)

La respuesta en el eco de los 5 primeros bytes del mensaje. Si se detecta un error en la petición se devuelve el mensaje de error asociado.

Error

Error code	1 Byte	0x90
Exception code	1 Byte	01 or 02 or 03 or 04

El diagrama de flujo de errores de la función es:



A2.9 Función 20 (0x14): Read File Record

Esta función se utiliza para la lectura de registros de datos de ficheros almacenados en el esclavo. Los datos pueden ser de configuración, de históricos de datos, de alarmas etc.

Un fichero es un conjunto de hasta 10000 registros numerados del 0 al 9999.

La función 20 permite leer simultáneamente registros de distintos ficheros. Cada bloque de datos de lectura se denomina un grupo. Los registros especificados de un grupo tienen que ser consecutivos.

Cada grupo se especifica por los siguientes datos:

- El tipo de referencia que obligatoriamente toma el valor 6
- El número de fichero que se especifica por 16 bits
- El registro inicial dentro del fichero que se especifica por 16 bits
- La longitud del registro que se especifica por 16 bits

Como siempre en Modbus, la longitud máxima del mensaje debe ser de 256 bytes.

La respuesta del esclavo consiste en una serie de sub-respuestas, una por cada sub-petición realizada por el maestro. Cada sub-respuesta contiene un byte de longitud que indica el número de bytes que contiene la sub-respuesta.

Petición

Function code	1 Byte	**0x14**
Byte Count	1 Byte	0x07 to 0xF5 bytes
Sub-Req. x, Reference Type	1 Byte	06
Sub-Req. x, File Number	2 Bytes	0x0000 to 0xFFFF
Sub-Req. x, Record Number	2 Bytes	0x0000 to 0x270F
Sub-Req. x, Register Length	2 Bytes	**N**
Sub-Req. x+1, ...		

Byte count indica el número de bytes que siguen al código de función.

Cada sub-petición contiene la información de los cuatro campos indicados con anterioridad. Las sub-peticiones se encadenan en el mensaje de la petición para los distintos grupos requeridos.

Respuesta

Function code	1 Byte	**0x14**
Resp. data Length	1 Byte	0x07 to 0xF5
Sub-Req. x, File Resp. length	1 Byte	0x07 to 0xF5
Sub-Req. x, Reference Type	1 Byte	6
Sub-Req. x, Record Data	**N** x 2 Bytes	
Sub-Req. x+1, ...		

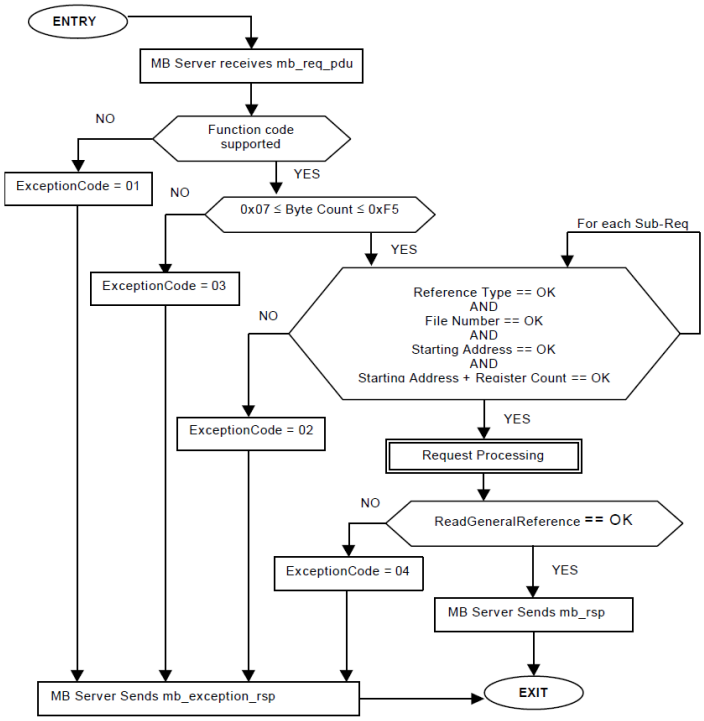
Podemos ver en la respuesta que hay un valor Resp. Data Length global que indica los bytes totales que siguen al código de función y por cada grupo tenemos el byte Sub-req. X, File Resp. Length que indica el número de bytes que siguen que pertenecen a ese grupo.

Si se detecta un error en la petición del maestro, el esclavo responde con el código de error correspondiente.

Error

Error code	1 Byte	**0x94**
Exception code	1 Byte	01 or 02 or 03 or 04 or 08

El diagrama de estados de error es el siguiente.



A2.10 Función 21 (0x15) Write File Record

Esta función se utiliza para escribir registros en ficheros. Los ficheros están definidos en la función 20.

Petición

Function code	1 Byte	**0x15**
Request data length	1 Byte	0x07 to 0xF5
Sub-Req. x, Reference Type	1 Byte	06
Sub-Req. x, File Number	2 Bytes	0x0000 to 0xFFFF
Sub-Req. x, Record Number	2 Bytes	0x0000 to 0x270F
Sub-Req. x, Record length	2 Bytes	**N**
Sub-Req. x, Record data	**N** x 2 Bytes	
Sub-Req. x+1, ...		

El campo Request data length indica el número total de bytes que siguen.

Cada grupo contiene:

- La información de referencia
- Número de fichero
- Número del primer registro
- Número de registros a escribir

- Bloque de los datos como conjunto de registros de 16 bits.

El número máximo de grupos que podemos usar dependerá del tamaño de los grupos anteriores ya que el mensaje no puede sobrepasar los 256 bytes que es el tamaño máximo de un mensaje Modbus.

Respuesta

Function code	1 Byte	**0x15**
Response Data length	1 Byte	
Sub-Req. x, Reference Type	1 Byte	06
Sub-Req. x, File Number	2 Bytes	0x0000 to 0xFFFF
Sub-Req. x, Record number	2 Bytes	0x0000 to 0xFFFF
Sub-Req. x, Record length	2 Bytes	0x0000 to 0xFFFF **N**
Sub-Req. x, Record Data	**N** x 2 Bytes	
Sub-Req. x+1, ...		

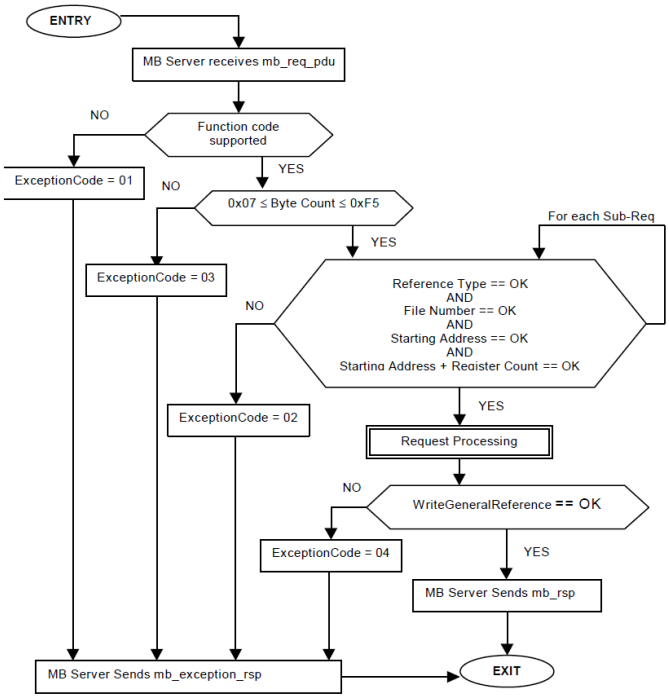
La respuesta es un eco de la petición del maestro, lo cual difiere del criterio seguido en otras funciones en las que el número de datos del mensaje es elevado.

Si se detectan errores en la petición del maestro se devuelve el mensaje de error.

Error

Error code	1 Byte	**0x95**
Exception code	1 Byte	01 or 02 or 03 or 04 or 08

El diagrama de flujo de errores es:



A2.11 Función 22 (0x16): Mask Write Register

Esta función se utiliza para modificar un holding register usando una combinación de máscaras AND y OR y el contenido actual del registro. La función se utiliza para borrar o poner a 1 bits individuales del registro.

La función realizada sobre el registro es la siguiente:

Resultado = (Contenido actual **AND** máscara_AND) **OR** (máscara_OR **AND** $\overline{\text{máscara\_AND}}$)

Si la máscara OR es cero la operación es solo una máscara AND con el valor actual del registro. Si la máscara AND es cero el resultado de la operación es solo la máscara OR.

Petición

Function code	1 Byte	**0x16**
Reference Address	2 Bytes	0x0000 to 0xFFFF
And_Mask	2 Bytes	0x0000 to 0xFFFF
Or_Mask	2 Bytes	0x0000 to 0xFFFF

El campo Reference Address indica la dirección del holding register que se quiere modificar.

Respuesta

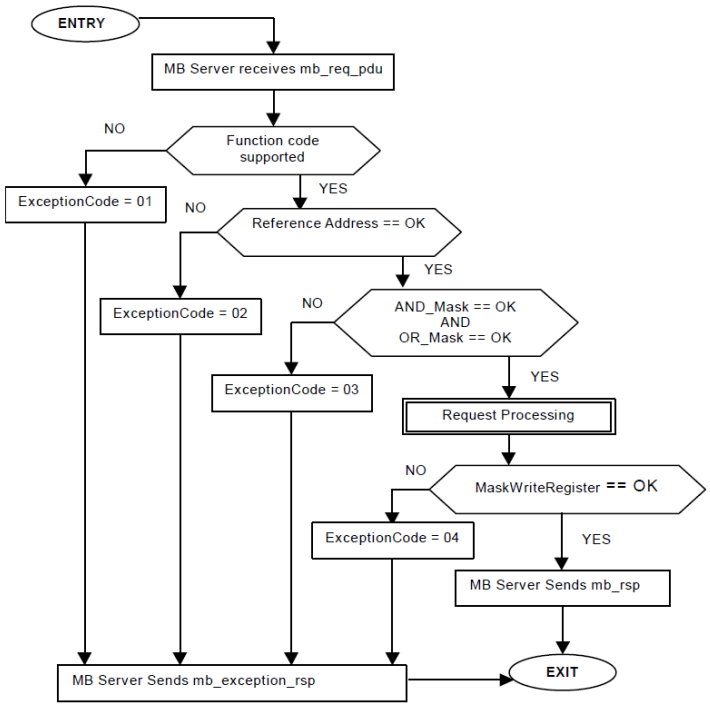
Function code	1 Byte	**0x16**
Reference Address	2 Bytes	0x0000 to 0xFFFF
And_Mask	2 Bytes	0x0000 to 0xFFFF
Or_Mask	2 Bytes	0x0000 to 0xFFFF

La respuesta es el eco de la petición si no se han producido errores. Si hay errores se devuelve el código de error correspondiente.

Error

Error code	1 Byte	**0x96**
Exception code	1 Byte	01 or 02 or 03 or 04

El diagrama de flujo de errores es:



A2.12 Función 23 (0x17): Read/Write Multiple Registers

Esta función realiza una combinación de una operación de lectura y otra de escritura en un solo mensaje.

Petición

Function code	1 Byte	**0x17**
Read Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity to Read	2 Bytes	0x0001 to approx.0x0076
Write Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity to Write	2 Bytes	0x0001 to approx. 0X0076
Write Byte Count	1 Byte	2 x **N***
Write Registers Value	**N*** x 2 Bytes	

El campo Read Starting Address indica la dirección del primer holding register a leer. Quantity to Read indica el número de registros consecutivos que se desean leer.

Write Starting Address indica la dirección del primer registro que se desea escribir. Quantity to Write indica el número de registros que se escribirán, y Write Byte Count es el número de bytes a escribir que es el doble del número de registros. Write Registers Value contiene los 2xN bytes a escribir en los registros.

Respuesta

Function code	1 Byte	**0x17**
Byte Count	1 Byte	2 x **N'***
Read Registers value	**N'*** x 2 Bytes	

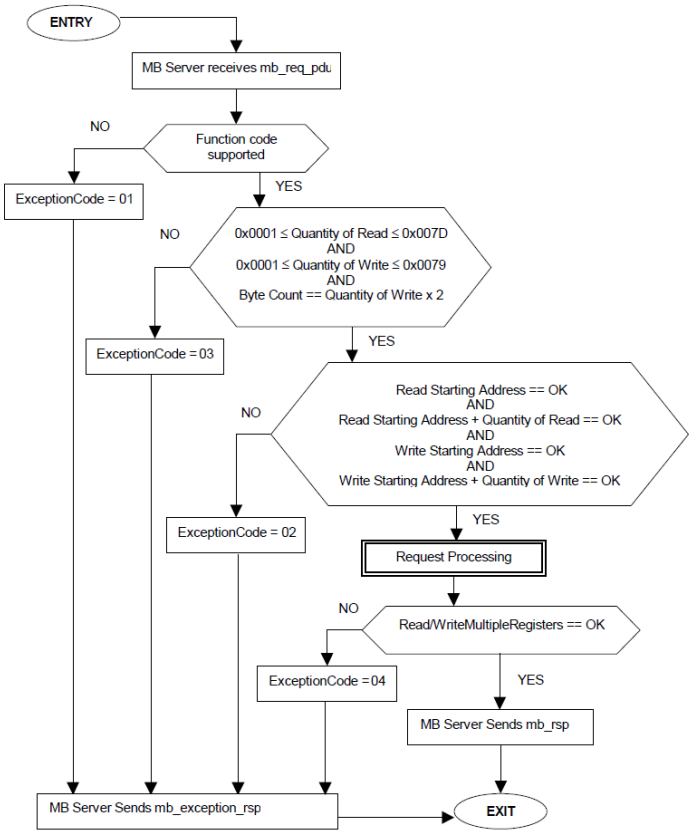
La respuesta contiene la información de los registros a leer. Byte Count indica el número de bytes de datos que siguen. A continuación se envían los 2xN bytes correspondientes a los N registros que se leen.

Si se detecta un error se responde con el mensaje de error correspondiente.

Error

Error code	1 Byte	**0x97**
Exception code	1 Byte	01 or 02 or 03 or 04

El diagrama de flujo de errores es:



A2.13 Función 43 (0x2B): Read Device Identification

Esta función permite leer la identificación e información adicional relativa a la descripción física y funcional de un esclavo.

La función está modelada como un espacio de direcciones compuesto de un conjunto de datos direccionables. Los datos se llaman objetos y contienen un identificador numérico id.

Los datos se estructuran en tres categorías: Basic (ids 0x00 a 0x02), Regular (ids 0x03 a 0x7F) y Extended (ids 0x80 a 0xff).

Los datos de la categoría Basic son obligatorios y los da las categoría Regular y extended son opcionales. Todos los datos se representan como cadenas de caracteres.

Object Id	Object Name / Description	Type	M/O	category
0x00	VendorName	ASCII String	**Mandatory**	**Basic**
0x01	ProductCode	ASCII String	**Mandatory**	
0x02	MajorMinorRevision	ASCII String	**Mandatory**	
0x03	VendorUrl	ASCII String	Optional	**Regular**
0x04	ProductName	ASCII String	Optional	
0x05	ModelName	ASCII String	Optional	
0x06	UserApplicationName	ASCII String	Optional	
0x07 … 0x7F	*Reserved*		Optional	
0x80 … 0xFF	*Private objects may be **optionally** defined* *The range [0x80 – 0xFF] is Product dependant.*	device dependant	Optional	**Extended**

La petición desde el maestro tiene lo siguientes campos:

Request PDU

Function code	1 Byte	**0x2B**
MEI Type	1 Byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Object Id	1 Byte	0x00 to 0xFF

Y la respuesta sin errores:

Response PDU

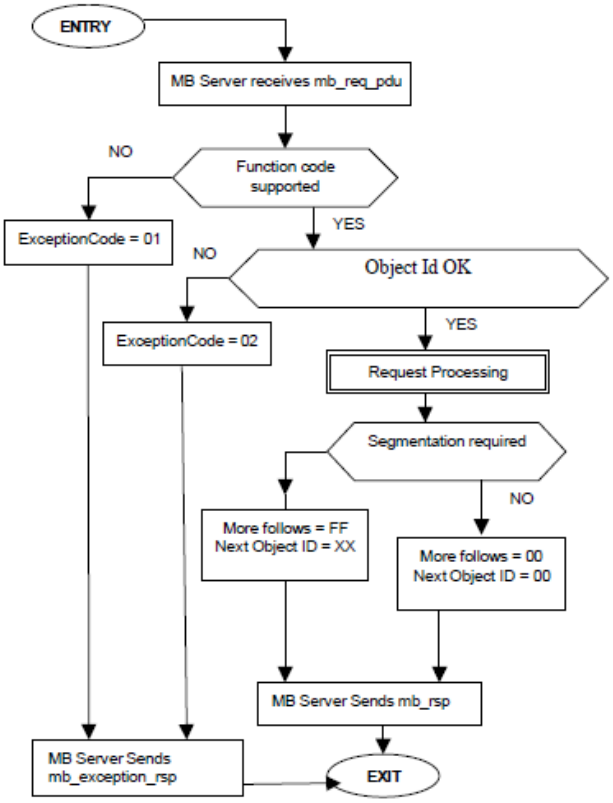
Function code	1 Byte	**0x2B**
MEI Type	1 byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Conformity level	1 Byte	
More Folows	1 Byte	00 / FF
Next Object Id	1 Byte	Object ID number
Number of objects	1 Byte	
List Of		
Object ID	1 Byte	
Object length	1 Byte	
Object Value	1 Byte	

En caso de error la respuesta pasa a ser:

Error

Function code	1 Byte	**0xAB :** **Fc 0x2B + 0x80**
MEI Type	1 Byte	14
Exception code	1 Byte	01, 02, 03, 04

Los códigos de excepción se detallan en el diagrama de flujo de la función.



Significado de los campos usados en la petición

El código MEI (Modbus Encapsulated Interface) con valor 14 (0x0E) identifica la operación Read Identification Request. No están definidos en el estándar otros códigos de operación.

El campo Read Device ID code toma los valores de 1 a 4 cuyo significado se describe a continuación.

El campo Object ID indica el registro a partir del cual se solicitan las informaciones.

Los valores 1,2 y 3 del campo ReadDeviceID, leen una secuencia consecutiva de registros (stream Access) comenzando con el indicado en el campo Object Id. El valor 1 se usa para leer

registros Basic, el valor 2 para leer registros Regular y el valor 3 para registros Extended. El valor 4 se usa para la lectura de un único registro (individual Access) de cualquiera de las tres zonas.

En el caso de que la respuesta no quepa en un mensaje, se realizarán varias peticiones y respuestas. En la primera petición, el maestro escribirá el ID del primer campo solicitado en el campo Object ID. En las siguientes peticiones se escribirá el valor devuelto por el esclavo en el campo NextObjectID para continuar con el resto de las peticiones.

Si el valor del campo Object ID no se corresponde con ningún registro válido se interpretará como cero salvo para el valor 4 del campo Read Device ID code, en cuyo caso se responderá con la respuesta de error con un código de excepción de 2.

Significado de los campos usados en la respuesta

Los campos Function Code, MEI Type y Read Device ID code son iguales a los del mensaje de petición.

conformity level

Este campo representa la identificación del nivel de conformidad del dispositivo y el tipo de acceso permitido. Se usan los siguientes valores:

0x01: lectura de registros basic stream access

0x02: lectura de registros regular stream access

0x03: lectura de registros extended stream Access

0x81: lectura de registros basic stream Access and individual access

0x82: lectura de registros regular stream access and individual access

0x83: lectura de registros extended stream Access and individual access

More Follows

Tiene el valof 0xFF si no caben todas los datos en una respuesta. En caso contrario devuelve cero.

Next object ID

Contiene el ID del campo a usar en la siguiente petición si el campo More Follows tenía el valor 0xFF. En caso contrario vale cero.

Number Of Objects

Contiene el número de objetos de la respuesta completa.

List Objects

Contiene la lista de objetos devueltos en la respuesta. Cada objeto contiene tres campos: Object ID para identificar el objeto, Object lenght que indica el tamaño del objeto indicado por el número de caracteres y Object Value que contiene la cadena de caracteres del valor del objeto.

Veamos unos ejemplos de petición respuesta para aclarar su funcionamiento.

Ejemplo 1: respuesta en un mensaje

Request		Response	
Field Name	*Value*	*Field Name*	*Value*
Function	2B	Function	2B
MEI Type	0E	MEI Type	0E
Read Dev Id code	01	Read Dev Id Code	01
Object Id	00	Conformity Level	01
		More Follows	00
		NextObjectId	00
		Number Of Objects	03
		Object Id	00
		Object Length	16
		Object Value	" Company identification"
		Object Id	01
		Object Length	0A
		Object Value	" Product code "
		Object Id	02
		Object Length	05
		Object Value	"V2.11"

Ejemplo 2: respuesta en dos peticiones

Petición/respuesta 1

Request		Response	
Field Name	*Value*	*Field Name*	*Value*
Function	2B	Function	2B
MEI Type	0E	MEI Type	0E
Read Dev Id code	01	Read Dev Id Code	01
Object Id	00	Conformity Level	01
		More Follows	FF
		NextObjectId	02
		Number Of Objects	03
		Object Id	00
		Object Length	16
		Object Value	" Company identification"
		Object Id	01
		Object Length	1A
		Object Value	" Product code XXXXXXXXXXXXXXX"

Petición/respuesta 2

Request		Response	
Field Name	*Value*	*Field Name*	*Value*
Function	2B	Function	2B
MEI Type	0E	MEI Type	0E
Read Dev Id code	01	Read Dev Id Code	01
Object Id	02	Conformity Level	01
		More Follows	00
		NextObjectId	00
		Number Of Objects	03
		Object Id	02
		Object Length	05
		Object Value	"V2.11"

Las informaciones suministradas en esta función son como vemos generales. No se describen las características físicas y funcionales del dispositivo como los registros usados en cada uno de los cuatro mapas y el significado de cada uno de ellos.

ANEXO 3: CÓDIGOS DE EXCEPCIÓN MODBUS

MODBUS Exception Codes		
Code	**Name**	**Meaning**
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 will generate exception 02.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
05	ACKNOWLEDGE	Specialized use in conjunction with programming commands. The server (or slave) has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client (or master). The client (or master) can next issue a Poll Program Complete message to determine if processing is completed.
06	SLAVE DEVICE BUSY	Specialized use in conjunction with programming commands. The server (or slave) is engaged in processing a long–duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
08	MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server (or slave) attempted to read record file, but detected a parity error in the memory. The client (or master) can retry the request, but service may be required on the server (or slave) device.
0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

ANEXO 4: EJEMPLO DE IMPLEMENTACIÓN DE LAS INSTANCIAS DE LOS OBJETOS DE CONTROL HORARIO

Datos del caso de ejemplo analizado

- La ciudad donde estamos realizando el control es Sevilla
- El año de estudio es 2015
- El controlador tiene 16 salidas binarias y 16 salidas analógicas
- El número de acciones de control por horarios se realizarán los 7 días de la semana
- El número de acciones de control para cada punto de control y cada día de la semana es de 4

Datos del calendario laboral de Sevilla para el año 2015. La relación de festivos es la siguiente:

- 1 Enero
- 6 Enero
- 28 Febrero
- 2 Abril
- 4 Junio
- 15 Agosto
- 12 Octubre
- 2 Noviembre

- 3 Abril
- 22 Abril
- 1 Mayo
- 7 Diciembre
- 8 Diciembre
- 25 Diciembre

Las salidas binarias y analógicas se agrupan en grupos o listas de 4 objetos. Usaremos la siguientes abreviaturas LOx lista de objetos x, BOx salida binaria x, AOx salida analógica x.

LO1: BO1, BO2, BO3, BO4

LO2: BO5, BO6, BO7, BO8

LO3: BO9, BO10, BO11, BO12

LO4: BO13, BO14, BO15, BO16

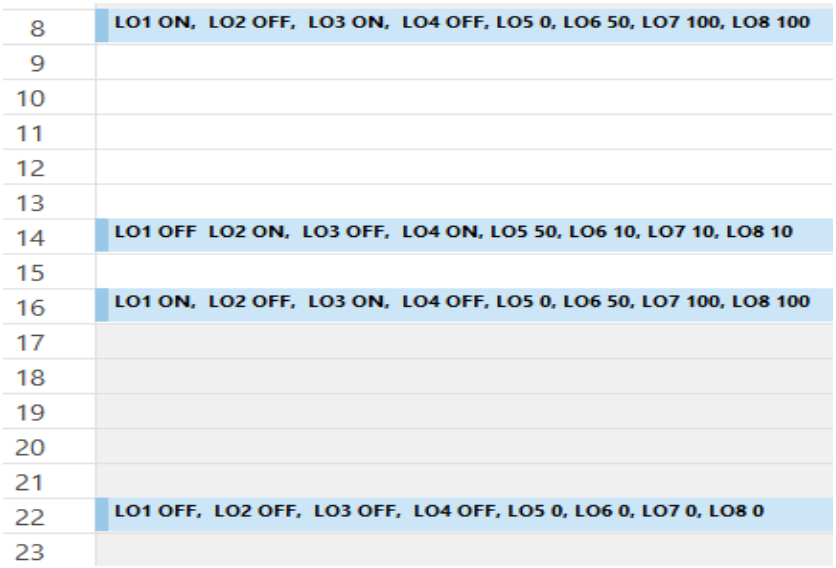
LO5: AO1, AO2, AO3, AO4

LO6: AO5, AO6, AO7, AO8

LO7: AO9, AO10, AO11, AO12

LO8: AO13, AO14, AO15, AO16

La relación de acciones de control que se quiere realizar cada día de la semana es:



Para los valores de los identificadores de los objetos comenzaremos su numeración por el valor 101.

A4.1 DIAS FESTIVOS

Necesitamos una sola instancia de festivos. Damos ahora valores a las propiedades de la instancia número 1.

NOMBRE	VALOR
Identificador del objeto	101
Nombre del objeto	Festivos
Número de instancia de la clase	1
Descripción del objeto	Festivos Sevilla 2015

Número de fechas festivas	10
Lista Fechas	1/1, 6/1, 28/2, 22/4, 1/5, 4/6, 15/8, 12/10, 2/11, 25/12
Número de intervalos festivos	2
Lista Intervalos	2-3/4, 7-8/12

A4.2 CALENDARIOS

Necesitamos un solo calendario asía que su número de instancia es 1. Usamos una sola lista de festivos cuyo identificador es su número de instancia que es 1. El dato de la semana laboral es 254 ya que todos los días de la semana son hábiles.

NOMBRE	VALOR
Identificador del objeto	102
Nombre del objeto	Calendario
Número de instancia de la clase	1
Descripción del objeto	Calendario de Control
Fecha Inicial	1/1/2015
Fecha Final	31/12/2015
Semana Laboral	254
Número de elementos de la lista	1
ID lista días festivos	1

A4.3 LISTA DE HORAS DE ACCIONES

Creamos 4 instancias de esta clase. Cada una corresponde con la hora que se repite los 7 días de la semana.

NOMBRE	VALOR
Identificador del objeto	103
Nombre del objeto	Horas 1
Número de instancia de la clase	1
Descripción del objeto	Control a las 8 horas
Número de elementos N de la lista	7
Día/Hora/Min	1/8/0, 2/8/0, 3/8/0, 4/8/0, 5/8/0, 6/8/0, 7/8/0

NOMBRE	VALOR
Identificador del objeto	104
Nombre del objeto	Horas 2
Número de instancia de la clase	2
Descripción del objeto	Control a las 14 horas
Número de elementos N de la lista	7
Día/Hora/Min	1/14/0, 2/14/0, 3/14/0, 4/14/0, 5/14/0, 6/14/0, 7/14/0

NOMBRE	VALOR
Identificador del objeto	105
Nombre del objeto	Horas 3
Número de instancia de la clase	3
Descripción del objeto	Control a las 16 horas
Número de elementos N de la lista	7
Día/Hora/Min	1/16/0, 2/16/0, 3/16/0, 4/16/0, 5/16/0, 6/16/0, 7/16/0

NOMBRE	VALOR
Identificador del objeto	106
Nombre del objeto	Horas 4
Número de instancia de la clase	4
Descripción del objeto	Control a las 22 horas
Número de elementos N de la lista	7
Día/Hora/Min	1/22/0, 2/22/0, 3/22/0, 4/22/0, 5/22/0, 6/22/0, 7/22/0

A4.4 LISTA OBJETOS

Definimos ahora las listas de los objetos. Para cada objeto usamos su identificador global para identificarlo, ya que si usamos su número de instancia tendríamos que especificar también el identificador de la clase a la que pertenece.

Consideramos que los identificadores de los objetos son:

BO1=18, BO2= 19, BO3=20, BO4=21, BO5= 22, BO6=23, BO7= 24, BO8= 25, BO9=26, BO10=27, BO11= 28, BO12=29, BO13=30, BO14= 31, BO15=32, BO16=33

AO1=50, AO2=51, AO3=52, AO4=53, AO5=54, AO6=55, AO7=56, AO8=57, AO9=58, AO10=59, AO11=60, AO12=61, AO13=62, AO14=63, AO15=64, AO16=65

Vamos a agrupar los objetos en grupos de 4 elementos así que necesitamos 8 instancias de esta clase.

NOMBRE	TIPO DATO
Identificador del objeto	107
Nombre del objeto	LO1
Número de instancia de la clase	1
Descripción del objeto	BO1-BO4
Número de elementos N de la lista	4
ID_OBJETOS	18, 19, 20, 21

NOMBRE	TIPO DATO
Identificador del objeto	108
Nombre del objeto	LO2
Número de instancia de la clase	2
Descripción del objeto	BO5-BO8
Número de elementos N de la lista	4
ID_OBJETOS	22, 23, 24, 25

NOMBRE	TIPO DATO
Identificador del objeto	109
Nombre del objeto	LO3
Número de instancia de la clase	3
Descripción del objeto	BO9-BO12
Número de elementos N de la lista	4
ID_OBJETOS	26, 27, 28, 29

NOMBRE	TIPO DATO
Identificador del objeto	110
Nombre del objeto	LO4
Número de instancia de la clase	4
Descripción del objeto	BO13-BO16

Número de elementos N de la lista	4
ID_OBJETOS	30, 31, 32, 33

NOMBRE	**TIPO DATO**
Identificador del objeto	111
Nombre del objeto	LO5
Número de instancia de la clase	5
Descripción del objeto	AO1-AO4
Número de elementos N de la lista	4
ID_OBJETOS	50, 51, 52, 53

NOMBRE	**TIPO DATO**
Identificador del objeto	112
Nombre del objeto	LO6
Número de instancia de la clase	6
Descripción del objeto	AO5-AO8
Número de elementos N de la lista	4
ID_OBJETOS	54, 55, 56, 57

NOMBRE	**TIPO DATO**
Identificador del objeto	113
Nombre del objeto	LO7
Número de instancia de la clase	7
Descripción del objeto	AO9-AO12
Número de elementos N de la lista	4
ID_OBJETOS	58, 59, 60, 61

NOMBRE	**TIPO DATO**
Identificador del objeto	114
Nombre del objeto	LO8
Número de instancia de la clase	8
Descripción del objeto	AO13-AO16

Número de elementos N de la lista	4
ID_OBJETOS	62, 63, 64, 65

A4.5 HORARIOS

Cada acción de control contiene:

- La instancia de la lista de objetos usada
- La instancia de la lista de horas usada
- El tipo de datos asociado
- El valor numérico de la acción

Puesto que hemos normalizado el tamaño del campo VALOR a 8 bytes y el resto de las informaciones usan 2 bytes cada una, cada acción contiene 14 bytes en total. Puesto que el campo datos en el mensaje de escritura

Los tipos de datos para las salidas binarias son Bool que tiene el identificador 1 y para las salidas binarias Float32L que tiene el identificador 22.

NOMBRE	**VALOR**
Identificador del objeto	115
Nombre del objeto	Horario 1
Número de instancia de la clase	1
Descripción del objeto	Horario Control 1
Activo	1
Calendario asociado	1
Número de elementos de la lista de excepciones	0
Lista días excepciones	
Número de acciones N	32
Lista de acciones de control	1/1/1/1, 2/1/1/0, 3/1/1/1, 4/1/1/0, 5/1/22/0, 6/1/22/50, 7/1/22/100, 8/1/22/100, 1/2/1/0, 2/2/1/1, 3/2/1/0, 4/2/1/1, 5/2/22/50, 6/2/22/10, 7/2/22/10, 8/2/22/10, 1/3/1/1, 2/3/1/0, 3/3/1/1, 4/3/1/0, 5/3/22/0, 6/3/22/50, 7/3/22/100, 8/3/22/100, 1/4/1/0, 2/4/1/0, 3/4/1/0, 4/4/1/0, 5/4/22/0, 6/4/22/0, 7/4/22/0, 8/4/22/0

BIBLIOGRAFÍA

[1] Open systems interconnection – Basic Model: the basic model. ISO/IEC7498-1

[2] Profibus y Profinet International www.profibus.com

[3] Modicon Modbus protocol reference Guide PI-MBUS-300 Rev .J

[4] Modbus Application Protocol Specification V1.1.b3 April 26, 2012

[5] Modbus over serial line specification & implementation guide V1.0 12/02/2002

[6] Modbus messaging on TCP/IP implementation guide Rev 1.0 8 May 2002

[7] ODVA incluye DeviceNet™, EtherNet/IP™, CompoNet™, and ControlNet™ basados sobre Common Industrial Protocol (CIP™) www.odva.org.[8] Fieldbus Foundation www.fieldbus.org

[9] "Industrial communication networks. Fieldbus specifications IEC61158 , IEC61784

[10] Ralph McKiewicz. IEC61850. Heights,MI,USA. 2002.

[11] DNP Distributed Network Protocol www.dnp.org

[12] IEC60870-5-101 2003 Telecontrol equipment and systems Part 5-101 Transmission protocols – Companion standard for basic telecontrol tasks

[13] IEC60870-5-104 2006 Telecontrol equipment and systems Part 5-104 Transmission protocols-Network access for IEC 60870-101 using standard transport profiles

[14] PLX8X-MNET-61850 Modbus TCP/IP to IEC 61850 Gateway. www.prosoft-technology.com

[15] PowerLogic G3200 Modbus to IEC 61850 Gateway. www.Schneider-electric.com

[16] Organización KNX www.knx.org

[17] Echelon Corporation www.echelon.com

[18] Lonmark International www.lonmark.org

[19] Standard 135-2012 -- BACnet®--A Data Communication Protocol for Building Automation and Control Networks (www.ashrae.org)

[20] Honeywell EAGLE www.centraline.com

[21] Librería libmodbus libmodbus.org

[22] Librería FreeModbus www.freemodbus.org

[23] Java Modbus library http://sourceforge.net/projects/jamod

[24] Modbus library for Labview http://www.ni.com/example/29756/en

[25] PhpModbus library https://code.google.com/p/phpmodbus

[26] Implementation of Modbus Communication Protocol Based on ARM Coretx-MO Kelong Wang!, Daogang Peng!,2, Lei Song!, Hao Zhang 2014 IEEE International Conference on System Science ang Engineering (JCSSE) July 11-13 2014,Shanghai, China

[27] Research and Implementation of Networked Data Acquisition System Based on Cortex-M3 Qingpeng Sheng, Shubo Qiu Proceeding of the IEEE International Conference on Information and Automation Yinchuan, China, August 2013

[28] Design and Realization of Modbus Protocol Based on Embedded Linux System PENG Dao-gang, ZHANG Hao, YANG Li, LI Hui The 2008 International Conference on Embedded Software and Systems Symposia (ICESS2008)

[29] A remote monitoring system for voltage, current, power and temperature measurements E. Barakat, N. Sinno, C. Keyrouz Eight International Conferences on Material Sciences (CSM8-ISM5)

[30] Exploiting Modbus Protocol in Wired and Wireless Multilevel Communication Architecture Giuliano B. M. Guarese, Felipe G. Sieben, Thais Webber, Marcos R. Dillenburg*, César Marcon 2012 Brazilian Symposium on Computing System Engineering

[31] A Novel Modbus RTU-Based Communication System for Adjustable Speed Drives Hu Sideng, Zhao Zhengming, Zhang Yingchao, Wang Shuping IEEE Vehicle Power and Propulsion Conference (VPPC), September 3-5, 2008, Harbin, China

[32] Proposal of a Communication Protocol for Smart Sensory Systems Juraj Ďuďák, Martin Skovajsa and Ivan Sládek

[33] Exploiting Modbus Protocol in Wired and Wireless Multilevel Communication Architecture Giuliano B. M. Guarese, Felipe G. Sieben, Thais Webber, Marcos R. Dillenburg, César Marcon 2012 Brazilian Symposium on Computing System Engineering

[34] A. Flammini et al. Wired and wireless sensor networks for industrial applications. Microelectronics Journal. v. 40, pp. 1322-1336, 2009.

[35] Research and Implementation of Collision Detection Based on Modbus Protocol Yinglan Fang, Xianfeng Han and Bing Han Journal of Engineering Science and Technology Review 6 (1) (2013) 91-96

[36] Weng Jiannian, Zhang Hao, Peng Daogang, Li Hui, "On embedded ARM based Modbus TCP Protocol and its Implementation", Computer Applications and Software, V01.26(10), pp.36-38, 2009

[37] LIU Wen-jun, LI Xiang-yang, "Design a Class Modbus Protocol Based on Automatic Addressing", Science Technology and Engineering, Vol.12(6), pp.1412-1415, 2011

[38] Yung-Hsiang Liu, Kuo-Hsin Chu, wen-Cheng Liang, Method for setting addresses of slave devices in communications network. Patent US 9015267 B2

[39] Secure SCADA communication by using a modified key management scheme, Abdalhossein Rezai, Parviz Keshavarzi, ZahraMoravej. ISA Transactions 52 (2013) 517-524

[40] Accurate modelling of Modbus/TCP for intrusion detection in SCADA systems, Niv Goldenberg, Avishai Wool. . International Journal of critical infrastructure protection 6 (2013) 63-75

[41] An experimental investigation of malware attacks on SCADA systems, *Igor Nai Fovinoa, Andrea Carcanoa, Marcelo Maseraa, Alberto Trombettab.* . International Journal of critical infrastructure protection 2 (2009) 139-145

[42] An unsupervised anomaly-based detection approach for integrity attacks on SCADA systems, Abdulmohsen Almalawi, Xinghuo Yu, Zahir Tari, Adil Fahad, Ibrahim Khalil. Computer & security 46 (2014) 94-110

[43] Attack taxonomies for the Modbus protocols, Peter Huitsing, Rodrigo Chandia, Mauricio Papa, Sujeet Shenoi. International Journal of critical infrastructure protection 1 (2008) 37-44

[44] Security of industrial sensor network-based remote substations in the context of the Internet of Things. Cristina Alcaraz, Rodrigo Román, Pablo Najera, Javier López. Ad Hoc Networks 11 (2013) 1091-1104

[45] A survey of cyber security management in industrial control systems. William Knowles, Daniel Prince, David Hutchison y otros. International Journal of critical infrastructure protection 9 52-80

[46] A control system testbed to validate critical infrastructure protection concepts. Thomas Morris, Anurag Srivastava, Bradley Reaves y otros. International Journal of critical infrastructure protection 4 88-103

[47] Community Research and Development Information Service, Critical Information Infrastructure Research Coordination (CI2RCO), European Commission, Luxembourg (cordis. europa.eu/projects/rcn/79305_en.html), 2007.

[48] Community Research and Development Information Service, Critical Utility Infrastructural Resilience (CRUTIAL), European Commission, Luxembourg (cordis.europa.eu/ projects/rcn/79318_en.html), 2008.

[49] Community Research and Development Information Service, Emergency Management in Large Infrastructures (EMILI), European Commission, Luxembourg (cordis.europa.eu/ project/rcn/93509_en.html), 2012.

[50] Community Research and Development Information Service, Semantically Enhanced Resilient and Secure Critical Infrastructure Services (SERSCIS), European Commission, Luxembourg (cordis.europa.eu/projects/rcn/88496_en.html), 2011.

[51] Community Research and Development Information Service, Increasing Security and Protection through Infrastructure Resilience (INSPIRE), European Commission, Luxembourg (cordis.europa.eu/projects/rcn/87757_en.html), 2011.

[52] Community Research and Development Information Service, European Network for the Security of Control and Real-Time Systems (ESCoRTS), European Commission, Luxembourg (cordis.europa.eu/project/rcn/87538_en.html), 2010.

[53] Community Research and Development Information Service, European Risk Assessment and Contingency Planning Methodologies for Interconnected Energy Networks (EURACOM), European Commission, Luxembourg (cordis. europa.eu/project/rcn/92076_en.html), 2011.

[54] IEC 61131-3:2013 Programmable controllers – Part 3: Programming languajes. Webstore.iec.ch/publication/4552

[55] KNX System Specifications – Interworking – Datapoint Types v1.5.00

[56] LONMARK® SNVT and SCPT Master List Version 14 Revision 00 December 2012

[57] LONMARK® SNVT Master List Version 14 Revision 00 December 2012

[58] LONMARK Resource Files types.lonmark.org

[59] LONMARK Standar Enumeration Master List. Version 14 Revision 00 December 2012

[60] Zigbee Cluster Library December 16, 2014

[61] Object Messaging Specification for the MODBUS/TCP Protocol. Version 1.1 November 08, 2004

[62] Alternate Transport Mechanism for Modbus/TCP -- Utilizing “Existing” Register Read/Write Function Codes to Support the Modbus/TCP Object Messaging Protocol