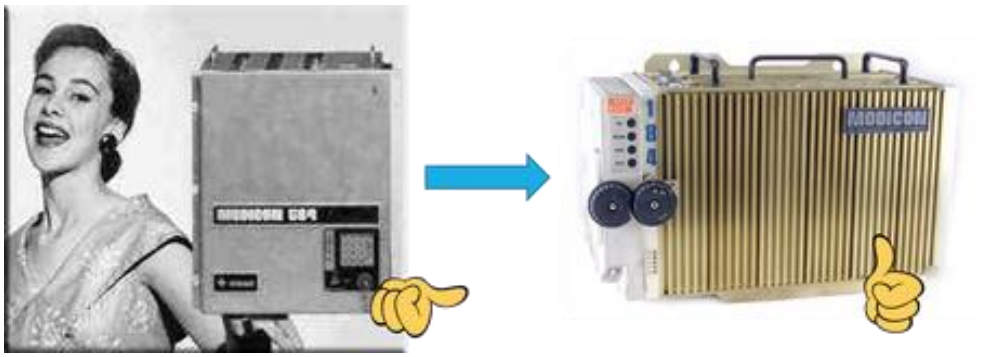
Tema 7. Protocolo MODBUS

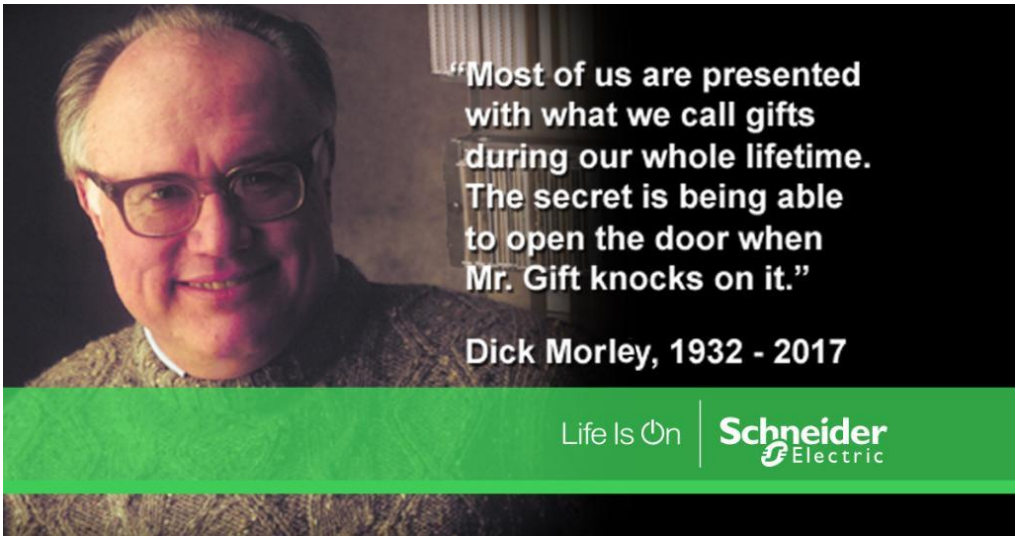
1.- Introducción

La historia del **PLC** (**Controlador Lógico Programable**), se inició a finales de los años 60, cuando apareció el propósito de eliminar el enorme costo que significaba el reemplazo de un sistema de control, basados en **relés**.

En **1968** La empresa **Bedford Associates** (Bedford, MA), propuso a General Motors, un sistema que denominó **Modular Digital Controller o MODICON**. El **MODICON 084** fue el primer PLC producido comercialmente. Con este sistema de control, la producción en las empresas, se podía variar en función de las demandas del mercado.



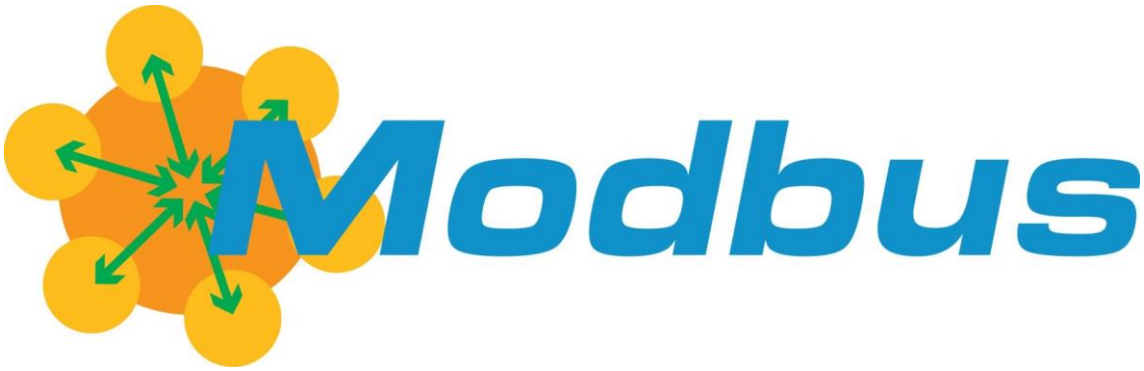
Este nuevo controlador (**PLC**) tenía que ser fácilmente programable, su vida útil tenía que ser larga y poder resistir en ambientes difíciles. Esto se logró con técnicas de programación conocidas y reemplazando los **relés** por elementos de **estado sólido**. En **1973** apareció el primer protocolo de comunicación entre PLC's: **MODBUS** de **Modicon**. Esto permitió a los PLC's, estar alejados de las maquinas que controlaban y comunicarse entre ellos.



La mayoría de nosotros recibimos lo que llamamos regalos durante toda nuestra vida. El secreto es poder abrir la puerta cuando el Sr. Regalo llama.

Desde que Modicon creó MODBUS para sus PLC's, no ha dejado de extenderse su uso haciéndose **estándar de facto** por tener las siguientes características:

- Es **público** y **gratuito**.
- Es **fácil** de implementar y no requiere mucho desarrollo.
- Maneja **bloques de datos**.
- Tiene una estructura de comunicación **maestro-esclavo**.



Por todo lo anterior se considera el protocolo adecuado para cubrir las necesidades requeridas para la industria.

2.- Variantes de MODBUS

- **Modbus RTU** — Es la implementación más común disponible para Modbus. Se utiliza en la comunicación serie y hace uso de una representación binaria compacta de los datos para el protocolo de comunicación. El formato RTU sigue a los comandos/datos con una **suma de comprobación de redundancia cíclica** (**CRC**) como un mecanismo de comprobación de errores para garantizar la fiabilidad de los datos. Un mensaje Modbus RTU debe transmitirse continuamente sin vacilaciones entre caracteres. Los mensajes Modbus son entramados (separados) por períodos inactivos (silenciosos) de 3.5 veces, el tiempo que se tarda en enviar un byte.

- **Modbus ASCII** — Se utiliza en la comunicación serie y hace uso de caracteres ASCII para el protocolo de comunicación. El formato ASCII utiliza un checksum de **control de redundancia longitudinal** (**LRC**). Los mensajes Modbus ASCII están encabezados por los dos puntos (":") y finalizados por los caracteres CR y LF.

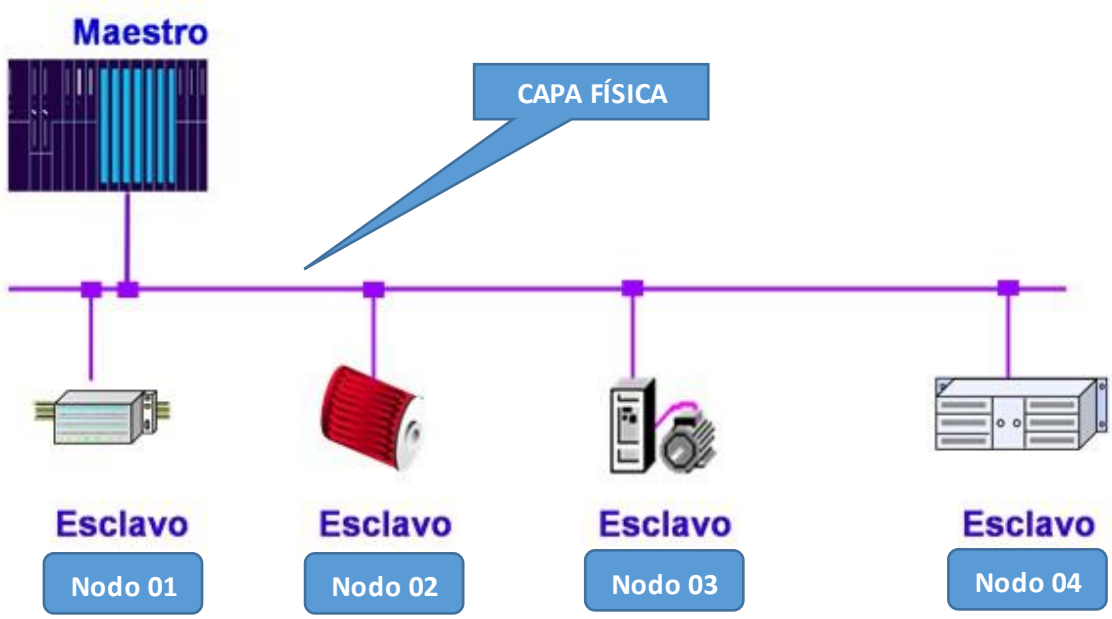
- **Modbus TCP/IP o Modbus TCP** — Se trata de una variante Modbus utilizada para comunicaciones a través de redes TCP/IP,

conectándose a través del puerto 502.2. No requiere un cálculo de suma de verificación (checksum), ya que las capas inferiores ya proporcionan protección de checksum.

- **Modbus sobre TCP/IP o Modbus sobre TCP o Modbus RTU/IP** — Esta es una variante de Modbus que difiere del Modbus TCP en que se incluye una suma de comprobación en la carga útil como en Modbus RTU.

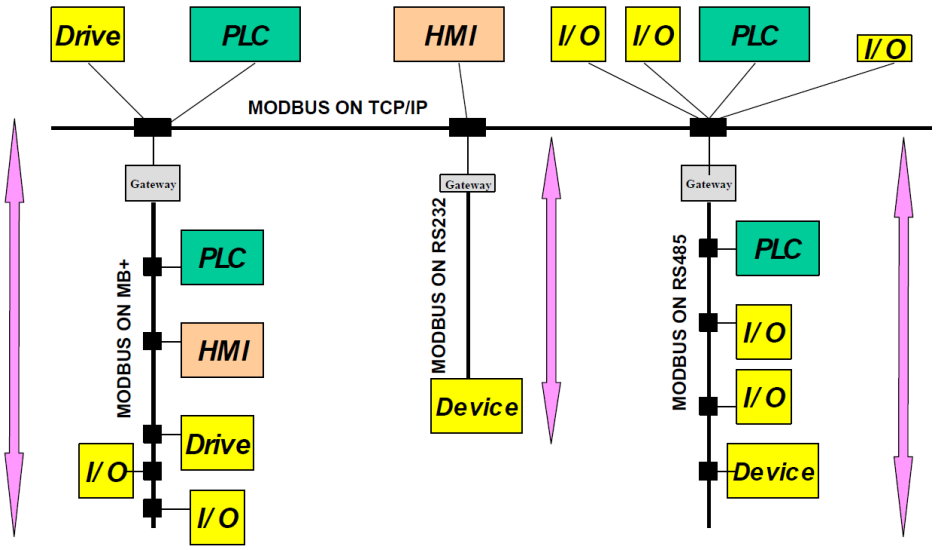
3.- Capa física en MODBUS RTU

La capa física se encarga de la transmisión de bits al canal de comunicación. Define los niveles de la señal eléctrica con la que se trabajará y la velocidad de transmisión.



MODBUS RTU usa una capa física tipo **BUS**, por lo que es muy sencillo **añadir o quitar esclavos**. El error en uno de los esclavos **no afectará al resto**.

La norma **RS422** (4 hilos) o **RS485** (2 hilos) suelen ser las elegidas para implementar las conexiones físicas. Habitualmente, por el BUS también transitan las líneas de alimentación como por ejemplo GND y +5V o GND y +24V, que usaran los esclavos para su funcionamiento.



Como podemos apreciar en la anterior ilustración, las capas físicas de **MODBUS** pueden trabajar conjuntamente, usando los oportunos dispositivos para convertir las señales. En la ilustración, **Gateway**.

4.- Formato de tramas MODBUS RTU

Una trama Modbus está compuesta por una **Unidad de Datos de Aplicación** (ADU), que incluye una **Unidad de Datos de Protocolo** (PDU):

Unidad de datos de aplicación (ADU)		
Dirección	PDU	Comprobación de errores

Unidad de datos de protocolo (PDU)	
Código de función	Datos

El **modelo de datos** y las **llamadas de función** son idénticas para todas las variantes de protocolos que vimos en el punto 2 de este tema; sólo la encapsulación es diferente. Sin embargo, las variantes no son interoperables, ni sus formatos de trama tampoco.

El tamaño máximo de la ADU MODBUS, está limitado por la restricción de tamaño, heredada de la primera implementación MODBUS en comunicación serie, que corresponde de un total de **256 bytes**. Por lo tanto, el tamaño máximo para la PDU en comunicación serie, será:

Tamaño máximo PDU= 256 bytes - Dirección (1 byte) - CRC (2 bytes) = 253 bytes.

Las tramas que envía el maestro y las que recibe del esclavo, son ligeramente diferentes:

Trama MODBUS RTU General

ID ESCLAVO 1byte	FUNCIÓN 1 byte	DATOS N bytes	CRC 2 bytes

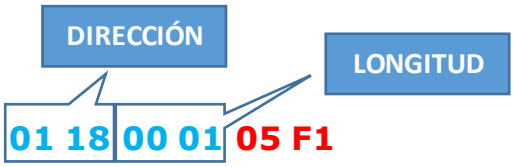
Trama MODBUS RTU Maestro

ID ESCLAVO 1 byte	FUNCIÓN 1 byte	DIRECCIÓN 2 bytes	LONGITUD 2 bytes	CRC 2 bytes

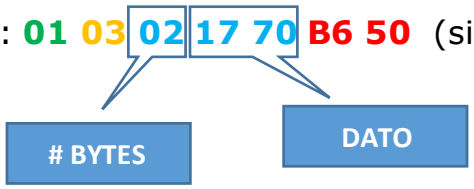
Trama MODBUS RTU Esclavo

ID ESCLAVO 1 byte	FUNCIÓN 1 byte	# BYTES 1 bytes	RESPUESTA N bytes	CRC 2 bytes

Ejemplos de tramas MODBUS RTU

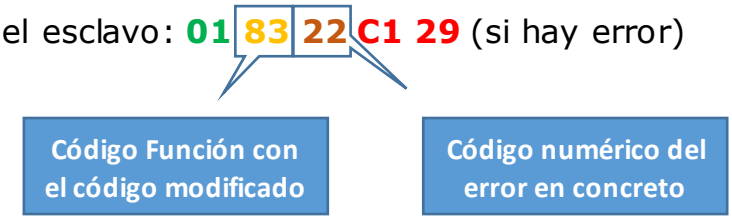


Trama Enviada por el maestro: **01 03 01 18 00 01 05 F1**



Respuesta del esclavo: **01 03 02 17 70 B6 50** (si no hay error)

Si ha habido algún problema y el esclavo no ha podido ejecutar la orden del maestro o proporcionar la información pedida por el maestro, el esclavo responde una trama distinta.

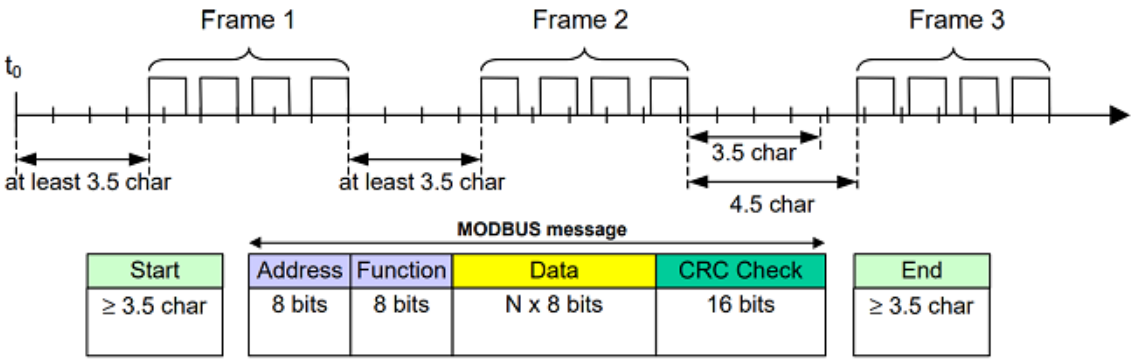


Respuesta del esclavo: **01 83 22 C1 29** (si hay error)

Pero… ¿cómo ha modificado el esclavo el código de la función? ¿Por qué lo ha hecho?

- La modificación consiste en poner a 1 el bit más significativo en el código de función original.
  Código original de la función: **03**-> **0 0 0 0 0 1 1**
  Código modificado de la función: **1 0 0 0 0 0 1 1 -> 83**

- Lo ha hecho para que el Maestro se dé cuenta de que ha habido algún problema. Este problema será identificado y concretado por el código numérico del error en el tercer byte.



Desde el punto de vista dinámico, las tramas MODBUS, deben estar flanqueadas, por al menos un silencio que dure como mínimo, 3,5 veces lo que dura la transmisión de un byte. Esto se puede observar en al anterior ilustración.

4.1.- Dirección

Este campo tiene un tamaño de **1 byte**, es decir, **2 cifras en hexadecimal**: 00, 01, 02, 4B, 2A, etc.

Cada una de estas direcciones determina la dirección del nodo al que irá dirigida la trama, a través del medio físico que se esté usando: RS232, RS422 o RS485. Esto puede llevar a error de concepto si pensamos que la trama la oirá **solamente** el nodo al que va dirigido. Lo que ocurre es que **la oyen todos**, pero **solo contestará**, aquel nodo que **coincida** su **dirección**, con la **dirección de la trama leída**.

Permite un máximo de **63** esclavos que van desde el **01** hasta el **3F**. La dirección **00** se reserva para mensajes de tipo **Broadcast** (mensaje, sin respuesta, del maestro a todos los esclavos)

4.2.- Función

Este campo de 8 bits (1 byte), contiene el tipo de acción que ha de ejecutarse. Los códigos vàlidos van desde el 01 hasta el FF en hexadecimal (del 1 al 255 en decimal). Hay tres categorias de códigos de función:

4.2.1.- Códigos de funciones públicas

Códigos del 1 al 127 en decimal (del 01 al 7F en hexadecimal), excepto los códigos definidos por el usuario que también pueden estar en este rango. Estos códigos están

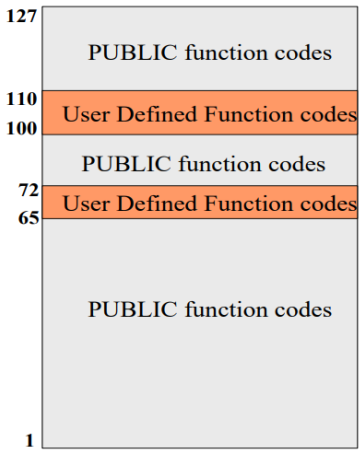
perfectamente validados y aprobados por la corporación modbus.org (propietaria del protocolo desde 2004).

4.2.2.- Códigos de función definidos por el usuario

En dos rangos de 65 a 72 en decimal (41 a 48 en hexadecimal) y de 100 a 110 en decimal (64 a 6E en hexadecimal). Estos códigos son definibles por el usuario para implementar funciones que no se incluyen en el protocolo. No hay garantias de que la nueva función sea única.

4.2.3.- Códigos de función reservados

Utilizados por algunas compañías para productos propios y no disponibles para uso público.



Mapa de Códigos de funciones

Código Función	Función	Descripción	Tipo	
0x01	**Read Coils**	**Leer salidas digitales**	**Acceso de 1 bit**	Acceso a Datos
0x02	**Read Discrete Inputs**	**Leer entradas digitales**		
0x05	**Write Single Coil**	**Escribir un bit de salida**		
0x0F	**Write Multiple Coils**	**Escribir n bits de salida**		
0x03	**Read Holding Registers**	**Leer n registros de salida**	**Acceso de 16 bits**	
0x04	**Read Input Register**	**Leer n registros de entrada**		
0x06	**Write Single Register**	**Escribir un registro**		
0x10	**Write Multiple Registers**	**Escribir n registros**		
0x20	Read File record	Leer fichero	Acceso a ficheros	
0x21	Write File record	Escribir fichero		

0x07	Read Exception status	Lectura del estado de excepción	Diagnóstico
0x08	Diagnostic	Diagnóstico (loopback)	

Códigos de Función públicos mas comunes

A continuación realizamos una explicación, con ejemplos incluidos, de las funciones públicas, usadas para la gestión de **bits** y de **registros**:

4.2.3.1.- (0x01) Read Coils. Leer salidas digitales.

Este código de función se utiliza para leer de **1** a **2000** estados de **salidas digitales contiguas**, en un dispositivo MODBUS esclavo.

La PDU de solicitud (Request en inglés), especifica la dirección de inicio, es decir, la dirección de la primera salida y el número de salidas, a partir de ella.

En la PDU, las salidas se direccionan a partir de cero. Por lo tanto, las salidas numeradas del 1 al 16 en el esclavo, se direccionan como 0-15 en la PDU enviada.

Request

Function code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of coils	2 Bytes	1 to 2000 (0x7D0)

Las salidas, en el mensaje de respuesta, se mostraran como un bit por cada salida. El estado se indicará como 1 = ON y 0 = OFF.

El LSB del primer byte de datos contiene la salida direccionada en la consulta. Las otras salidas siguen hacia el extremo de orden superior de este byte, y de la misma manera, en bytes posteriores.

Si la cantidad de salidas devueltas, no es un múltiplo de ocho (ya que en cada byte devuelto se pueden representar el estado de 8 salidas), los bits restantes en el byte de datos final, se rellenarán con ceros (hacia el extremo de orden superior del byte). El campo Byte Count especificará, de todas formas, la cantidad de bytes completos.

Response

Function code	1 Byte	**0x01**
Byte count	1 Byte	**N***
Coil Status	**n** Byte	n = N or N+1

***N** = Número de salidas/8, si el resto de la anterior división, es diferente de 0, n=N+1

Ejemplo

*Veamos en este ejemplo, como sería la trama enviada y la respuesta, si el maestro tratara de leer el estado de las salidas digitales de la **20** a la **38** (ambas inclusive), de un esclavo cualquiera.*

*Suponemos que en este esclavo, las salidas se direccionan a partir del **1**. Por esto la dirección de la primera salida a leer, **no** será la **20** sino la **19** (0x13).*

*También serán **19** salidas contiguas, las que se desean leer (de 20 a 38 incluidos ambos). Por tanto, con **2** bytes podemos representar el estado de **16** salidas. Nos quedamos cortos. La respuesta del esclavo, tendrá pues, **3 bytes de datos**.*

El estado de las salidas **27** a **20** se muestra como el valor de byte **0xCD** (este valor es por ejemplo) o en binario **1100 1101**. La salida **27** es el **MSB** de este byte y la salida **20** es el **LSB**.

MSB

Byte 1 de datos en la respuesta								
Dirección salidas en el esclavo	27	26	25	24	23	22	21	20
Valor por ejemplo en cada salida	1	1	0	0	1	1	0	1
Valor en hexadecimal del Byte	C				D			

LSB

El siguiente byte contiene el estado de las salidas **28** a **35**, de derecha a izquierda, de la misma manera que en el Byte 1. En este segundo byte suponemos que tenemos, por ejemplo, un valor de **0x6B**.

Byte 2 de datos en la respuesta								
Dirección salidas en el esclavo	35	34	33	32	31	30	29	28
Valor por ejemplo en cada salida	0	1	1	0	1	0	1	1
Valor en hexadecimal del Byte	6				B			

En el tercer byte de datos, leeremos el estado de las salidas **38** a **36**. Suponemos que tenemos los estados **1**, **0** y **1** en las respectivas salidas **36**, **37** y **38**. Los cinco bits de orden superior restantes, se rellenan con ceros.

Byte 3 de datos en la respuesta								
Dirección salidas en el esclavo	X	X	X	X	X	38	37	36
Valor por ejemplo en cada salida	0	0	0	0	0	1	0	1
Valor en hexadecimal del Byte	0				5			

Con todo lo expuesto, podemos comprobar cómo sería la trama enviada por el maestro y la respuesta del esclavo.

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**01**	Function	**01**
Starting Address Hi	**00**	Byte Count	**03**
Starting Address Lo	**13**	Outputs status 27-20	**CD**
Quantity of Outputs Hi	**00**	Outputs status 35-28	**6B**
Quantity of Outputs Lo	**13**	Outputs status 38-36	**05**

4.2.3.2.- (0x02) Read Discret Inputs. Leer entradas digitales.

Este código de función se utiliza para leer de **1** a **2000** estados de **entradas digitales contiguas**, en un dispositivo MODBUS esclavo.

En la petición que envía el maestro, especifica cual es la dirección inicial y el número de entradas a leer (incluyendo la inicial).

En la PDU, las entradas digitales se direccionan a partir de cero. Por lo tanto, las entradas digitales numeradas del 1 al 16, en el esclavo, se direccionaran entre 0 y 15.

Las entradas discretas en el mensaje de respuesta se empaquetan como una entrada por bit del campo de datos. El estado se indica como 1 = ON; 0 = APAGADO. El LSB del primer byte de datos contiene la entrada dirigida en la consulta. Las otras entradas siguen hacia el extremo de orden superior de este byte, y de orden inferior a orden superior en bytes posteriores.

Si la cantidad de entrada devuelta no es un múltiplo de ocho, los bits restantes en el byte de datos final se rellenarán con ceros (hacia el extremo superior del byte). El campo Byte Count especifica la cantidad de bytes completos de datos.

Ejemplo

*Veamos en este ejemplo, como sería la trama enviada y la respuesta, si el maestro tratará de leer el estado de las entradas digitales de la **197** a la **218** (ambas inclusive), de un esclavo cualquiera.*

*Suponemos que en este esclavo, las salidas se direccionan a partir del **1**. Por esto la dirección de la primera entrada a leer, **no** será la **197** sino la **196** (0xC4).*
*También serán **22** (0x16) salidas contiguas, las que se desean leer (de 197 a 218 incluidos ambos). Por tanto, con **2** bytes podemos representar el estado de **16** entradas. Nos quedamos cortos. La respuesta del esclavo, tendrá pues, **3 bytes de datos**.*

El estado de las entradas **204** a **197** se muestra como el valor de byte **0xAC** (este valor es por ejemplo) o en binario **1010 1101**. La salida **204** es el **MSB** de este byte y la salida **197** es el **LSB**.

Byte 1 de datos en la respuesta								
Dirección entradas en el esclavo	204	203	202	201	200	199	198	197
Valor por ejemplo en cada entrada	1	0	1	0	1	1	0	0
Valor en hexadecimal del Byte	A				C			

LSB

El siguiente byte contiene el estado de las salidas **205** a **212**, de derecha a izquierda, de la misma manera que en el Byte 1. En este

segundo byte suponemos que tenemos, por ejemplo, un valor de **0xDB**.

Byte 2 de datos en la respuesta								
Dirección entradas en el esclavo	212	211	210	209	208	207	206	205
Valor por ejemplo en cada entrada	1	1	0	1	1	0	1	1
Valor en hexadecimal del Byte	D				B			

En el tercer byte de datos, leeremos el estado de las salidas **213** a **218**. Suponemos que tenemos los estados **1**, **0**, **1**, **0**, **1** y **1** en las respectivas salidas **213**, **214**, **215**, **216**, **217** y **218**. Los 2 bits de orden superior restantes, se rellenan con ceros.

Byte 3 de datos en la respuesta								
Dirección salidas en el esclavo	X	X	218	217	216	215	214	213
Valor por ejemplo en cada salida	0	0	1	1	0	1	0	1
Valor en hexadecimal del Byte	3				5			

Con todo lo expuesto, podemos comprobar cómo sería la trama enviada por el maestro y la respuesta del esclavo.

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**02**	Function	**02**
Starting Address Hi	**00**	Byte Count	**03**
Starting Address Lo	**C4**	Inputs Status 204-197	**AC**
Quantity of Inputs Hi	**00**	Inputs Status 212-205	**DB**
Quantity of Inputs Lo	**16**	Inputs Status 218-213	**35**

4.2.3.3.- (0x03) Read Holding Registers. Leer n registros de salida.

Este código de función se utiliza para leer el contenido de un bloque contiguo de **registros de salida** en un dispositivo remoto. La PDU de solicitud especifica la **dirección del registro inicial** y el número de registros, a **partir del inicial**. En la PDU, los registros se direccionan a partir de cero. Por lo tanto, los registros numerados del 1 al 16, en el esclavo, se direccionaran de 0 a 15.

Los datos de registro en el mensaje de respuesta se empaquetan en dos bytes por registro, con el contenido binario justificado a la derecha dentro de cada byte. Para cada registro, el primer byte contiene los bits de orden superior y el segundo contiene los bits de orden inferior.

Ejemplo

Veamos en este ejemplo, como sería la trama enviada y la respuesta, para leer los registros 108, 109 y 110, del esclavo.

La primera dirección a leer será la 107 (0x6B) y la cantidad de registros a leer serán 3 (0x03)

Supongamos que los contenidos de los registros a leer son los siguientes:

Dirección registros en el esclavo	108	109	110
Valor por ejemplo en cada registro	02 2B	00 00	00 64

Tenemos en cuenta que en la respuesta del esclavo, tendremos un total de 6 bytes de datos.

Con todo lo expuesto, podemos comprobar cómo sería la trama enviada por el maestro y la respuesta del esclavo.

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	03	Function	03
Starting Address Hi	00	Byte Count	06
Starting Address Lo	6B	Register value Hi (108)	02
No. of Registers Hi	00	Register value Lo (108)	2B
No. of Registers Lo	03	Register value Hi (109)	00
		Register value Lo (109)	00
		Register value Hi (110)	00
		Register value Lo (110)	64

4.2.3.4.- (0x04) Read Input Register. Leer n registros de entrada

Este código de función se utiliza para **leer** de 1 a 125 **registros de entrada** contiguos en un dispositivo esclavo MODBUS.

La PDU de solicitud especifica la dirección de registro inicial y el número de registros a leer. Los registros de PDU se direccionan comenzando en cero. Por lo tanto, los registros de entrada numerados del 1 al 16 son direccionados como 0-15, en la PDU.

Los datos del registro en el mensaje de respuesta se agrupan en dos bytes por registro, con el contenido binario justificado a la derecha dentro de cada byte. Para cada registro, el primer byte contiene el bit de orden superior y el segundo contiene los bits de orden inferior.

Ejemplo

Veamos en este ejemplo, como sería la trama enviada y la respuesta, para leer solo un registro de dirección en el esclavo de 0009.

Suponiendo que en el esclavo, se direccionen a partir del 1, en la petición del maestro, se referenciará la dirección 0008. La cantidad de registros a leer será 0001. Suponiendo que el contenido de este registro sea 0x0A (10 en decimal), la trama enviada y la respuesta será:

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	04	Function	04
Starting Address Hi	00	Byte Count	02
Starting Address Lo	08	Input Reg. 9 Hi	00
Quantity of Input Reg. Hi	00	Input Reg. 9 Lo	0A
Quantity of Input Reg. Lo	01		

4.2.3.5.- (0x05) Write Single Coil. Escribir un bit de salida

Este código de función se utiliza para escribir una única salida en ON u OFF en un dispositivo esclavo MODBUS. El estado ON / OFF solicitado se especifica mediante una constante en el campo de datos de la trama enviada. Un valor de **0xFF00** solicita que la salida esté a ON. Un valor de **0x0000** solicita que esté a OFF.

La Solicitud de PDU especifica la dirección de la salida a forzar. Las salidas se direccionan comenzando en cero. Por lo tanto, la salida numerada 1 se direcciona como 0.

Ejemplo

Veamos en este ejemplo, como sería la trama enviada y la respuesta, para activar (poner a ON) la salida digital 173 del esclavo MODBUS.

Suponemos que en el esclavo, las salidas se direccionan desde 1, por tanto, la dirección que usaremos en la trama enviada será la 172 (0x00AC). La trama respuesta, si no ha habido ningún problema, será idéntica a la trama enviada por el maestro (ECO).

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**05**	Function	**05**
Output Address Hi	**00**	Output Address Hi	**00**
Output Address Lo	**AC**	Output Address Lo	**AC**
Output Value Hi	**FF**	Output Value Hi	**FF**
Output Value Lo	**00**	Output Value Lo	**00**

4.2.3.6.- (0x06) Write Single Register. Escribir un Registro.

Este código de función se utiliza para escribir un solo registro de en un dispositivo remoto.

La PDU de solicitud, especifica la dirección del registro que se va a escribir. Los registros se direccionan comenzando en cero. Por lo tanto, el registro numerado 1 se direcciona como 0, suponiendo que en el esclavo, los registros se enumeren desde 1. La respuesta normal es un eco de la solicitud, devuelto después de que el contenido del registro haya sido escrito.

Ejemplo

Veamos en este ejemplo, como sería la trama enviada y la respuesta, para escribir el valor 0x0003 en el registro 0x0002 del esclavo.

La dirección que usaremos en la PDU enviada, será la 0x0001, suponiendo que en el esclavo, los registros se direccionen a partir de la dirección 1. Con todo esto:

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**06**	Function	**06**
Register Address Hi	**00**	Register Address Hi	**00**
Register Address Lo	**01**	Register Address Lo	**01**
Register Value Hi	**00**	Register Value Hi	**00**
Register Value Lo	**03**	Register Value Lo	**03**

4.2.3.7.- (0x0F) Write Multiple Coils. Escribir n bits de Salida.

Este código de función se utiliza para escribir múltiples bits de salida en el dispositivo esclavo MODBUS, cada uno de ellos a ON o a OFF. La Solicitud de PDU especifica las referencias de las salidas a escribir. Si las salidas en el esclavo, se direccionan desde 1, en la PDU enviada por el Maestro, se referenciarán desde 0.

Los estados ON / OFF de cada salida, se especifican por el contenido del campo de datos de la solicitud. Un '1' lógico en una posición de bit del campo, solicita que la salida correspondiente esté ON. Un '0' lógico solicita que esté a OFF.

En la petición del maestro, tendremos que incluir también, el número de bytes de datos que necesitamos.

La respuesta normal devuelve el código de función, la dirección de inicio y la cantidad de salidas forzadas.

Ejemplo

Veamos en este ejemplo, como sería la trama enviada y la respuesta, para escribir una serie de 10 salidas comenzando en la salida de dirección 20.

*La dirección que usaremos será la 19 (**0x13**), suponiendo que en el esclavo, las salidas digitales se direccionen a partir de la dirección 1.*

Como tenemos 10 (0x0A) salidas, necesitaremos 2 bytes (en un byte, podremos activar o desactivar, 8 salidas digitales).

Supongamos que queremos que las salidas queden en el siguiente estado:

Valores deseados de las 10 salidas digitales a partir de la dirección 20								
Dirección salidas en el esclavo	27	26	25	24	23	22	21	20
Valor por ejemplo en cada salida	1	1	0	0	1	1	0	1
Valor en hexadecimal del Byte	C				D			
Dirección salidas en el esclavo	35	34	33	32	31	30	29	28
Valor por ejemplo en cada salida	x	x	x	x	x	x	0	1
Valor en hexadecimal del Byte	0				1			

*El primer byte transmitido (**0xCD**) direcciona las salidas 27-20, y el bit menos significativo se dirige a la salida más baja (20) de este conjunto.*

*El siguiente byte transmitido (**0x01** hexadecimal) direcciona las salidas 29-28, y el bit menos significativo se dirige a la salida más baja (28) de este conjunto. **Los bits no utilizados en el último byte de datos deben llenarse con ceros**.*

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**0F**	Function	**0F**
Starting Address Hi	**00**	Starting Address Hi	**00**
Starting Address Lo	**13**	Starting Address Lo	**13**
Quantity of Outputs Hi	**00**	Quantity of Outputs Hi	**00**
Quantity of Outputs Lo	**0A**	Quantity of Outputs Lo	**0A**
Byte Count	**02**		
Outputs Value Hi	**CD**		
Outputs Value Lo	**01**		

4.2.3.8.- (0x10) Write Multiple Registers. Escribir n registros de Salida.

Este código de función se utiliza para escribir un bloque de registros contiguos (1 a 123 registros) en un dispositivo esclavo de MODBUS.

Los valores a introducir en cada registro, se especifican en el campo de datos de la trama enviada. Los datos se agrupan en dos Bytes por registro.

La respuesta normal devuelve el código de función, la dirección de inicio y la cantidad de registros que se han escrito.

Ejemplo

*Veamos un ejemplo en el que escribiremos en dos registros a partir de la dirección **0x0002**, los valores **0x000A** y **0x0102**.*

*Suponiendo que en el esclavo, los registros se direccionan a partir del 1, en la trama enviada, pondremos como dirección del primer registro a escribir, **0x0001**. El número de registros a escribir será de 2, por tanto 4 bytes.*

Con todo lo expuesto, las PDU enviadas y recibidas serán:

Request		Response	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	10	Function	10
Starting Address Hi	00	Starting Address Hi	00
Starting Address Lo	01	Starting Address Lo	01
Quantity of Registers Hi	00	Quantity of Registers Hi	00
Quantity of Registers Lo	02	Quantity of Registers Lo	02
Byte Count	04		
Registers Value Hi	00		
Registers Value Lo	0A		
Registers Value Hi	01		
Registers Value Lo	02		

4.3.- Datos

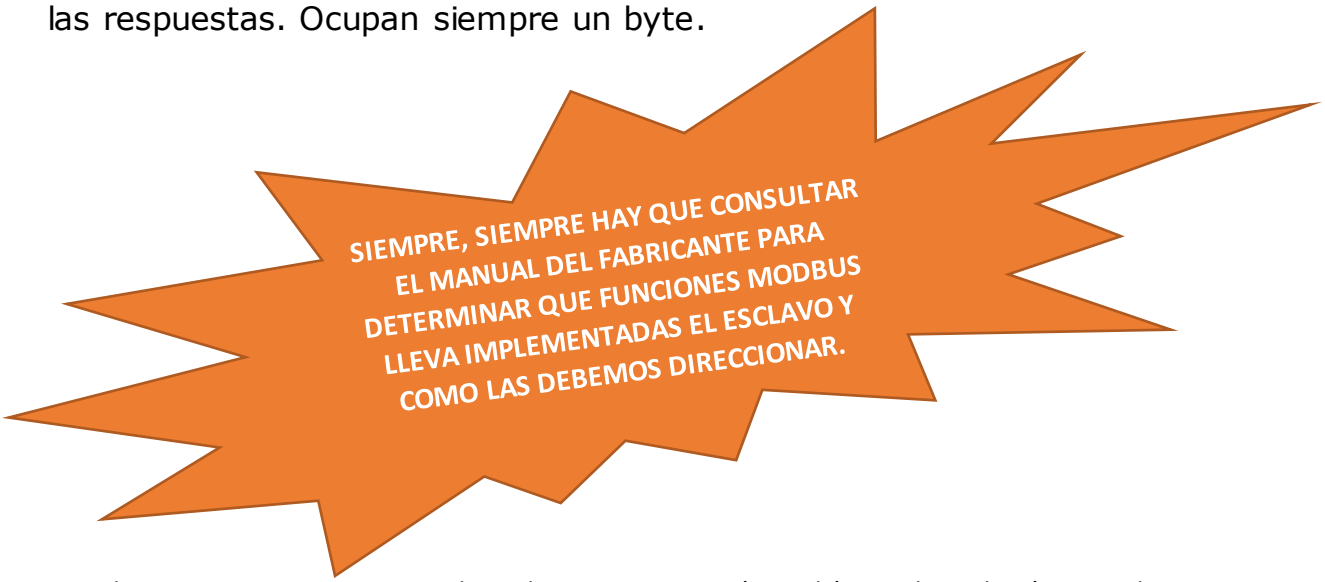
El campo de datos, en una trama de MODBUS, agrupa **direcciones**, **número de registros**, **contenidos** y **números de bytes**. Todos estos datos, se escriben siempre en hexadecimal.

En cuanto a las **direcciones**, hay que tener en cuenta, que en los casos en los que en el esclavo, se direccionen desde la 1, en la trama, deberemos restar 1, pues siempre se deben considerar direccionados desde 0.

Los **números de registros**, se escribirán usando 2 bytes. Pueden aparecer tanto en las órdenes enviadas como en las respuestas.

Los **contenidos**, es suficiente en escribirlos en hexadecimal para los registros y para los bits, hay que usar las constantes preestablecidas 0xFF00 (HIGH) y 0x0000 (LOW).

Los **números de bytes**, pueden aparecer en las órdenes o en las respuestas. Ocupan siempre un byte.



En la siguiente tabla, podemos ver como acostumbra MODBUS, nombrar a las diferentes zonas de memoria usadas por sus esclavos.

Tipo de objeto	Acceso	Tamaño
Discrete input (entrada digital)	lectura	1 bit
Coil (salida digital)	Lectura/escritura	1 bit
Input register (registro de entrada)	lectura	16 bits
Holding register (registro de salida)	Lectura/escritura	16 bits

4.4.- Comprobación de errores

Ya hemos comprobado en el punto 4. Formato de las tramas MODBUS, como el esclavo **modifica** el código de función en su respuesta, para **avisar** al Maestro de que algo **no ha salido bien**.

También vimos, en ese mismo punto, que después del código de función alterado, el esclavo envía **un byte** con un **código numérico**, donde **identifica el error en concreto**. Pues bien, a continuación podemos observar una **tabla** con los códigos numéricos de error y su significado. Esta tabla es de tremenda **ayuda** para localizar los **problemas** que puedan surgir en comunicaciones MODBUS.

Códigos de Error MODBUS		
Código	Nombre	Contenido
0x01	ILLEGAL FUNCTION	El código de función recibido en la consulta no es una acción permitida para el esclavo. Esto puede deberse a que el código de función solo se aplica a los dispositivos más nuevos y no se implementó en la unidad seleccionada. También podría indicar que el esclavo está en un estado incorrecto para procesar una solicitud de este tipo, por ejemplo, porque no está configurado y se le pide que devuelva valores de registro.
0x02	ILLEGAL DATA ADDRESS	La dirección de datos recibida en la consulta no es una dirección permitida para el esclavo. Más específicamente, la combinación de número de referencia y longitud de transferencia no es válida. Para un controlador con 100 registros, la PDU direcciona el primer registro como 0 y el último como 99. Si se envía una solicitud con una dirección de registro inicial de 96 y

		una cantidad de registros de 4, esta solicitud funcionará correctamente (dirección al menos) en los registros 96, 97, 98, 99. Si se envía una solicitud con una dirección de registro inicial de 96 y una cantidad de registros de 5, esta solicitud fallará con el código de excepción 0x02 "Illegal Data Address" ya que intenta operar en los registros 96, 97, 98, 99 y 100, y no hay registro con la dirección 100.
0x03	ILLEGAL DATA VALUE	Un valor contenido en el campo de datos de la consulta no es un valor permitido para el servidor. Esto indica un fallo en la estructura del resto de una solicitud compleja, como que la longitud implícita es incorrecta. Específicamente NO significa que un elemento de datos enviado para su almacenamiento en un registro tenga un valor fuera de las expectativas del programa de aplicación, ya que el protocolo MODBUS desconoce la importancia de cualquier valor particular de cualquier registro en particular.
0x04	SERVER DEVICE FAILLURE	Se produjo un error irrecuperable mientras el servidor intentaba realizar la acción solicitada.
0x05	ACKNOWLEDGE	Uso especializado junto con comandos de programación. El servidor ha aceptado la solicitud y la está

		procesando, pero se necesitará un período de tiempo prolongado para hacerlo. Esta respuesta se devuelve para evitar que se produzca un error de tiempo de espera en el cliente. A continuación, el cliente puede emitir un mensaje de Programa de encuesta completo para determinar si se completó el procesamiento.
0x06	SERVER DEVICE BUSSY	Uso especializado junto con comandos de programación. El servidor está procesando un comando de programa de larga duración. El cliente debe retransmitir el mensaje más tarde cuando el servidor esté libre.
0x08	MEMORY PARITY ERROR	Uso especializado junto con los códigos de función 20 y 21 y el tipo de referencia 6, para indicar que el área de archivo extendida no pasó una verificación de coherencia. El servidor intentó leer el archivo de registro, pero detectó un error de paridad en la memoria. El cliente puede volver a intentar la solicitud, pero es posible que se requiera servicio en el dispositivo del servidor.
0x0A	GATEWAY PATH UNAVAILABLE	El uso especializado junto con las puertas de enlace indica que la puerta de enlace no pudo asignar una ruta de comunicación interna desde el puerto de entrada al puerto de salida para procesar la solicitud.

		Por lo general, significa que la puerta de enlace está mal configurada o sobrecargada.
0x0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	El uso especializado junto con pasarelas indica que no se obtuvo respuesta del dispositivo de destino. Por lo general, significa que el dispositivo no está presente en la red.

Links interesantes

https://www.modbus.org/

Web oficial de la corporación MODBUS. En el apartado *Technical Resources*, puedes encontrar las especificaciones oficiales de Modbus.

https://www.modbustools.com

Estupenda página para descargar tanto información como un simulador de MODBUS, Que junto con el emulador de puertos virtuales que vimos en el tema 5 de Comunicación serie, te pueden ayudar a estudiar las tramas, aunque no tengas ningún dispositivo en casa.

Videos explicativos

https://www.youtube.com/watch?v=PIjK2flfqYQ

https://www.youtube.com/watch?v=RgTm9g9WJgI&list=PLrRS0TuMUqyaXtLOo47N2f7DnXwUZJQS8&index=7&t=474s

https://www.youtube.com/watch?v=GlblujGwX9c&list=PLrRS0TuMUqyaXtLOo47N2f7DnXwUZJQS8&index=6

https://www.youtube.com/watch?v=xD1ts98eUZI&list=PLrRS0TuMUqyaXtLOo47N2f7DnXwUZJQS8&index=5&t=213s

https://www.youtube.com/watch?v=dlg3Pl2RmHg&list=PLrRS0TuMUqyaXtLOo47N2f7DnXwUZJQS8&index=4&t=11s

https://www.youtube.com/watch?v=lJzeM0-cexI&list=PLrRS0TuMUqyaXtLOo47N2f7DnXwUZJQS8&index=3&t=601s