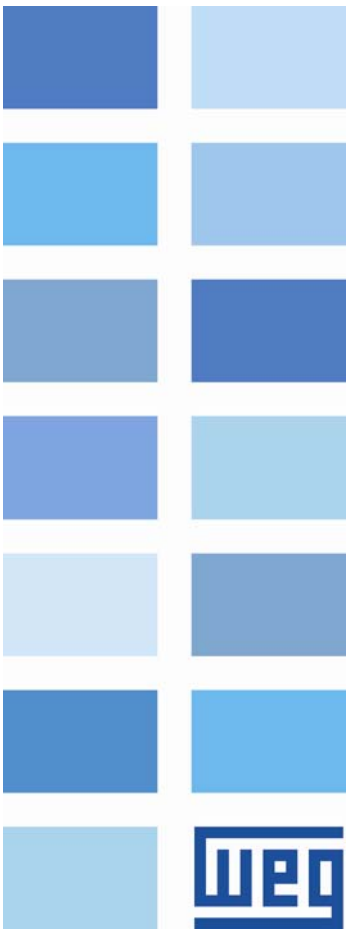
Modbus RTU

SCA06

Manual del Usuario



Manual del Usuario Modbus RTU

Serie: SCA06

Idioma: Español

N ° del Documento: 10001625786 / 00

Fecha de la Publicación: 12/2012

CONTENIDOS

A RESPECTO DEL MANUAL

Este manual provee la descripción necesaria para la operación del servoconvertidor SCA06 utilizando el protocolo Modbus. Este manual debe ser utilizado en conjunto con el manual del usuario del SCA06.

ABREVIACIONES Y DEFINICIONES

ASCII	American Standard Code for Information Interchange
CRC	Cycling Redundancy Check
EIA	Electronic Industries Alliance
TIA	Telecommunications Industry Association
RTU	Remote Terminal Unit

REPRESENTACIÓN NUMÉRICA

Números decimales son representados a través de dígitos sin sufijo. Números hexadecimales son representados con la letra 'h' luego del número. Números binarios son representados con la letra 'b' luego del número.

DOCUMENTOS

El protocolo Modbus RTU fue desarrollado con base en las siguientes especificaciones y documentos:

Documento	Versión	Fuente
MODBUS Application Protocol Specification, December 28th 2006.	V1.1b	MODBUS.ORG
MODBUS Protocol Reference Guide, June 1996.	Rev. J	MODICON
MODBUS over Serial Line, December 20th 2006.	V1.02	MODBUS.ORG

Para obtener esta documentación, de debe consultar la MODBUS.ORG, que actualmente es la organización que mantiene, promociona y actualiza las informaciones relativas a la red Modbus.

1 INTRODUCCIÓN A LA COMUNICACIÓN SERIAL

En una interfaz serial los bits de datos son enviados de modo secuencial a través de un canal de comunicación o bus. Diversas tecnologías utilizan comunicación serial para la transferencia de datos, incluyendo las interfaces RS232 y RS485.

Las normas que especifican los padrones RS232 y RS485, sin embargo, no especifican el formato ni la secuencia de caracteres para la transmisión y recepción de datos. En este sentido, además de la interface, es necesario identificar también el protocolo utilizado para la comunicación. Entre los diversos protocolos existentes, un protocolo muy utilizado en la industria es el protocolo Modbus RTU.

A seguir serán presentadas las características de las interfaces serial RS232 y RS485 disponible para el producto, así como los protocolos para la utilización de estas interfaces.

2 DESCRIPCIÓN DE LAS INTERFACES

Para posibilitar la comunicación serial en el servoconvertidor SCA06, es necesario utilizar el módulo de expansión y comunicación ECO1 descrito a seguir. Informaciones sobre la instalación de este módulo pueden ser obtenidas en la guía que acompaña al accesorio.

2.1 MÓDULO DE EXPANSIÓN Y COMUNICACIÓN RS232 Y RS485 ECO1

- Item WEG: 11330271.
- Compuesto por el módulo de comunicación ECO1 (figura al lado) y una guía de montaje.
- Interfaz aislada galvánicamente y con señal diferencial, otorgando mayor robustez contra interferencia electromagnética.
- Las señales RS232 y RS485 son canales independientes, pudiendo ser utilizados simultáneamente.

2.2 RS232

2.2.1 Indicaciones

El led LA121 indica (enciende) cuando hay transmisión de datos a través de la comunicación RS232.

2.2.2 Conexión con la Red RS232

- Las señales RX y TX del convertidor deben ser conectadas respectivamente a las señales TX y RX del maestro, además de la conexión de la sinal de referencia (GND).
- La interfaz RS232 es muy susceptible a interferencias. Por este motivo, el cable utilizado para comunicación debe ser lo más corto posible – siempre menor que 10 metros – y debe ser colocado separado del cableado de potencia que alimenta al convertidor y al motor.

2.2.3 Cables para Conexión en RS232

En caso que sea deseado, están disponibles los ítems de los siguientes cables para conexión en RS232 entre el servoconvertidor y un maestro de red, como un PC:

Tabla 2.1: *Cables RS232*

Cable	Item
Cable RS232 blindado con conectores DB9 Longitud: 3 metros	10050328
Cable RS232 blindado con conectores DB9 hembra Longitud: 10 metros	10191117

2.2.4 Sujeción del Conector

La conexión para la interfaz RS232 está disponible a través de los conectores XA121 y XA122 utilizando la siguiente sujeción:

Tabla 2.2: *Sujeción del conector para RS232 XA121*

Perno	Función
1	Tierra
2	RX_232
3	TX_232
4	Reservado[1]
5	GND
6	Reservado
7	232 RTS
8	A
9	B

Tabla 2.3: *Sujeción del conector para RS232 XA122*

Perno	Función
1	NC
2	RX_232
3	TX_232
4	Reservado
5	GND
6	NC
7	232 RTS
8	NC
9	NC
Carcasa	Tierra

2.3 RS485

2.3.1 Indicaciones

El led LA122 indica (enciende) cuando hay transmisión de datos a través de la comunicación RS485.

2.3.2 Características de la interfaz RS485

- La interfaz sigue el estándar EIA/TIA-485.
- Puede operar como esclavo de la red Modbus RTU.
- Posibilita comunicación utilizando tasas de 9600 hasta 57600 Kbit/s.
- Interfaz aislada galvánicamente y con señal diferencial, otorgando mayor robustez contra interferencia electromagnética.
- Permite la conexión de hasta 32 dispositivos en el mismo segmento. Una cantidad mayor de dispositivos puede ser conectada con el uso de repetidores.[2]
- Longitud máxima del embarrado de 1000 metros.

2.3.3 Sujeción del Conector

La conexión para la interfaz RS485 está disponible a través del conector XC1 utilizando la siguiente sujeción:

Tabla 2.4: *Sujeción del conector para RS485 XA121*

Perno	Función
1	Tierra
2	RX_232
3	TX_232
4	Reservado
5	GND
6	Reservado
7	232 RTS
8	A
9	B

[1] No conecte los pernos reservados
[2] El número límite de equipos que pueden ser conectados en la red también depende del protocolo utilizado.

Tabla 2.5: *Sujeción del conector para RS485 XA123*

Perno	Función
1	NC
2	NC
3	NC
4	NC
5	GND
6	Reservado
7	NC
8	A (fecha -)
9	B (fecha +)
Carcasa	Tierra

2.3.4 Resistor de terminación

Para cada segmento de la red RS485, es necesario habilitar un resistor de terminación en los puntos extremos del embarrado principal. El accesorio ECO1 posee dos llaves DIP Switch que pueden ser activadas (colocando ambas llaves SA121 en la posición ON) para habilitar el resistor de terminación.

Figura 2.1: *Resistor de terminación*

2.3.5 Conexión con la Red RS485

Deben ser observados los siguientes puntos, para la conexión del servoconvertidor SCA06 utilizando la interfaz RS485:

- Es recomendado el uso de un cable con par trenzado blindado.
- Se recomienda también que el cable posea un alambre más para conexión de la señal de referencia (GND). En caso que el cable no posea alambre adicional, se debe dejar la señal GND desconectada.
- El pasaje del cable deve ser realizado separadamente (y si es posible distante) de los cables para alimentación de potencia.
- Todos los dispositivos de la red deben estar debidamente puestos a tierra, preferentemente en la misma conexión con la tierra. El blindaje del cable también debe ser puesto a tierra.
- Habilite los resistores de terminación solamente en dos puntos, en los extremos del embarrado principal, aunque existan derivaciones a partir del embarrado.

3 PARAMETRIZACIÓN

A seguir es presentado solo os parámetros del servoconvertidor SCA06 que poseen relación con la comunicación Modbus RTU.

3.1 SÍMBOLOS PARA DESCRIPCIÓN DE LAS PROPIEDADES

RW	Parámetro de lectura y escritura
AC	Parámetro visible en la HMI solamente cuando el accesorio correspondiente está conectado

P00650 – DIRECCIÓN DEL SERVOCONVERTIDOR EN LA COMUNICACIÓN SERIAL 1 – RS232

P00656 – DIRECCIÓN DEL SERVOCONVERTIDOR EN LA COMUNICACIÓN SERIAL 2 – RS485

Rango de Valores:	1 a 247	**Padrón:** 1
Propiedades:	RW, AC	

Descripción:
Permite programar la dirección utilizada para comunicación serial del equipo. Es necesario que cada equipo de la red posea una dirección diferente de las demás. Las direcciones válidas para este parámetro dependen del protocolo programado en el servoconvertidor.

- P00654/P00660 = 1 (WEGTP) → direcciones válidas: 1 a 30.
- P00654/P00660 = 2 (Modbus RTU) → direcciones válidas: 1 a 247.

P00652 – BIT RATE SERIAL 1 – RS232

P00658 – BIT RATE SERIAL 2 – RS485

Rango de Valores:	0 = 4800 bits/s 1 = 9600 bits/s 2 = 14400 bits/s 3 = 19200 bits/s 4 = 24000 bits/s 5 = 28800 bits/s 6 = 33600 bits/s 7 = 38400 bits/s 8 = 43200 bits/s 9 = 48000 bits/s 10 = 52800 bits/s 11 = 57600 bits/s	**Padrón:** 1
Propiedades:	RW, AC	

Descripción:
Permite programar el valor deseado para la tasa de comunicación de la interfaz serial, en bits por segundo. Esta tasa debe ser la misma para todos los equipos conectados en la red.

P00653 – CONFIGURACIÓN SERIAL 1 – RS232

P00659 – CONFIGURACIÓN SERIAL 2 – RS485

Rango de Valores:	0 = 8 bits de datos, sin paridad, 1 stop bit 1 = 8 bits de datos, paridad par, 1 stop bit 2 = 8 bits de datos, paridad impar, 1 stop bit 3 = 8 bits de datos, sin paridad, 2 stop bits 4 = 8 bits de datos, paridad par, 2 stop bits 5 = 8 bits de datos, paridad impar, 2 stop bits 6 = 7 bits de datos, sin paridad, 1 stop bit 7 = 7 bits de datos, paridad par, 1 stop bit 8 = 7 bits de datos, paridad impar, 1 stop bit 9 = 7 bits de datos, sin paridad, 2 stop bits 10 = 7 bits de datos, paridad par, 2 stop bits 11 = 7 bits de datos, paridad impar, 2 stop bits	**Padrón:** 3
Propiedades:	RW, AC	

Descripción:
Permite la configuración del número de bits de datos, paridad y *stop* bits en los bytes de la interfaz serial. Esta configuración debe ser la misma para todos los equipos conectados en la red.

P00654 – SELECCIONA PROTOCOLO SERIAL 1 – RS232

P00660 – SELECCIONA PROTOCOLO SERIAL 2 – RS485

Rango de Valores:	1 = WEGTP 2 = Modbus RTU	**Padrón:** 1
Propiedades:	RW	

Descripción:
Permite seleccionar el protocolo deseado para la interfaz serial. La descripción detallada del protocolo es realizada en el ítem 4 de este manual.

P00662 – ACCIÓN PARA ERROR DE COMUNICACIÓN

Rango de Valores:	0 = Inactivo 1 = Causa Falla 2 = Causa alarma y Stop 3 = Causa alarme y deshabilita drive	**Padrón:** 0
Propiedades:	RW	

Descripción:
Este parámetro permite seleccionar qué acción debe ser ejecutada por el equipo, en caso que sea controlado vía red y sea detectado un error de comunicación.

Tabla 3.1: *Opciones para el parámetro P00662*

Opción	Descripción
0 = Solamente indica alarma	Sólo muestra el código de alarma en la HMI del servoconvertidor.
1 = Causa Falla	Causa falla y el servoconvertidor sólo vuelve a operar en caso de que sea hecho reset de fallas.
2 = Indica alarma y ejecuta STOP	Será hecha la indicación de alarma junto con la ejecución del comando STOP. Para que el servo salga de esta condición, será necesario realizar el reset de fallas o deshabilitar el drive.
3 = Indica alarma y deshabilita drive	Será hecha la indicación de alarma junto con la ejecución del comando “deshabilita”.

Son considerados errores de comunicación los siguientes eventos:

Comunicación Serial (RS232/RS485):
- Alarma A00128/Falla F00028: *timeout* de la interfaz serial.

P00663 – WATCHDOG SERIAL

Rango de Valores:	0.0 a 999.0s	**Padrón:** 0.0
Propiedades:	RW	

Descripción:
Permite programar un tiempo para la detección de error de comunicación vía interfaz serial. En caso que el convertidor quede sin recibir telegramas válidos por un tiempo mayor que el programado en este parámetro, será considerado que ocurrió un error de comunicación, mostrando la alarme A00128 en la HMI (o falla F00028, dependiendo de la programación hecha en el P00662) y la acción programada en el P00662 será ejecutada.

Luego de energizado, el convertidor comenzará a contar este tiempo, a partir del primer telegrama válido recibido. El valor 0,0 deshabilita esta función.

P00664 – GUARDA PARÁMETROS EN MEMORIA NO-VOLÁTIL

Rango de Valores:	0 = No guarda parámetro en la memoria no-volátil 1 = Guarda parámetro en la memória volátil	**Padrón:** 1
Propiedades:	RW	

Descripción:
Permite seleccionar si la escritura de parámetros vía serial debe, o no, guardar el contenido de los parámetros en memoria no-volátil (EEPROM). Cuando es utilizado el protocolo Modbus, es apenas ese parámetro que determina si los parámetros escritos vía serial serán, o no, guardados en la memoria no- volátil. No obstante, cuando es utilizado el protocolo WEGTP, se debe observar que en el byte de código del telegrama conste la información sobre guardar, o no, el parámetro en la EEPROM. Para que, vía WEGTP los mismos sean salvos en la memoria no-volátil. Es necesario que las dos informaciones, el byte de código del telegrama y el parámetro P00664, sean verdaderas.

¡NOTA!
Este tipo de memoria posee un número límite de escrituras (100.000 veces). Dependiendo de la aplicación, este límite puede ser sobrepasado, en caso de que algunos parámetros sean escritos cíclicamente vía serial (referencia de velocidad, torque, etc.). En estos casos, puede ser deseado que, durante la operación del servoconvertidor, la escritura vía serial no guarde el contenido de los parámetros en la memoria no-volátil, para que no sobrepase el límite de escrituras en el servoconvertidor.

¡NOTA!
Este parámetro no se aplica cuando la escritura es hecha utilizando la interfaz USB.

P00667 – GUARDA EN MARCADORES

Rango de Valores:	0 = Lee y escribe normalmente el contenido en el parámetro correspondiente 1 = Lee y escribe contenido en marcadores de WORD volátil a partir del MW13000	**Padrón:** 0
Propiedades:	RW	

Descripción:
Propiedad verificada cuando el parámetro es escrito y leído vía serial. Selecciona si desea que contenido a ser escrito/leído sea guardado en parámetro o en marcador de Word volátil.

¡NOTA!
Siendo este parámetro P00667 = 1, al escribir vía serial en el parámetro P00105 = 30, el contenido del parámetro será almacenado en el marcador de Word 13105 (MWinicial + Numero_par => 13000 + 105). Por tanto, MW13105 = 30.

Observación: Una vez que P00667 = 1, el mismo no podrá ser alterado vía serial. Ya que en el intento de escribir en el parámetro P00667, estará escribiendo en el marcador de Word P13667.

4 PROTOCOLO MODBUS RTU

El protocolo Modbus fue inicialmente desarrollado en 1979. Actualmente, es un protocolo abierto ampliamente difundido, utilizado por varios fabricantes en diversos equipos.

4.1 MODOS DE TRANSMISIÓN

En la especificación del protocolo están definidos dos modos de transmisión: ASCII y RTU. Los modos definen la forma como son transmitidos los bytes del mensaje No es posible utilizar los dos modos de transmisión en la misma red.

El servoconvertidor SCA06 utiliza solamente el modo RTU para la transmisión de telegramas. Los bytes son transmitidos en el formato hexadecimal, y su configuración depende de la programación hecha a través del do P00659 o P00663.

4.2 ESTRUCTURA DE LOS MENSAJES EN EL MODO RTU

La red Modbus RTU utiliza el sistema maestro-esclavo para el intercambio de mensajes. Permite hasta 247 esclavos, más solamente un maestro. Toda comunicación inicia con el maestro haciendo una solicitación a un esclavo, y este contesta al maestro el que fue solicitado. En ambos los telegramas (pregunta y respuesta), la estructura utilizada es la misma: Dirección, Código de la Función, Datos y CRC. Luego el campo de datos podrá tener tamaño variable, dependiendo del que está siendo solicitado.

Maestro (telegrama de solicitación):

Dirección (1 byte)	Función (1 byte)	Datos de la solicitación (n bytes)	CRC (2 bytes)

Esclavo (telegrama de respuesta):

Dirección (1 byte)	Función (1 byte)	Datos de la respuesta (n bytes)	CRC (2 bytes)

4.2.1 Dirección

El maestro inicia la comunicación enviando un byte con la dirección del esclavo para el cual se destina el mensaje. Al enviar la respuesta, el esclavo también inicia el telegrama con el su propia dirección. El maestro también puede enviar un mensaje destinado a la dirección “0” (cero), lo que significa que el mensaje es destinada a todos los esclavos de la red (broadcast). En este caso, ninguno esclavo irá contestar al maestro.

4.2.2 Código de la Función

Este campo también contiene un único byte, donde el maestro especifica el tipo de servicio o función solicitada al esclavo (lectura, escrita, etc.). De acuerdo con el protocolo, cada función es utilizada para acceder un tipo específico de dato.

Para la lista de funciones disponibles para acceso a los datos, consulte el ítem 5.

4.2.3 Campo de Datos

Campo con tamaño variable. El formato y el contenido de este campo dependen de la función utilizada y de los valores transmitidos. Este campo está descrito juntamente con la descripción de las funciones (consultar ítem 5).

4.2.4 CRC

La última parte del telegrama es el campo para el chequeo de errores de transmisión. El método utilizado es el CRC-16 (Cycling Redundancy Check). Este campo es formado por dos bytes, donde primero es transmitido el byte menos significativo (CRC-), y después el más significativo (CRC+). La forma de cálculo del CRC es descrita en la especificación del protocolo, sin embargo informaciones para su implementación también son suministradas en el Apéndice B.

4.2.5 Tiempo entre Mensajes

En el modo RTU no existe un carácter específico que indique el inicio o el fin de un telegrama. La indicación de cuando un nuevo mensaje empieza o cuando elle termina es hecha por la ausencia de transmisión de datos en la red, por un tiempo mínimo de 3,5 veces el tiempo de transmisión de un byte de datos (11 bits). Siendo así, caso un telegrama tenga iniciado luego de transcurrido este tiempo mínimo, los elementos de la red irán asumir que el primero carácter recibido representa el inicio de un nuevo telegrama. Y de la misma forma, los elementos de la red irán asumir que el telegrama ha llegado al fin cuando, recibidos los bytes del telegrama, este tiempo transcurrir nuevamente.

Si durante la transmisión de un telegrama, el tempo entre los bytes fue mayor que este tempo mínimo, el telegrama será considerado inválido, pues el servoconvertidor irá rechazar los bytes ya recibidos y montará un nuevo telegrama con los bytes que estuvieren siendo transmitidos.

Para tasas de comunicación superiores a 19200 bits/s, los tiempos utilizados son los mismos que para esta tasa. La tabla a seguir preséntanos los tiempos para distintas tasas de comunicación:

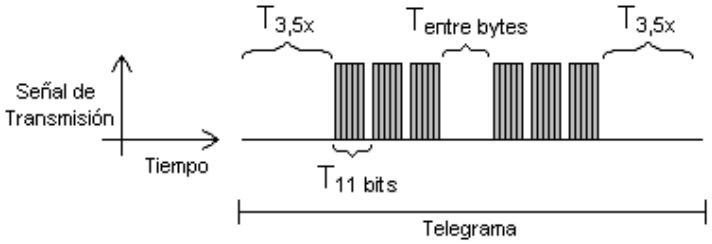


Tabla 4.1: *Tasas de comunicación y tiempos involucrados en la transmisión de telegramas*

Tasa de Comunicación	T11 bits	T3,5x
1200 bits/s	9,167 ms	32,083 ms
2400 bits/s	4,583 ms	16,042 ms
4800 bits/s	2,292 ms	8,021 ms
9600 bits/s	1,146 ms	4,010 ms
19200 bits/s	573 µs	2,005 ms
38400 bits/s	573 µs	2,005 ms
57600 bits/s	573 µs	2,005 ms

- $T_{11\ bits}$ = Tiempo para transmitir una palabra del telegrama.
- $T_{entre\ bytes}$ = Tiempo entre bytes.
- $T_{3,5x}$ = Intervalo mínimo para indicar el inicio y el fin de telegrama (3,5 x T11bits).

5 OPERACIÓN EN LA RED MODBUS RTU – MODO ESCLAVO

El servoconvertidor SCA06 posee las siguientes características cuando operado como esclavo en red Modbus RTU:

- Conexión de la red vía interfaz serial RS485.
- La dirección, tasa de comunicación y formato de los bytes definidos a través de parámetros.
- Permite la parametrización y control del servoconvertidor a través del acceso a parámetros.

5.1 FUNCIONES DISPONIBLES Y TIEMPOS DE RESPUESTA

En la especificación del protocolo Modbus RTU son definidas funciones utilizadas para acceder diferentes tipos de datos. En el SCA06, los parámetros fueran definidos como siendo registradores del tipo holding. Para acceder estos registradores, fueran colocados disponibles los siguientes servicios (o funciones):

- Read Coils[3]
  Descripción: lectura de bloque bits del tipo coil.
  Código de la función: 01.

- Read Discrete Inputs[3]
  Descripción: lectura de bloque bits del tipo entradas discretas.
  Código de la función: 02.

- Read Holding Registers
  Descripción: lectura de bloque de registradores del tipo holding.
  Código de la función: 03.

- Read Input Registers[3]
  Descripción: lectura de bloque de registradores del tipo input.
  Código de la función: 04.

- Write Single Coil[3]
  Descripción: escrita en un único bit del tipo coil.
  Código de la función: 05.

- Write Single Register
  Descripción: escrita en un único registrador del tipo holding.
  Código de la función: 06.

- Write Multiple Coils[3]
  Descripción: escrita en bloque de bit del tipo coil.
  Código de la función: 15.

- Write Multiple Registers
  Descripción: escrita en bloque de registradores del tipo holding.
  Código de la función: 16.

- Read Device Identification
  Descripción: identificación del modelo del equipo.
  Código de la función: 43.

El tiempo de respuesta, fin de la transmisión del maestro hasta el inicio de la respuesta del esclavo, varia de 2 a 10 ms, para cualquier una de las funciones arriba.

5.2 MAPA DE MEMORIA

La comunicación Modbus para el servoconvertidor SCA06 es basada en la lectura/escritura de parámetros del equipo. Toda la lista de parámetros del equipo es disponibilizada como registradores de 16 bits del tipo holding. El direccionamiento de los datos es realizado con offset igual a cero, lo que significa que el número del

[3] Funciones utilizadas para acceder a los datos utilizados por la función SoftPLC.

parámetro equivale al número del registrador. La tabla a seguir ilustra el direccionamiento de los parámetros, que pueden accesarse como registradores del tipo holding.

Tabla 5.1: *Mapa de memoria para la interfaz Modbus RTU*

Número del Parámetro	Dirección del dato Modbus	
	Decimal	Hexadecimal
P0000	0	0000h
P0001	1	0001h
⋮	⋮	⋮
P0100	100	0064h
⋮	⋮	⋮

Para la operación del equipo, es necesario conocer la lista de parámetros del producto. De esta forma se pueden identificar cuales datos son necesarios para monitoreo de los estados y control de las funciones. Dentro de los principales parámetros se pueden citar:

Monitoreo (lectura):
- P0680 (holding register 680): Palabra de estado
- P0681 (holding register 681): Velocidad del motor

Comando (escritura):
- P0682 (holding register 682): Palabra de comando
- P0683 (holding register 683): Referencia de velocidad

Consulte el manual de programación para la lista completa de parámetros del equipo.

¡NOTA!
- Todos los parámetros son tratados como registradores del tipo holding. Dependiendo del maestro utilizado, estos registradores son referenciados a partir del enderezo base 40000 o 4x. En este caso, la dirección para un parámetro que debe ser programado en el maestro es la dirección presentada en la tabla arriba adicionado a la dirección base. Consulte la documentación del maestro para saber como acceder registradores del tipo holding.
- Se debe observar que parámetros con la propiedad de solamente lectura apenas pueden ser leídos del equipo, mientras que demás parámetros pueden leerse y escribirse a través de la red.
- Además de los parámetros, otros tipos de datos como marcadores de bit, word o float también pueden ser accedidos utilizando la interfaz Modbus RTU. Estos marcadores son utilizados principalmente por la función SoftPLC disponible para el SCA06. Para la descripción de estos marcadores, bien como la dirección para accederlos vía Modbus, se debe consultar el Manual de la SoftPLC.

6 DESCRIPCIÓN DETALLADA DE LAS FUNCIONES

En este ítem es hecha una descripción detallada de las funciones disponibles en el servoconvertidor SCA06 para comunicación Modbus RTU. Para la elaboración de los telegramas, es importante observar lo siguiente:

- Los valores son siempre transmitidos en hexadecimal.
- La dirección de un dato, el número de datos y el valor de los registradores son siempre representados en 16 bits. Por eso, es necesario transmitir estos campos utilizando dos bytes – superior (high) e inferior (low).
- Los telegramas, tanto para pregunta cuanto para respuesta, no pueden ultrapasar 64 bytes.
- Los valores transmitidos son siempre números enteros, independiente de poseyeren representación con casa decimal. De esta forma, el valor 9,5 seria transmitido como siendo 95 (5Fh) vía serial. Consulte la lista de parámetros del SCA06 para obtener la resolución utilizada para cada parámetro.

6.1 FUNCIÓN 03 – READ HOLDING REGISTER

Lee el contenido de un grupo de registradores, que necesariamente deben estar en secuencia numérica. Esta función posee la siguiente estructura para los telegramas de lectura y respuesta (cada campo representa un byte):

Solicitación (Maestro)	**Respuesta (Esclavo)**
Dirección del esclavo	Dirección del esclavo
Función	Función
Dirección del registrador inicial (byte high)	Campo Byte Count
Dirección del registrador inicial (byte low)	Dato 1 (high)
Número de registradores (byte high)	Dato 1 (low)
Número de registradores (byte low)	Dato 2 (high)
CRC-	Dato 2 (low)
CRC+	Etc...
	CRC-
	CRC+

Ejemplo: lectura de la velocidad del motor (P0002) y corriente del motor (P0003) del esclavo en la dirección 1 (suponiendo P0002 = 1000 rpm y P0003 = 3,5 A).

- Dirección: 1 = 01h (1 byte)
- Dirección del registrador inicial: 2 = 0002h (2 bytes)
- Valor del primer parámetro: 1000 = 03E8h (2 bytes)
- Valor del segundo parámetro: 35 = 0023h (2 bytes)

Solicitación (Maestro)		**Respuesta (Esclavo)**	
Campo	*Valor*	*Campo*	*Valor*
Dirección del esclavo	01h	Dirección del esclavo	01h
Función	03h	Función	03h
Registrador inicial (high)	00h	Byte Count	04h
Registrador inicial (low)	02h	P002 (high)	03h
No. de registradores (high)	00h	P002 (low)	E8h
No. de registradores (low)	02h	P003 (high)	00h
CRC-	65h	P003 (low)	23h
CRC+	CBh	CRC-	3Bh
		CRC+	9Ah

6.2 FUNCIÓN 06 – WRITE SINGLE REGISTER

Esta función es utilizada para escribir un valor para un único registrador. Posee la siguiente estructura (cada campo representa un byte):

Solicitación (Maestro)	**Respuesta (Esclavo)**
Dirección del esclavo	Dirección del esclavo
Función	Función
Dirección del registrador (byte high)	Dirección del registrador (byte high)
Dirección del registrador (byte low)	Dirección del registrador (byte low)
Valor para el registrador (byte high)	Valor para el registrador (byte high)
Valor para el registrador (byte low)	Valor para el registrador (byte low)
CRC-	CRC-
CRC+	CRC+

Ejemplo: escrita de la consigna de velocidad (P0121) en 2000 rpm para el esclavo en la dirección 3.
- Dirección: 3 = 03h (1 byte)
- Dirección del registrador inicial: 121 = 0079h (2 bytes)
- Valor para el parámetro: 07D0h (2 bytes)

Solicitación (Maestro)		**Respuesta (Esclavo)**	
Campo	*Valor*	*Campo*	*Valor*
Dirección del esclavo	03h	Dirección del esclavo	03h
Función	06h	Función	06h
Registrador (high)	00h	Registrador (high)	00h
Registrador (low)	79h	Registrador (low)	79h
Valor (high)	07h	Valor (high)	07h
Valor (low)	D0h	Valor (low)	D0h
CRC-	5Ah	CRC-	5Ah
CRC+	5Dh	CRC+	5Dh

Note que para esta función, la respuesta del esclavo es una copia idéntica de la solicitación hecha por el maestro.

6.3 FUNCIÓN 16 – WRITE MULTIPLE REGISTERS

Esta función permite escribir valores para un grupo de registradores, que deben estar en secuencia numérica. También puede ser usada para escribir un único registrador (cada campo representa un byte).

Solicitación (Maestro)	**Respuesta (Esclavo)**
Dirección del esclavo	Dirección del esclavo
Función	Función
Dirección del registrador inicial (byte high)	Dirección del registrador inicial (byte high)
Dirección del registrador inicial (byte low)	Dirección del registrador inicial (byte low)
Número de registradores (byte high)	Número de registradores (byte high)
Número de registradores (byte low)	Número de registradores (byte low)
Campo Byte Count (nº de bytes de datos)	CRC-
Dato 1 (high)	CRC+
Dato 1 (low)	
Dato 2 (high)	
Dato 2 (low)	
etc...	
CRC-	
CRC+	

Ejemplo: escritura de la función de las tres entradas digitales P0300, P0301 y P0302 igual a 4, 4 y 10, respectivamente, de un esclavo en la dirección 15.
- Dirección: 15 = 0Fh (1 byte)
- Dirección del registrador inicial: 300 = 012Ch (2 bytes)
- Valor para el primer parámetro: 4 = 0004h (2 bytes)
- Valor para el segundo parámetro: 4 = 0004h (2 bytes)
- Valor para el tercero parámetro: 10 = 000Ah (2 bytes)

Solicitación (Maestro)		**Respuesta (Esclavo)**	
Campo	*Valor*	*Campo*	*Valor*
Dirección del esclavo	0Fh	Dirección del esclavo	0Fh
Función	10h	Función	10h
Registrador inicial (high)	01h	Registrador inicial (high)	01h
Registrador inicial (low)	2Ch	Registrador inicial (low)	2Ch
No. de registradores (high)	00h	No. de registradores (high)	00h
No. de registradores (low)	03h	No. de registradores (low)	03h
Byte Count	06h	CRC-	41h
P0300 (high)	00h	CRC+	13h
P0300 (low)	04h		
P0301 (high)	00h		
P0301 (low)	04h		
P0302 (high)	00h		
P0302 (low)	0Ah		
CRC-	05h		
CRC+	A1h		

6.4 FUNCIÓN 43 – READ DEVICE IDENTIFICATION

Función auxiliar, que permite la lectura del fabricante, modelo y versión de firmware del producto. Posee la siguiente estructura:

Solicitación (Maestro)	Respuesta (Esclavo)
Dirección del esclavo	Dirección del esclavo
Función	Función
MEI Type	MEI Type
Código de lectura	Conformity Level
Número del objeto	More Follows
CRC-	Próximo objeto
CRC+	Número de objetos
	Código del primer objeto
	Tamaño del primer objeto
	Valor del primer objeto (n bytes)
	Código del segundo objeto
	Tamaño del segundo objeto
	Valor del segundo objeto (n bytes)
	etc...
	CRC-
	CRC+

Esta función permite la lectura de tres categorías de informaciones: Básica, Regular y Extendida, y cada categoría es formada por un grupo de objetos. Cada objeto es formado por una secuencia de caracteres ASCII. Para el SCA06, solo informaciones básicas están disponibles, formadas por tres objetos:

- Objeto 00h – VendorName: representa el nombre del fabricante del producto.
- Objeto 01h – ProductCode: Formado por el código del producto (SCA06).
- Objeto 02h – MajorMinorRevision: indica la versión de firmware del producto, en el formato 'VX.XX'.

El código de lectura indica cuales las categorías de informaciones son leídas, y si los objetos son accedidos en secuencia o individualmente. En el caso, el SCA06 soporta los códigos 01 (informaciones básicas en secuencia), y 04 (acceso individual a los objetos). Los demás campos son especificados por el protocolo y para el SCA06 poseen valores fijos.

Ejemplo: lectura de las informaciones básicas en secuencia, a partir del objeto 02h, de un esclavo en la dirección 1:

Solicitación (Maestro)		Respuesta (Esclavo)	
Campo	*Valor*	*Campo*	*Valor*
Dirección del esclavo	01h	Dirección del esclavo	01h
Función	2Bh	Función	2Bh
MEI Type	0Eh	MEI Type	0Eh
Código de leitura	01h	Código de lectura	01h
Número del objeto	02h	Conformity Level	81h
CRC-	70h	More Follows	00h
CRC+	77h	Próximo Objeto	00h
		Número de objetos	01h
		Código del objeto	02h
		Tamaño del objeto	05h
		Valor del objeto	'V1.00'
		CRC-	3Ch
		CRC+	53h

En este ejemplo, el valor de los objetos no fue representado en hexadecimal, más sí utilizando los caracteres ASCII correspondientes. Por ejemplo, para el objeto 02h, el valor 'V1.00' fue transmitido como siendo cinco caracteres ASCII, que en hexadecimal poseen los valores 56h ('V'), 31h ('1'), 2Eh ('.'), 30h ('0') y 30h ('0').

6.5 ERRORES DE COMUNICACIÓN

Errores de comunicación pueden ocurrir tanto en la transmisión de los telegramas cuanto en el contenido de los telegramas transmitidos. De acuerdo con el tipo de error, el SCA06 podrá o no enviar respuesta para el maestro.

Cuando el maestro envía un mensaje para un esclavo configurado en una determinada dirección de la red, este no irá contestar al maestro caso ocurra:

- Error en el bit de paridad.
- Error en el CRC.
- Timeout entre los bytes transmitidos (3,5 veces el tempo de transmisión de un byte).

En estos casos, el maestro deberá detectar la ocurrencia del error por el timeout en la espera de la contestación del esclavo. En el caso de una recepción con suceso, durante el tratamiento del telegrama, el esclavo puede detectar problemas y enviar un mensaje de error, indicando el tipo de problema encontrado:

- Función inválida (código del error = 1): la función solicitada no está implementada para el equipo.
- Dirección de dato inválido (código del error = 2): la dirección del dato no existe.
- Valor de dato inválido (código del error = 3): ocurre en las siguientes situaciones:
  - Valor está fuera del rango permitido.
  - Escrita en dato que no puede ser modificado (registrador solamente de lectura).

¡NOTA!
Es importante que sea posible identificar en el maestro cual el tipo de error ocurrido para que se pueda diagnosticar problemas durante la comunicación.

En el caso de la ocurrencia de algún de estos errores, el esclavo debe retornar un mensaje para el maestro que indica el tipo de error ocurrido. Los mensajes de error enviados por el esclavo poseen la siguiente estructura:

Solicitación (Maestro)	**Respuesta (Esclavo)**
Dirección del esclavo	Dirección del esclavo
Función	Función (con el bit más significativo en 1)
Datos	Código del error
CRC-	CRC-
CRC+	CRC+

Ejemplo 5: maestro solicita para el esclavo de la dirección 1 la escrita en el registrador 2900 (suponiendo registrador 2900 como siendo inexistente):

Solicitación (Maestro)		**Respuesta (Esclavo)**	
Campo	*Valor*	*Campo*	*Valor*
Dirección del esclavo	01h	Dirección del esclavo	01h
Función	06h	Función	86h
Registrador (high)	0Bh	Código del error	02h
Registrador (low)	54h	CRC-	C3h
Valor (high)	00h	CRC+	A1h
Valor (low)	00h		
CRC-	CAh		
CRC+	3Eh		

7 FALLAS Y ALARMAS RELACIONADAS CON LA COMUNICACIÓN MODBUS RTU

A00128/F00028 – TIMEOUT EN LA RECEPCIÓN DE TELEGRAMAS

Descripción:
Único alarma/falla relacionado con la comunicación serial. Indica que el producto ha parado de recibir telegramas seriales válidos por un período mayor del que el programado en el P00663.

Actuación:
El parámetro P00663 permite programar un tiempo dentro del cual el esclavo deberá recibir al menos un telegrama válido vía interface serial RS485 – con dirección y campo de chequeo de errores correctos – caso contrario será considerado que ha ocurrido algún problema en la comunicación serial. El conteo del tiempo es iniciado luego de la recepción del primer telegrama válido.

Después de identificado el timeout en la comunicación serial, será señalizada a través del HMI el mensaje de alarma A00128 – o falla F00028, dependiendo de la programación hecha en el parámetro P00662. En condición de alarma, caso la comunicación sea restablecida y nuevos telegramas válidos sean recibidos, la señalización de falla en el IHM desaparecerá automáticamente.

Posibles Causas/Corrección:
- Verificar factores que puedan provocar fallas en la comunicación (cables, instalación, puesta a tierra).
- Garantizar que el maestro envíe telegramas para el esclavo siempre en un tiempo menor que el programado en el P00663.
- Deshabilitar esta función en el P00663.

I. APÉNDICES

APÉNDICE A. TABLA ASCII

Tabla I.1: *Caracteres ASCII*

Dec	Hex	Chr		Dec	Hex	Chr	Dec	Hex	Chr	Dec	Hex	Chr
0	00	NUL	(Null char.)	32	20	Sp	64	40	@	96	60	`
1	01	SOH	(Start of Header)	33	21	!	65	41	A	97	61	a
2	02	STX	(Start of Text)	34	22	"	66	42	B	98	62	b
3	03	ETX	(End of Text)	35	23	#	67	43	C	99	63	c
4	04	EOT	(End of Transmission)	36	24	$	68	44	D	100	64	d
5	05	ENQ	(Enquiry)	37	25	%	69	45	E	101	65	e
6	06	ACK	(Acknowledgment)	38	26	&	70	46	F	102	66	f
7	07	BEL	(Bell)	39	27	'	71	47	G	103	67	g
8	08	BS	(Backspace)	40	28	(	72	48	H	104	68	h
9	09	HT	(Horizontal Tab)	41	29	)	73	49	I	105	69	i
10	0A	LF	(Line Feed)	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	(Vertical Tab)	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	(Form Feed)	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	(Carriage Return)	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	(Shift Out)	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	(Shift In)	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	(Data Link Escape)	48	30	0	80	50	P	112	70	p
17	11	DC1	(Device Control 1)	49	31	1	81	51	Q	113	71	q
18	12	DC2	(Device Control 2)	50	32	2	82	52	R	114	72	r
19	13	DC3	(Device Control 3)	51	33	3	83	53	S	115	73	s
20	14	DC4	(Device Control 4)	52	34	4	84	54	T	116	74	t
21	15	NAK	(Negative Acknowledgement)	53	35	5	85	55	U	117	75	u
22	16	SYN	(Synchronous Idle)	54	36	6	86	56	V	118	76	v
23	17	ETB	(End of Trans. Block)	55	37	7	87	57	W	119	77	w
24	18	CAN	(Cancel)	56	38	8	88	58	X	120	78	x
25	19	EM	(End of Medium)	57	39	9	89	59	Y	121	79	y
26	1A	SUB	(Substitute)	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	(Escape)	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	(File Separator)	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	(Group Separator)	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	(Record Separator)	62	3E	>	94	5E	^	126	7E	~
31	1F	US	(Unit Separator)	63	3F	?	95	5F	_	127	7F	DEL

APÉNDICE B. CÁLCULO DEL CRC UTILIZANDO TABLAS

A seguir es presentada una función, utilizando lenguaje de programación "C", que implementa el cálculo del CRC para el protocolo Modbus RTU. El cálculo utiliza dos tablas para suministrar valores precalculados de los desplazamientos necesarios para la realización del cálculo.

```
/* Table of CRC values for high–order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40 };


/* Table of CRC values for low–order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40 };


/* The function returns the CRC as a unsigned short type */
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg;                 /* message to calculate CRC upon    */
unsigned short usDataLen;               /* quantity of bytes in message     */
{
    unsigned char uchCRCHi = 0xFF;      /* high byte of CRC initialized     */
    unsigned char uchCRCLo = 0xFF;      /* low byte of CRC initialized      */
    unsigned uIndex;                    /* will index into CRC lookup table */
    while (usDataLen--)                 /* pass through message buffer      */
    {
        uIndex = uchCRCLo ^ *puchMsgg++; /* calculate the CRC               */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex};
        uchCRCHi = auchCRCLo[uIndex];
    }

    return (uchCRCHi << 8 | uchCRCLo);
}
```

WEG Equipamentos Elétricos S.A.
Jaraguá do Sul - SC - Brasil
Teléfono 55 (47) 3276-4000 - Fax 55 (47) 3276-4020
São Paulo - SP - Brasil
Teléfono 55 (11) 5053-2300 - Fax 55 (11) 5052-4212
automacao@weg.net
www.weg.net