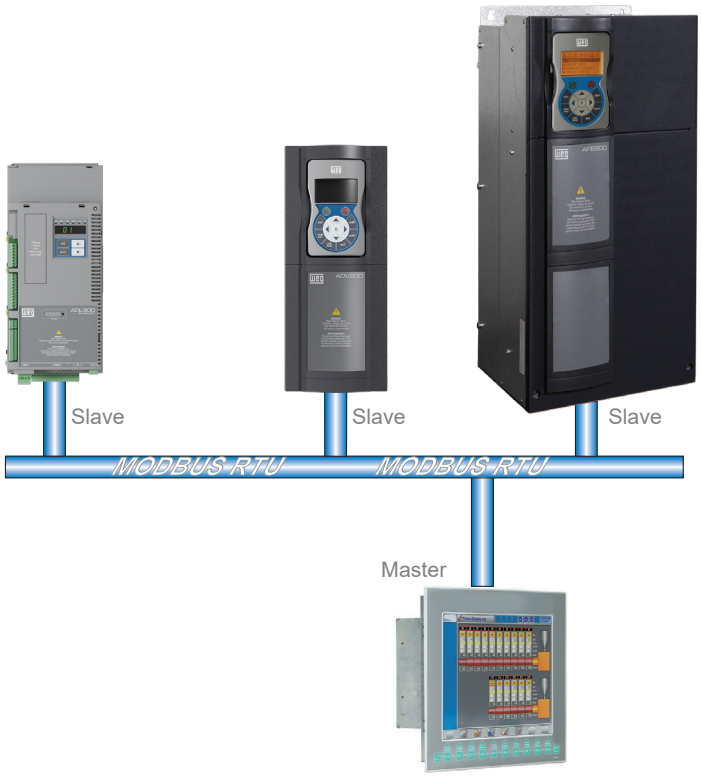
Modbus RTU Protocol
ADV200•ADL300•AFE200

Modbus RTU

Manuale d'uso
Instruction Manual

Language: English - Italiano



Sommario

1. Protocollo Modbus RTU

1.1 Introduzione

I parametri Drive, in funzione del Tipo (BIT, ENUM, FLOAT, INT16, eccetera), sono visti da Modbus come 16bit oppure 32 bit.

I parametri visti come 16 bit occcupano 1 registro Modbus.

I parametri visti come 32 bit occcupano 2 registri Modbus.

Numero registro Modbus = Indice parametro -1.

Esempi:

Parametro 600 "Dig ramp ref 1", Tipo INT16 => (1) Registro Modbus 599.

Parametro 3700 "Pad 1", Tipo INT32 => (2) Registri Modbus 3699 3700.

1.2 Il Protocollo MODBUS

Il protocollo MODBUS definisce il formato e la modalità di comunicazione tra un “master” che gestisce il sistema e uno o più “slave” che rispondono alle interrogazioni del master. Esso definisce come il master e gli slave stabiliscono e interrompono la comunicazione, come vengono scambiati i messaggi e come gli errori sono rilevati.

Si possono avere un master e fino a 255 slave su una linea comune; occorre notare che questo è un limite logico del protocollo, l’interfaccia fisica può peraltro limitare ulteriormente il numero di dispositivi; nell’implementazione attuale si prevede un **massimo di 32 slave connessi alla linea**.

Solo il master può iniziare una transazione. Una transazione può avere il formato domanda/risposta diretta ad un singolo slave o broadcast in cui il messaggio viene inviato a tutti gli slave sulla linea che non danno risposta. Una transazione è composta da una struttura (frame) singola domanda/singola risposta o una struttura singolo messaggio broadcast/nessuna risposta.

Alcune caratteristiche del protocollo non sono definite. Queste sono: standard di interfaccia, baud rate, parità, numero di stop bits. Il protocollo consente inoltre di scegliere tra due “modi” di comunicazione, ASCII e RTU (Remote Terminal Unit). Nel Drive viene implementato solo il modo RTU, in quanto più efficiente.

Il protocollo JBUS è funzionalmente identico al MODBUS e se ne differenzia per la diversa numerazione degli indirizzi: nel MODBUS questi partono da zero (0000 = 1° indirizzo) mentre nel JBUS partono da uno (0001 = 1° indirizzo) mantenendo questo scostamento per tutta la numerazione. Nel seguito, se non esplicitamente menzionato, pur facendo riferimento al MODBUS la descrizione si considera valida per entrambi i protocolli.

Esempio:

	Modbus	Jbus
PAR 600 Dig ramp ref 1, tipo INT16	1 Registro Modbus 599	1 Registro Jbus 600
PAR 3700 PAD 1, tipo INT32	2 Registri Modbus 3699 - 3700	2 Registri Jbus 3700 - 3701

1.3 Formato dei Messaggi

Per poter comunicare tra due dispositivi, il messaggio deve essere contenuto in un “involucro”. L’involucro lascia il trasmettitore attraverso una “porta” ed è “portato” lungo la linea fino ad una analoga “porta” sul ricevitore. MODBUS stabilisce il formato di questo involucro che, tanto per il master che per lo slave, comprende:

- L’indirizzo dello slave con cui il master ha stabilito la transazione (l’indirizzo 0 corrisponde ad un messaggio broadcast inviato a tutti i dispositivi slave).
- Il codice della funzione che deve essere o è stata eseguita.
- I dati che devono essere scambiati.
- Il controllo d’errore composto secondo l’algoritmo CRC16.

Se uno slave individua un errore nel messaggio ricevuto (di formato, di parità o nel CRC16) il messaggio viene

considerato non valido e scartato, uno slave che rilevi un errore nel messaggio quindi non esegue l'azione e non risponde alla domanda, così come se l'indirizzo non corrisponde ad uno slave in linea.

1.3.1 L'indirizzo

Come sopra menzionato, le transazioni MODBUS coinvolgono sempre il master, che gestisce la linea, ed uno slave per volta (tranne nel caso di messaggi broadcast). Per identificare il destinatario del messaggio viene trasmesso come primo carattere un byte che contiene l'indirizzo numerico dello slave selezionato. Ciascuno degli slave ha quindi assegnato un diverso numero di indirizzo che lo identifica univocamente. Gli indirizzi legali sono quelli da 1 a 255, mentre l'indirizzo 0, che non può essere assegnato ad uno slave, posto in testa al messaggio trasmesso dal master indica che questo è "broadcast", cioè diretto a tutti gli slave contemporaneamente. Possono essere trasmessi come broadcast solo messaggi che non richiedono risposta per espletare la loro funzione, quindi solo le assegnazioni.

1.3.2 Codice funzione

Il secondo carattere del messaggio identifica la funzione che deve essere eseguita nel messaggio trasmesso dal master, cui lo slave risponde a sua volta con lo stesso codice ad indicare che la funzione è stata eseguita.

È implementato un sottoinsieme delle funzioni MODBUS che comprende:

- 01 Read Coil Status (Non usato per i drive ADV-ADL-AFE)
- 02 Read Input Status (Non usato per i drive ADV-ADL-AFE)
- 03 Read Holding Registers
- 04 Read Input registers
- 05 Force Single Coil (Non usato per i drive ADV-ADL-AFE)
- 06 Preset Single register
- 07 Read Status
- 15 Force multiple Coils (Non usato per i drive ADV-ADL-AFE)
- 16 Preset Multiple Registers

Le funzioni 01 e 02 sono operativamente identiche e intercambiabili, così come le funzioni 03 e 04. Per una descrizione completa e dettagliata delle funzioni si rimanda al capitolo 3.

1.3.3 Il CRC16

Gli ultimi due caratteri del messaggio contengono il codice di ridondanza ciclica (Cyclic Redundancy Check) calcolato secondo l'algoritmo CRC16. Per il calcolo di questi due caratteri il messaggio (indirizzo, codice funzione e dati scartando i bit di start, stop e l'eventuale parità) viene considerato come un unico numero binario continuo di cui il bit più significativo (MSB) viene trasmesso prima. Il messaggio viene innanzitutto moltiplicato per x16 (spostato a sinistra di 16 bit) e poi diviso per $x^{16}+x^{15}+x^2+1$ espresso come numero binario (1100000000000101). Il quoziente intero viene poi scartato e il resto a 16 bit (inizializzato a FFFFh all'inizio per evitare il caso di un messaggio di soli zeri) viene aggiunto di seguito al messaggio trasmesso. Il messaggio risultante, quando diviso dallo slave ricevente per lo stesso polinomio ($x^{16}+x^{15}+x^2+1$) deve dare zero come resto se non sono intervenuti errori (lo slave ricalcola il CRC).

Di fatto, dato che il dispositivo che serializza i dati da trasmettere (UART) trasmette prima il bit meno significativo (LSB) anziché il MSB come dovrebbe essere per il calcolo del CRC, questo viene effettuato invertendo il polinomio. Inoltre, dato che il MSB del polinomio influenza solo il quoziente e non il resto, questo viene eliminato rendendolo quindi 1010000000000001.

La procedura passo-passo per il calcolo del CRC16 è la seguente:

1. Caricare un registro a 16 bit con FFFFh (tutti i bit a 1).
2. Fare l'OR esclusivo del primo carattere con il byte superiore del registro, porre il risultato nel registro.
3. Spostare il registro a destra di un bit.
4. Se il bit uscito a destra dal registro (flag) è un 1, fare l'OR esclusivo del polinomio generatore 1010000000000001 con il registro.
5. Ripetere per 8 volte i passi 3 e 4.
6. Fare l'OR esclusivo del carattere successivo con il byte superiore del registro, porre il risultato nel registro.
7. Ripetere i passi da 3 a 6 per tutti i caratteri del messaggio.
8. Il contenuto del registro a 16 bit è il codice di ridondanza CRC che deve essere aggiunto al messaggio.

1.3.4 Sincronizzazione dei messaggi

La sincronizzazione del messaggio tra trasmettitore e ricevitore viene ottenuta interponendo una pausa tra i messaggi pari ad almeno 3.5 volte il tempo di un carattere. Se il ricevitore non riceve per un tempo di 4 caratteri, ritiene completato il messaggio precedente e considera che il successivo byte ricevuto sarà il primo di un nuovo messaggio e quindi un indirizzo.

1.3.5 Impostazione linea seriale

La comunicazione prevede le seguenti impostazioni di default:

- 1 bit di start
- 8 bits di dati (RTU protocol)
- 1 bit di stop
- no parity

Le impostazioni sono selezionabili tra i seguenti valori:

N, 8, 1	(default)
N, 8, 2	
E, 8, 1	
O, 8, 1	

La comunicazione prevede il seguente baudrate:

Baudrate	Timeout byte-byte	
9600	4ms	
19200	2ms	
38400	1ms	(default)

1.4 Le funzioni Modbus per Drive

Viene riportata di seguito la descrizione dettagliata delle funzioni MODBUS implementate per i Drive. Tutti i valori riportati nelle tabelle sono in esadecimale.

1.4.1 Lettura Registri Uscite (03)

Questa funzione permette di leggere il valore di registri a 16 bit (word) contenenti parametri Drive. Il modo broadcast non è permesso.

Esempio: Parametro a 16 bit

Richiesta

Oltre all'indirizzo del Drive e al codice funzione (03) il messaggio contiene l'indirizzo di partenza dei registri (starting Address) espresso su due bytes e il numero dei registri da leggere anch'esso su due bytes. **Il numero massimo di registri che possono essere letti è 125**.

Esempio:

- Drive address 01 (01_{hex})
- Parametro 600 "Dig ramp ref 1" 600 -1 = 257_{hex}.

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	CRC HI	CRC LO
01	03	02	57	00	01	34	62

Risposta

Oltre all'indirizzo del Drive e al codice funzione (03), il messaggio comprende un carattere che contiene il numero di bytes di dati e i caratteri contenenti i dati. I registri richiedono due bytes, il primo dei quali contiene la parte più significativa.

Esempio:

Risposta alla richiesta sopra riportata. (Valore 100 = 64_{hex}).

ADDR	FUNC Byte	DATA word Count	DATA word HI	DATA word LO	CRC HI	CRC LO
01	03	02	00	64	B9	AF

Nota ! Nel caso si selezioni un range di registri che include dei registri riservati o mancanti, il valore di tali registri verrà posto a 0, vedere Appendice.

Lettura registri

Nel caso di parametri a 32 bits la lettura è realizzata tramite 2 registri Modbus.

Con il parametro 3808 "Serial swap data" è possibile configurare il contenuto dei due registri cioè nel primo registro la parte bassa e nel secondo registro la parte alta oppure viceversa.

		Registro 1	Registro 2
Serial swap data	OFF	L	H
Serial swap data	ON	H	L

Esempio Parametro Long:

Richiesta

- **Parametro 3808 "Serial swap data" = OFF**
- Drive address 01 (01_{hex})
- Parametro 3700 "Pad1" 3700 -1 = $E73_{hex}$ - Lettura a 2 registri.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	0E	73	00	02	37	38

Risposta

Valore 456 = 01C8hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	01	C8	00	00	7A	31
			Low part		High part			

- **Parametro 3808 "Serial swap data" = ON**
- Drive address 01 (01_{hex})
- Parametro 3700 "Pad1" 3700 -1 = $E73_{hex}$ - Lettura a 2 registri.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	0E	73	00	02	37	38

Risposta

Valore 456 = 01C8hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	00	00	01	C8	FA	35
			High part		Low part			

Oltre all'indirizzo del Drive e al codice funzione (03), il messaggio comprende un carattere che contiene il numero di bytes di dati e i caratteri contenenti i dati. I registri richiedono due bytes, il primo dei quali contiene la parte più significativa.

Esempio: Parametro Float

Richiesta

- **Parametro 3808 "Serial swap data" = OFF**
- Drive address 01 (01_{hex})
- Parametro 700 "Accelleration time" 700 -1 = $2BB_{hex}$ - Lettura a 2 registri.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	02	BB	00	02	B5	96

Risposta

Valore 1.0 = 3F80 0000 hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	00	00	3F	80	EA	63
			Low part		High part			

- **Parametro 3808 "Serial swap data" = ON**
- Drive address 01 (01_{hex})
- Parametro 700 "Accelleration time" 700 -1 = $2BB_{hex}$ - Lettura a 2 registri.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	02	BB	00	02	B5	96

Risposta

Valore 1.0 = 3F80 0000 hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	3F	80	00	00	F7	CF
			High part		Low part			

Oltre all'indirizzo del Drive e al codice funzione (03), il messaggio comprende un carattere che contiene il numero di bytes di dati e i caratteri contenenti i dati. I registri richiedono due bytes, il primo dei quali contiene la parte più significativa.

1.4.2 Lettura Registri Ingressi (04)

Questa funzione è operativamente identica alla precedente.

1.4.3 Preimpostazione Singoli Registri (06)

Questa funzione permette di impostare il valore di un singolo registro a 16 bit. Il modo broadcast è permesso.

Richiesta

Oltre all'indirizzo del Drive e al codice funzione (06) il messaggio contiene l'indirizzo del registro (parametro) espresso su due bytes e il valore che deve essere assegnato.

Esempio:

Scrittura parametro INT16 bit 600

- Drive address 01 (01_{hex})
- Registro 600 -1 (257_{hex})
- Valore 1234 ($4D2_{hex}$)

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	DATA word HI	DATA word LO	CRC HI	CRC LO
01	06	02	57	04	D2	BB	3F

Risposta

La risposta consiste nel ritrasmettere il messaggio ricevuto dopo che il registro è stato modificato.

Esempio:

Risposta alla richiesta sopra riportata.

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	DATA word HI	DATA word LO	CRC HI	CRC LO
01	06	02	57	04	D2	BB	3F

1.4.4 Lettura Stato (07)

Questa funzione permette di leggere lo stato di otto bit predeterminati con un messaggio compatto. Il modo broadcast non è permesso.

Richiesta

Il messaggio comprende solo l'indirizzo del Drive e il codice funzione (07).

Esempio:

- Drive address 01 (01_{hex})

ADDR	FUNC	CRC HI	CRC LO
01	07	41	E2

Risposta

Oltre all'indirizzo del Drive e al codice funzione (07) il messaggio comprende un carattere che contiene i bit di stato.

Esempio:

Risposta alla richiesta sopra riportata.

ADDR	FUNC	DATA status byte	CRC HI	CRC LO
01	07	01	E3	F0

Il significato del bit è il seguente:

Bit	Significato
0	Reserved
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	0 = Application; 1 = Boot

1.4.5 Preimpostazione Registri Multipli (16)

Questa funzione permette di impostare il valore di un blocco consecutivo di registri a 16 bit. Il modo broadcast è permesso.

Richiesta

Oltre all'indirizzo del Drive e al codice funzione (16) il messaggio contiene l'indirizzo di partenza dei registri da scrivere (starting Address), il numero di registri da scrivere, il numero di byte che contengono i dati e i caratteri di dati.

Esempio:

- Drive address 01 (01_{hex})
- Registro di partenza 3700 (3700 -1 = $E73_{hex}$) Parametro Pad1 - Tipo Long 32bit.
- Numero registri da scrivere 2 (02_{hex})
- Valore 16909069 (01020304_{hex})

ADDR	FUNC start	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	DATA word Count	DATA word HI	DATA word LO	DATA word HI	DATA word LO	CRC HI	CRC LO
01	10	0E	73	00	02	04	03	04	01	02	39	2A

Risposta

Oltre all'indirizzo del Drive e al codice funzione (16) il messaggio comprende l'indirizzo di partenza (starting Address).

Esempio:

Risposta alla richiesta sopra riportata.

ADDR	FUNC start	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	CRC HI	CRC LO
01	10	0E	73	00	02	B2	FB

Scrittura registri

Nel casi di parametri a 32 bits la scrittura è realizzata tramite due registri Modbus.

Con il parametro 3808 "Serial swap data" è possibile configurare il contenuto dei due registri cioè nel primo registro la parte bassa e nel secondo registro la parte alta oppure viceversa.

		Registro 1	Registro 2
Serial swap data	OFF	L	H
Serial swap data	ON	H	L

Esempio parametro Long

- **Parametro 3808 “Serial swap data” = OFF**
- Drive address = 1
- Parametro 3700 Pad 1 – Long 3700 - 1 E73Hex – Scrittura a 2 registri.
- Valore 456 = 01C8hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	0E	3E	00	02	04	01	C8	00	00	78	FC
							Low part		High part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	0E	3E	00	02	B2	FB

- **Parametro 3808 “Serial swap data” = ON**
- Drive address = 1
- Parametro 3700 Pad 1 – Long 3700 - 1 E73Hex – Scrittura a 2 registri.
- Valore 456 = 01C8hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	0E	3E	00	02	04	00	00	01	C8	F8	F8
							High part		Low part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	0E	3E	00	02	B2	FB

Esempio parametro Float

- **Parametro 3808 “Serial swap data” = OFF**
- Drive address = 1
- Parametro 700 Acceleration1 – Float 700 – 1 2BBHex – Scrittura a 2 registri.
- Valore 1.0 = 3F80 0000hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	02	BB	00	02	04	00	00	3F	80	B0	58
							Low part		High part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	02	BB	00	02	30	55

- **Parametro 3808 “Serial swap data” = ON**
- Drive address = 1

- Parametro 700 Acceleration1 – Float 700 – 1 2BBHex – Scrittura a 2 registri.
- Valore 1.0 = 3F80 0000hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	02	BB	00	02	04	3F	80	00	00	AD	F4
							High part		Low part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	02	BB	00	02	30	55

1.5 Gestione Errore

Nel MODBUS esistono due tipi di errori, gestiti in modo diverso: errori di trasmissione ed errori operativi. Gli errori di trasmissione sono errori che alterano il messaggio, nel suo formato, nella parità (se è usata), o nel CRC16. Il Drive che rileva errori di questo tipo nel messaggio lo considera non valido e non dà risposta. Qualora invece il messaggio sia corretto nella sua forma ma la funzione richiesta, per qualsiasi motivo, non sia eseguibile, si ha un errore operativo. A questo errore il Drive risponde con un messaggio di eccezione. Questo messaggio è composto dall'indirizzo del Drive, dal codice della funzione richiesta, da un codice d'errore e dal CRC. Per indicare che la risposta è la notifica di un errore il codice funzione viene ritornato con il bit più significativo a "1".

Esempio (parametro non esistente):

- Drive address 01 (01_{hex})
- Registro 601 (601 -1 = 258_{hex})

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	CRC HI	CRC LO
01	03	02	58	00	01	04	61

Risposta

La richiesta chiede il contenuto del Registro 601, che non esiste nel Drive slave. Questi risponde con il codice d'errore "02" (ILLEGAL DATA ADDRESS) e ritorna il codice funzione 83hex (131).

Esempio:

Eccezione alla richiesta sopra riportata.

ADDR	FUNC	DATA Except. Code	CRC HI	CRC LO
01	83	02	C0	F1

1.5.1 Codici d'eccezione

L'implementazione attuale del protocollo prevede solo quattro codici d'eccezione:

Code	Name	Meaning
01	ILLEGAL FUNCTION	Il codice di funzione ricevuto non corrisponde ad una funzione permessa sullo slave indirizzato.
02	ILLEGAL DATA ADDRESS	Il numero indirizzo cui fa riferimento il campo dati non è un registro permesso sullo slave indirizzato.
03	ILLEGAL DATA VALUE	Il valore da assegnare cui fa riferimento il campo dati non è permesso per questo registro.
07	NAK - NEGATIVE ACKNOWLEDGEMENT	La funzione non può essere eseguita nelle attuali condizioni operative o si è tentato di scrivere in un parametro a sola lettura.

1.6 Configurazione del sistema

La configurazione della linea seriale può essere eseguita programmando i parametri indicati di seguito.

23 – COMUNICAZIONE

23.1 - COMUNICAZIONE/RS485

Il drive ADV200 è provvisto di serie di una porta (connettore a vaschetta 9 poli D-SUB: XS) per il collegamento della linea seriale RS485 utilizzata per la comunicazione punto-punto drive-PC (tramite il software di configurazione GF-eXpress) oppure per il collegamento multidrop.

Il formato della linea seriale RS485 è: 8 bits dati, nessuna parità ed un bit di stop.

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc	Mod
23.1.1	**3800**	**Indirizzo drive**		UINT16		1	1	255	ERW	FVS

Impostazione dell'indirizzo al quale risponde il drive quando è connesso alla linea seriale RS485.

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc	Mod
23.1.2	**3802**	**Baud rate seriale**		ENUM		38400	0	2	ERW	FVS

Impostazione della velocità della comunicazione seriale RS485 (Baud Rate).

0 9600
1 19200
2 38400

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc	Mod
23.1.3	**3810**	**Parametri seriale**		ENUM		None,8,1	0	3	ERW	FVS

Impostazione del formato dei dati nella comunicazione seriale RS485.

0 None,8,1
1 None,8,2
2 Even,8,1
3 Odd,8,1

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc	Mod
23.1.4	**3804**	**Protocollo seriale**		ENUM		Modbus	0	1	ERW	FVS

Impostazione del protocollo di comunicazione seriale:

0 Modbus
1 Jbus

Impostando **0** si seleziona il protocollo di comunicazione seriale Modbus RTU (Remote Terminal Unit).

Impostando **1** si seleziona il protocollo ci comunicazione seriale Jbus. Il protocollo Jbus è funzionalmente identico al Modbus e se ne differenzia per la diversa numerazione degli indirizzi: nel Modbus questi partono da zero (0000 = 1° indirizzo) mentre nel JBUS partono da uno (0001 = 1° indirizzo) mantenendo questo scostamento per tutta la numerazione.

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc	Mod
23.1.5	**3806**	**Ritardo seriale**	ms	UINT16		0	0	1000	ERW	FVS

Impostazione del ritardo minimo tra la ricezione da parte del drive dell'ultimo byte e l'inizio della sua risposta. Tale ritardo evita conflitti sulla linea seriale quando l'interfaccia RS485 utilizzata non è preimpostata per una commutazione automatica Tx/Rx. Il parametro riguarda esclusivamente l'utilizzo della linea seriale standard RS485.

Esempio: se il ritardo della commutazione Tx/Rx sul master è al massimo di 20ms, l'impostazione del parametro Ritardo seriale deve essere superiore di 20ms: 22ms

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc	Mod
23.1.6	**3808**	**Dati scambio seriale**		BIT		0	0	1	ERW	FVS

Questo parametro abilita lo scambio della lettura delle parti Alta e Bassa delle words per i parametri di tipo FLOAT, UINT32 e INT32 quando si utilizza il protocollo Modbus o Jbus.

1.6.2 ADL300

20 - COMUNICAZIONE

Il drive ADL300 è provvisto di serie di una porta (connettore a vaschetta 9 poli D-SUB: XS) per il collegamento della linea seriale RS232 utilizzata per la comunicazione punto-punto drive-PC (tramite il software di configurazione GF-eXpress).

20.1 - COMUNICAZIONE/RS232

Menu	PAR	Descrizione	EU	Tipo	FB BIT	Def	Min	Max	Acc	Mod
20.1.1	**3800**	**Indirizzo drive**		UINT16		1	1	255	ERW	F__

Impostazione dell'indirizzo al quele risponde il drive quando è connesso alla linea seriale RS232.

Menu	PAR	Descrizione	EU	Tipo	FB BIT	Def	Min	Max	Acc	Mod
20.1.2	**3802**	**Baud rate seriale**		ENUM		38400	0	2	ERW	F__

Impostazione della velocità della comunicazione seriale RS232 (Baud Rate).
0 9600
1 19200
2 38400

Menu	PAR	Descrizione	EU	Tipo	FB BIT	Def	Min	Max	Acc	Mod
20.1.3	**3810**	**Parametri seriale**		ENUM		None,8,1	0	3	ERW	F__

Impostazione del formato della linea seriale RS232.
0 None,8,1
1 None,8,2
2 Even,8,1
3 Odd,8,1

Menu	PAR	Descrizione	EU	Tipo	FB BIT	Def	Min	Max	Acc	Mod
20.1.4	**3804**	**Protocollo seriale**		ENUM		Modbus	0	1	ERW	F__

Impostazione del protocollo di comunicazione seriale:
0 Modbus
1 Jbus

Impostando **0** si seleziona il protocollo di comunicazione seriale Modbus RTU (Remote Terminal Unit).

Impostando **1** si seleziona il protocollo di comunicazione seriale Jbus. Il protocollo Jbus è funzionalmente identico al Modbus e se ne differenzia per la diversa numerazione degli indirizzi: nel Modbus questi partono da zero (0000 = 1° indirizzo) mentre nel JBUS partono da uno (0001 = 1° indirizzo) mantenendo questo scostamento per tutta la numerazione.

Menu	PAR	Descrizione	EU	Tipo	FB BIT	Def	Min	Max	Acc	Mod
20.1.5	**3806**	**Ritardo seriale**	ms	UINT16		0	0	1000	ERW	F__

Impostazione del ritardo minimo tra la ricezione da parte del drive dell'ultimo byte e l'inizio della sua risposta. Tale ritardo evita conflitti sulla linea seriale quando l'interfaccia RS232 utilizzata non è preimpostata per una commutazione automatica Tx/Rx. Il parametro riguarda esclusivamente l'utilizzo della linea seriale standard RS232.

Esempio: se il ritardo della commutazione Tx/Rx sul master è al massimo di 20msec, l'impostazione del parametro **Ritardo seriale** deve essere superiore di 20msec: 22msec

Menu	PAR	Descrizione	EU	Tipo	FB BIT	Def	Min	Max	Acc	Mod
20.1.6	**3808**	**Dati scambio seriale**		BIT		0	0	1	ERW	F__

Questo parametro abilita lo scambio della lettura delle parti Alta e Bassa delle words per i parametri di tipo FLOAT, UINT32 e INT32 quando si utilizza il protocollo Modbus.

1.6.3 AFE200

14 - COMUNICAZIONE

Il drive AFE200 è provvisto di serie di una porta (connettore a vaschetta 9 poli D-SUB: XS) per il collegamento della linea seriale RS485 utilizzata per la comunicazione punto-punto drive-PC (tramite il software di configurazione GF-eXpress) oppure per il collegamento multidrop.

Il formato della linea seriale RS485 è: 8 bits dati, nessuna parità ed un bit di stop.

14.1 - COMUNICAZIONE/RS485

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc
14.1.1	**3800**	**Indirizzo drive**		UINT16		1	1	255	ERW

Impostazione dell'indirizzo al quele risponde il drive quando è connesso alla linea seriale RS485.

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc
14.1.2	**3802**	**Baud rate seriale**		ENUM		38400	0	2	ERW

Impostazione della velocità della comunicazione seriale RS485 (Baud Rate).
0 9600
1 19200
2 38400

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc
14.1.3	**3810**	**Parametri seriale**		ENUM		None,8,1	0	3	ERW

Impostazione del formato dei dati nella comunicazione seriale RS485.
0 None,8,1
1 None,8,2
2 Even,8,1
3 Odd,8,1

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc
14.1.4	**3804**	**Protocollo seriale**		ENUM		Modbus	0	1	ERW

Impostazione del protocollo di comunicazione seriale:
0 Modbus
1 Jbus

Impostando **0** si seleziona il protocollo di comunicazione seriale Modbus RTU (Remote Terminal Unit).

Impostando **1** si seleziona il protocollo ci comunicazione seriale Jbus. Il protocollo Jbus è funzionalmente identico al Modbus e se ne differenzia per la diversa numerazione degli indirizzi: nel Modbus questi partono da zero (0000 = 1° indirizzo) mentre nel JBUS partono da uno (0001 = 1° indirizzo) mantenendo questo scostamento per tutta la numerazione.

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc
14.1.5	**3806**	**Ritardo seriale**	ms	UINT16		0	0	1000	ERW

Impostazione del ritardo minimo tra la ricezione da parte del drive dell'ultimo byte e l'inizio della sua risposta. Tale ritardo evita conflitti sulla linea seriale quando l'interfaccia RS485 utilizzata non è preimpostata per una commutazione automatica Tx/Rx. Il parametro riguarda esclusivamente l'utilizzo della linea seriale standard RS485.

Esempio: se il ritardo della commutazione Tx/Rx sul master è al massimo di 20ms, l'impostazione del parametro Ser answer delay deve essere superiore di 20ms: 22ms

Menu	PAR	Descrizione	UM	Tipo	FB BIT	Def	Min	Max	Acc
14.1.6	**3808**	**Dati scambio seriale**		BIT		0	0	1	ERW

Questo parametro abilita lo scambio della lettura delle parti Alta e Bassa delle words per i parametri di tipo FLOAT, UINT32 e INT32 quando si utilizza il protocollo Modbus o Jbus.

2. Appendice

2.1 Condizioni anomale

Nei drive ADV200, ADL e AFE200 esistono parametri a 16 bits e parametri a 32 bits.

Per leggere e scrivere parametri del drive a 16 bits è sufficiente un Registro Modbus.

Per leggere e scrivere parametri del drive a 32 bits sono necessari due Registri Modbus.

Nei drive ADV200, ADL e AFE200 a tutti i parametri di sistema è assegnato un Ipa pari, non esistono parametri di sistema assegnati a Ipa dispari. Questa convenzione, per i parametri a 32 bits, permette di sfruttare Registro Modbus abbinato a Ipa dispari (Ipa + 1) per leggere parte alta.

Nel drive ADV200, ADL e AFE200 nella numerazione dei parametri oltre ai buchi lasciati da Ipa dispari esistono altri buchi.

Nei comandi Modbus dove è possibile leggere e scrivere più parametri del drive, a causa delle convenzioni adottare nell'assegnamento degli Ipa, si possono presentare diverse condizioni anomale (Parametri non esistenti a causa dei buchi, non specificata la corretta quantità di registri, ecc).

Di seguito in funzione del comando Modbus sono descritte alcune possibili situazioni con la corrispondente gestione.

2.2 ReadHoldingRegisters (03 – 0x03)

Questo comando permette di leggere Registri a 16 bits a cui sono abbinati i parametri del drive.

Nel messaggio di richiesta viene specificato il Registro di partenza da cui si ricava Ipa di partenza e il numero di Registri da leggere.

In caso di lettura di parametri inesistenti nel registro sarà tornato il valore 0.

Caso 1. Registri da leggere = 1

Tipicamente questo caso si presenta per leggere parametri del drive a 16 bits.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro a 16 bits esistente	Corretta
Se viene specificato Ipa di partenza di un parametro a 32 bits esistente	Errore
Se viene specificato Ipa di partenza di un parametro inesistente	Errore
Se viene specificato Ipa di partenza dispari che identifica la parte alta di un parametro a 32 bits	Errore

Caso 2. Registri da leggere = 2

Tipicamente questo caso si presenta per leggere parametri del drive a 32 bits.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro a 32 bits esistente	Corretta
Se viene specificato Ipa di partenza di un parametro a 16 bits	Corretta
Se viene specificato Ipa di partenza di un parametro inesistente	Errore
Se viene specificato Ipa di partenza dispari abbinato alla parte alta di un parametro a 32 bits	Errore
Se viene specificato Ipa di partenza di un parametro esistente o almeno uno dei successivi registri è abbinato ad un parametro esistente ma ultimo registro è abbinato a un parametro che causa il superamento del numero di registri richiesti	Errore

Caso 3. Registri da leggere = 3

Tipicamente questo caso si presenta per leggere contemporaneamente più parametri drive a 16 o 32 bits.

Il comando termina correttamente se almeno un Ipa esiste.

Se un ipa a 32 bits viene richiesto solo parzialmente solo la parte alta perché si fornisce Ipa di partenza abbinato alla parte alta o solo la parte bassa perché non si richiede un numero sufficiente di Registri il comando termina con errore.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro inesistente e anche tutti i successivi registri sono abbinati a parametri inesistenti	Errore
Se viene specificato Ipa di partenza di un parametro inesistente ma almeno uno dei successivi registri è abbinato ad un parametro esistente	Corretta
Se viene specificato Ipa di partenza di un parametro esistente ma alcuni dei successivi registri sono abbinati a parametri inesistenti	Corretta
Se viene specificato Ipa di partenza dispari, abbinato alla parte alta di un parametro a 32 bits	Errore
Se viene specificato Ipa di partenza di un parametro esistente e i successivi registri sono abbinati a parametri esistenti	Corretta
Se viene specificato Ipa di partenza di un parametro esistente o almeno uno dei successivi registri è abbinato ad un parametro esistente ma ultimo registro è abbinato a un parametro che causa il superamento del numero di registri richiesti	Errore

Caso 4. Registri da leggere = 4

Tipicamente questo caso si presenta per leggere contemporaneamente più parametri drive a 16 o 32 bits.

Il comando termina correttamente se almeno un Ipa esiste.

Se un ipa a 32 bits viene richiesto solo parzialmente solo la parte alta perché si fornisce Ipa di partenza abbinato alla parte alta o solo la parte bassa perché non si richiede un numero sufficiente di Registri il comando termina con errore.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro inesistente e anche tutti i successivi registri sono abbinati a parametri inesistenti	Errore
Se viene specificato Ipa di partenza di un parametro inesistente ma almeno uno dei successivi registri è abbinato ad un parametro esistente	Corretta
Se viene specificato Ipa di partenza di un parametro esistente ma alcuni dei successivi registri sono abbinati a parametri inesistenti	Corretta
Se viene specificato Ipa di partenza dispari, abbinato alla parte alta di un parametro a 32 bits	Errore
Se viene specificato Ipa di partenza di un parametro esistente e i successivi registri sono abbinati a parametri esistenti	Corretta
Se viene specificato Ipa di partenza di un parametro esistente o almeno uno dei successivi registri è abbinato ad un parametro esistente ma ultimo registro è abbinato a un parametro che causa il superamento del numero di registri richiesti	Errore

Caso 5. Registri da leggere > 3 (numero dispari)

L'analisi di questo caso può essere eseguita applicando gli stessi criteri applicati per Caso 3.

Caso 6. Registri da leggere > 4 (numero pari)

L'analisi di questo caso può essere eseguita applicando gli stessi criteri applicati per Caso 4.

2.3 Preset Single Register (06 – 0x06)

Questo comando permette di scrivere un Registro a 16 bits a cui è abbinato un parametro del drive.

Nel messaggio di richiesta viene specificato il Registro di partenza da cui si ricava Ipa di partenza.

Tipicamente questo caso si presenta per scrivere parametri del drive a 16 bits.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro a 16 bits esistente	Corretta
Se viene specificato Ipa di partenza di un parametro a 32 bits esistente	Errore
Se viene specificato Ipa di partenza di un parametro inesistente	Errore
Se viene specificato Ipa di partenza dispari abbinato alla parte alta di un parametro a 32 bits	Errore

2.4 Preset Multiple Registers (16 – 0x10)

Questo comando permette di scrivere Registri a 16 bits a cui sono abbinati i parametri del drive.

Nel messaggio di richiesta viene specificato il Registro di partenza da cui si ricava Ipa di partenza, il numero di Registri da scrivere e il numero di Byte dati da scrivere.

Caso 1. Registri da scrivere = 1

Tipicamente questo caso si presenta per scrivere parametri del drive a 16 bits.

Se si scrive il Registro ad un valore fuori dal range il comando termina con errore.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro a 16 bits esistente	Corretta
Se viene specificato Ipa di partenza di un parametro a 32 bits esistente	Errore
Se viene specificato Ipa di partenza di un parametro inesistente	Errore
Se viene specificato Ipa di partenza dispari abbinato alla parte alta di un parametro a 32 bits	Errore

Caso 2. Registri da scrivere = 2

Tipicamente questo caso si presenta per scrivere parametri del drive a 32 bits. Se errore viene intercettato analizzando Registri successivi al primo il comando termina con errore ma le operazioni di scrittura precedenti sono comunque state eseguite.

Se si scrivono più registri ed uno ha un valore fuori dal range il comando termina con errore ma le operazioni di scrittura precedenti sono comunque state eseguite.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro a 32 bits esistente	Corretta
Se viene specificato Ipa di partenza di un parametro a 16 bits esistente	Corretta
Se viene specificato Ipa di partenza di un parametro inesistente	Errore
Se viene specificato Ipa di partenza dispari abbinato alla parte alta di un parametro a 32 bits	Errore
Se viene specificato Ipa di partenza di un parametro esistente o almeno uno dei successivi registri è abbinato ad un parametro esistente ma ultimo registro è abbinato a un parametro che causa il superamento del numero di registri da scrivere	Errore

Caso 3. Registri da scrivere = 3

Tipicamente questo caso si presenta per scrivere contemporaneamente più parametri drive a 16 o 32 bits. Se errore viene intercettato analizzando Registri successivi al primo il comando termina con errore ma le operazioni di scrittura precedenti sono comunque state eseguite.

Se si scrivono più registri ed uno ha un valore fuori dal range il comando termina con errore ma le operazioni di scrittura precedenti sono comunque state eseguite.

Per uniformarlo ai comandi di lettura il comando di scrittura termina correttamente se almeno un Ipa esiste.

Se un Ipa a 32 bits viene scritto solo parzialmente (solo la parte alta perché si fornisce Ipa di partenza abbinato alla parte alta del parametro oppure solo la parte bassa del parametro perché non sono presenti un numero sufficiente di Registri) il comando termina con errore.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro inesistente e anche tutti i successivi registri sono abbinati a parametri inesistenti	Errore
Se viene specificato Ipa di partenza di un parametro inesistente ma almeno uno dei successivi registri è abbinato ad un parametro esistente	Corretta
Se viene specificato Ipa di partenza di un parametro esistente ma alcuni dei successivi registri sono abbinati a parametri inesistenti	Corretta
Se viene specificato Ipa di partenza dispari, abbinato alla parte alta di un parametro a 32 bits	Errore
Se viene specificato Ipa di partenza di un parametro esistente e i successivi registri sono abbinati a parametri esistenti	Corretta
Se viene specificato Ipa di partenza di un parametro esistente o almeno uno dei successivi registri è abbinato ad un parametro esistente ma ultimo registro è abbinato a un parametro che causa il superamento del numero di registri da scrivere	Errore

Caso 4. Registri da scrivere = 4

Tipicamente questo caso si presenta per scrivere contemporaneamente più parametri drive a 16 o 32 bits. Se errore viene intercettato analizzando Registri successivi al primo il comando termina con errore ma le operazioni di scrittura precedenti sono comunque state eseguite.

Se si scrivono più registri ed uno ha un valore fuori dal range il comando termina con errore ma le operazioni di scrittura precedenti sono comunque state eseguite.

Per uniformarlo ai comandi di lettura il comando di scrittura termina correttamente se almeno un Ipa esiste.

Se un Ipa a 32 bits viene scritto solo parzialmente (solo la parte alta perché si fornisce Ipa di partenza abbinato alla parte alta del parametro oppure solo la parte bassa del parametro perché non sono presenti un numero sufficiente di Registri) il comando termina con errore.

Condizione	Risultato operazione
Se viene specificato Ipa di partenza di un parametro inesistente e anche tutti i successivi registri sono abbinati a parametri inesistenti	Errore
Se viene specificato Ipa di partenza di un parametro inesistente ma almeno uno dei successivi registri è abbinato ad un parametro esistente	Corretta
Se viene specificato Ipa di partenza di un parametro esistente ma alcuni dei successivi registri sono abbinati a parametri inesistenti	Corretta
Se viene specificato Ipa di partenza dispari, abbinato alla parte alta di un parametro a 32	Errore
Se viene specificato Ipa di partenza di un parametro esistente e i successivi registri sono abbinati a parametri esistenti	Corretta
Se viene specificato Ipa di partenza di un parametro esistente o almeno uno dei successivi registri è abbinato ad un parametro esistente ma ultimo registro è abbinato a un parametri che causa il superamento del numero di registri da scrivere	Errore

Caso 5. Registri da scrivere > 3 (numero dispari)

L'analisi di questo caso può essere eseguita applicando gli stessi criteri applicati per Caso 3.

Caso 6. Registri da scrivere > 4 (numero pari)

L'analisi di questo caso può essere eseguita applicando gli stessi criteri applicati per Caso 4.

Table of Contents

1. Modbus RTU Protocol

1.1 Introduction

The drive parameters are seen by Modbus as 16-bit or 32-bit, depending on the type (BIT, ENUM, FLOAT, INT16, etc.).

Parameters seen as 16-bit occupy 1 Modbus register.

Parameters seen as 32-bit occupy 2 Modbus registers.

Modbus register number = Parameter index -1.

Example:

Parameter 600 "Dig ramp ref 1", type INT16 => (1) Modbus Register 599.

Parameter 3700 "Pad 1", type INT32 => (2) Modbus Register 3699 3700.

1.2 The MODBUS Protocol

The MODBUS protocol defines the format and the communication modes between a system controlling “master” and one or more “slaves” aimed at answering to the master requests. The protocol states how the master and the slaves start and stop their communication, how the messages can be exchanged and how the errors can be detected. A common line can host one master and 255 slaves; this is a protocol logic limit, the device number can be further limited by the physical interface; the present implementation foresees a maximum number of 32 slaves to be line-connected.

A transaction can be started exclusively by the master. A transaction can have a direct demand/response format or a broadcast format. The former is addressed to a single slave, the latter to all the line slaves, which, on their turn, give no response. A transaction can have a single demand/single response frame or a single broadcast message/no response frame.

Some protocol features have not been defined. They are: interface standard, baud rate, parity, stop bit number. Theprotocol allows also to choose between two communication “modes”: ASCII and RTU (Remote Terminal Unit). The RTU mode, which is the most efficient, is implemented in the Drives.

The JBUS protocol is similar to the MODBUS protocol; the only difference is given by the address numbering system: in MODBUS the numbering system starts from zero (0000 = 1st address) while in JBUS it starts from one (0001 = 1st address); this variance is maintained throughout the whole system. The following descriptions, if not otherwise stated, refer to both protocols.

Example:

	Modbus	Jbus
Parameter PAR 600 Dig ramp ref 1, type INT16	1 Modbus Register 599	1 Jbus Register 600
Parameter PAR 3700 PAD 1, type INT32	2 Modbus Registers 3699 - 3700	2 Jbus Registers 3700 - 3701

1.3 Message format

In order to communicate between the two devices, the message has to be contained into a “casing”. The casing leaves the transmitter via a “port” and it is “brought” along the line to a similar

- The slave address for the master stated transaction (the address 0 corresponds to a broadcast message sent to all the slaves).
- The code of the function (already performed or to be performed).
- The data to be exchanged.
- The error control according to the CRC16 algorithm.

If a slave detects an error in the received message (a format, parity or CRC16 error), the message is invalid and therefore rejected; when a slave detects an error in the message, it does not perform the required action and does not answer to the demand as if the address does not correspond to an on-line slave.

1.3.1 The address

As stated above, the MODBUS transactions always involve the master (which controls the line) and one slave at the time (with the exception of broadcast messages). In order to detect the message receiver, the first sent character is a byte containing the numeric address of the selected slave. Each slave owns therefore a different address number for its identification. The legal addresses go from 1 to 99, while a master message starting with the address 0 means that this is a “broadcast” message simultaneously addressed to all the slaves (the address 0 can not be allocated to a slave). Broadcast messages are those messages which do not need a response to perform their function, i.e. the allocations.

1.3.2 The function code

The second character of the message states the function to be performed by the master message; the slave response contains the same code, thus stating that the function has been performed.

An implemented subset of the MODBUS functions contains:

- 01 Read Coil Status (Not used for ADV-ADL-AFE drives)
- 02 Read Input Status (Not used for ADV-ADL-AFEdrives)
- 03 Read Holding Registers
- 04 Read Input registers
- 05 Force Single Coil (Not used for ADV-ADL-AFEdrives)
- 06 Preset Single register
- 07 Read Status
- 15 Force multiple Coils (Not used for ADV-ADL-AFEdrives)
- 16 Preset Multiple Registers

The 01 and 02 functions, so as the 03 and 04 functions, are similar and interchangeable. See chapter 3 for a complete and detailed description of the functions.

1.3.3 CRC16

The last two characters of the message contain the cyclic redundancy code (Cyclic Redundancy Check) calculated according to the CRC16 algorithm. As for the calculation of these two characters, the message (address, function code and data thus rejecting the parity and the start and stop bits) is considered as a single and continuous binary number whose most significative bit (MSB) is transmitted as first. The message is multiplied by x16 (it undergoes a 16-bit shift on the left) and then it is divided by $x^{16}+x^{15}+x^{2}+1$; it is stated as a binary number (1100000000000101). The integer quotient is rejected and the 16-bit remainder (it is initialized with FFFFh in order to avoid a zero made message) is added to the sent message. The obtained message, when the receiver slave has divided it by the same polynomial ($x^{16}+x^{15}+x^{2}+1$), must have a zero remainder if no error occurred (if not the slave calculates the CRC again).

Considering that the data serializing device (UART) transmits first the less significative bit (LSB) instead of the MSB as required by the CRC calculation, such calculation is performed by inverting the polynomial. Furthermore, as the MSB polynomial influences only the quotient and not the remainder, the remainder is deleted by making it equal to 1010000000000001.

The step by step procedure for the CRC16 calculation is the following:

1. Load a 16-bit register with FFFFh (the bit value is 1).
2. Perform the exclusive OR of the first character with the highest byte in the register; place the result in the register.
3. Perform a one-bit shift of the register on the right.
4. If the bit outcoming the register right side (flag) is 1, perform the exclusive OR between the 1010000000000001 generating polynomial and the register.
5. Repeat the steps 3 and 4 for eight times.
6. Perform the exclusive OR of the following character with the highest byte in the register; place the result in the register.
7. Repeat the steps from 3 to 6 for all the message characters.
8. The content of the 16-bit register is the CRC redundancy code to be added to the message.

1.3.4 Message synchronization

The message synchronization between the transmitter and the receiver is obtained by interposing a pause between

the messages, such pause being equal to 3.5 times the character period. If the receiver does not receive for a period equal to 4 characters, the message is considered to be over; as a consequence the following received byte is treated as the first byte of a new message: an address.

1.3.5 Serial line setting

Default communication settings:

- 1 bit di start
- 8 bits di dati (RTU protocol)
- 1 bit di stop
- no parity

The following values can be set:

N, 8, 1	(default)
N, 8, 2	
E, 8, 1	
O, 8, 1	

Communication baud rate settings:

Baudrate	Timeout byte-byte	
9600	4ms	
19200	2ms	
38400	1ms	(default)

1.4 Modbus functions for the drive

Here following is a detailed description of the MODBUS functions implemented for the Drive. All the values listed in the tables are hexadecimal.

1.4.1 Read Output Registers (03)

This function allows to read the value of 16-bit (word) registers containing Drive parameters. The broadcast mode is not allowed.

Example: 16 bit Parameter

Request

Together with the Drive address and the function code (03), the message contains the register starting address (starting Address) and the number of the registers to be read; they are both stated on two bytes. **The maximum number of registers which can be read is 125.**

Example:

- Drive address 01 (01_{hex})
- Parameter 600 "Dig ramp ref 1" 600 -1 = 257_{hex}.

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	CRC HI	CRC LO
01	03	02	57	00	01	34	62

Response

Together with the Drive address and the function code (03), the message includes a character containing the data byte number and some other characters containing the data. The registers require two bytes where the first one contains the most significative section.

Example:

Response to the above mentioned request. (Value 100 = 64_{hex}).

ADDR	FUNC Byte	DATA word Count	DATA word HI	DATA word LO	CRC HI	CRC LO
01	03	02	00	64	B9	AF

Nota ! in case the register selected range includes some reserved or missing registers, the value of these registers is set to 0. See Appendix.

Read Registers

In the case of parameters to 32 bits the reading is carried out using 2 Modbus registers.

Parameter 3808 "Serial swap data" is used to configure the content of the two registers, i.e. the lower part in the first register and the upper part in the second, or vice versa.

		Register 1	Register 2
Serial swap data	OFF	L	H
Serial swap data	ON	H	L

Example Long Parameter:

Request

- **Parameter 3808 "Serial swap data" = OFF**
- Drive address 01 (01_{hex})
- Parameter 3700 "Pad1" 3700 -1 = $E73_{hex}$ - Two registers reading.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	0E	73	00	02	37	38

Response

Value 456 = 01C8hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	01	C8	00	00	7A	31
			Low part		High part			

- **Parameter 3808 "Serial swap data" = ON**
- Drive address 01 (01_{hex})
- Parameter 3700 "Pad1" 3700 -1 = $E73_{hex}$ - Two registers reading.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	0E	73	00	02	37	38

Response

Value 456 = 01C8hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	00	00	01	C8	FA	35
			High part		Low part			

Together with the Drive address and the function code (03), the message includes a character containing the data byte number and some other characters containing the data. The registers require two bytes where the first one contains the most significative section.

Example: Float Parameter

Request

- **Parameter 3808 "Serial swap data" = OFF**
- Drive address 01 (01_{hex})
- Parameter 700 "Accelleration time" 700 -1 = $2BB_{hex}$ - Two registers reading.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	02	BB	00	02	B5	96

Response

Value 1.0 = 3F80 0000 hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	00	00	3F	80	EA	63
			Low part		High part			

- **Parameter 3808 "Serial swap data" = ON**
- Drive address 01 (01_{hex})
- Parameter 700 "Accelleration time" 700 -1 = $2BB_{hex}$ - Two registers reading.

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	03	02	BB	00	02	B5	96

Response

Value 1.0 = 3F80 0000 hex

ADDR	FUNC	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	03	04	3F	80	00	00	F7	CF
			High part		Low part			

Together with the Drive address and the function code (03), the message includes a character containing the data byte number and some other characters containing the data. The registers require two bytes where the first one contains the most significative section.

1.4.2 Read Input Registers (04)

This function is similar to the previous one.

1.4.3 Preset Single Register (06)

This function allows to set the value of a single 16-bit register. The broadcast mode is allowed.

Request

Together with the Drive address and the function code (06), the message contains the register address (parameter) on two bytes and the value to be allocated.

Example:

Writing parameter INT16 bit 600

- Drive address 01 (01_{hex})
- Register 600 -1 (257_{hex})
- Value 1234 ($4D2_{hex}$)

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	DATA word HI	DATA word LO	CRC HI	CRC LO
01	06	02	57	04	D2	BB	3F

Response

The response is given by transmitting again the received message after the register has been modified.

Example:

Response to the above mentioned request.

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	DATA word HI	DATA word LO	CRC HI	CRC LO
01	06	02	57	04	D2	BB	3F

1.4.4 Read Status (07)

This function allows to read the status of eight predefined bits with a compact message. The broadcast mode is not allowed.

Request

The message contains only the Drive address and the function code (07).

Example:

- Drive address 01 (01_{hex})

ADDR	FUNC	CRC HI	CRC LO
01	07	41	E2

Response

Together with the Drive address and the function code (07), the message includes a character containing the status bits.

Example:

Response to the above mentioned request.

ADDR	FUNC	DATA status byte	CRC HI	CRC LO
01	07	01	E3	F0

The bit meaning is the following:

Bit	Significato
0	Reserved
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	0 = Application; 1 = Boot

1.4.5 Preset Multiple Registers (16)

This function allows to set the value of a consecutive block made of 16-bit registers. The broadcast mode is allowed.

Request

Together with the Drive address and the function code (16), the message contains the starting address of the registers to be written (starting Address), the number of registers to be written, the number of bytes containing the data and the data characters.

Example:

- Drive address 01 (01_{hex})
- Starting Register 3700 (3700 -1 = $E73_{hex}$) Parameter Pad1 - Type Long 32bit.
- Number of registers to be written 2 (02_{hex})
- Value 16909069 (01020304_{hex})

ADDR	FUNC start	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	DATA word Count	DATA word HI	DATA word LO	DATA word HI	DATA word LO	CRC HI	CRC LO
01	10	0E	73	00	02	04	03	04	01	02	39	2A

Response

Together with the Drive address and the function code (16), the message contains the starting address (starting Address) and the number of written registers.

Example:

Response to the above mentioned request.

ADDR	FUNC start	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	CRC HI	CRC LO
01	10	0E	73	00	02	B2	FB

Write Registers

In the case of parameters to 32 bits the writing is carried out using 2 Modbus registers.

Parameter 3808 "Serial swap data" is used to configure the content of the two registers, i.e. the lower part in the first register and the upper part in the second, or vice versa.

		Register 1	Register 2
Serial swap data	OFF	L	H
Serial swap data	ON	H	L

Example Long parameter

- **Parameter 3808 “Serial swap data” = OFF**
- Drive address = 1
- Parameter 3700 Pad 1 – Long 3700 - 1 E73Hex – Number of registers to be written 2
- Value 456 = 01C8hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	0E	3E	00	02	04	01	C8	00	00	78	FC
							Low part		High part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	0E	3E	00	02	B2	FB

- **Parameter 3808 “Serial swap data” = ON**
- Drive address = 1
- Parameter 3700 Pad 1 – Long 3700 - 1 E73Hex –Number of registers to be written 2
- Value 456 = 01C8hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	0E	3E	00	02	04	00	00	01	C8	F8	F8
							High part		Low part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	0E	3E	00	02	B2	FB

Example parametro Float

- **Parameter 3808 “Serial swap data” = OFF**
- Drive address = 1
- Parameter 700 Acceleration1 – Float 700 – 1 2BBHex – Number of registers to be written 2
- Value 1.0 = 3F80 0000hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	02	BB	00	02	04	00	00	3F	80	B0	58
							Low part		High part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	02	BB	00	02	30	55

- **Parameter 3808 “Serial swap data” = ON**
- Drive address = 1

- Parameter 700 Acceleration1 – Float 700 – 1 2BBHex – Number of registers to be written 2.
- Value 1.0 = 3F80 0000hex

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	DATA Word Count	DATA Word HI	DATA Word LO	DATA Word HI	DATA Word LO	CRC	CRC
01	10	02	BB	00	02	04	3F	80	00	00	AD	F4
							High part		Low part			

ADDR	FUNC	DATA Start Addr HI	DATA Start Addr LO	N Register HI	N Register LO	CRC	CRC
01	10	02	BB	00	02	30	55

1.5 Error management

In MODBUS there are two kinds of errors which are managed in different ways: transmission errors and operating errors.

The transmission errors change the format, the parity (if used) or the CRC16 of the message. When the Drive detects such errors, it considers the message invalid and gives no response. If the message format is the right one but its function can not be performed, the error is an operating one. The Drive answers to this error with a particular message. This message contains the Drive address, the code of the required function, an error code and the CRC. In order to underline that the response is aimed at stating the presence of an error, the function code is returned with the most significative bit set with “1”.

Example (parameter does not exist):

- Drive address 01 (01_{hex})
- Register 601 (601 -1 = 258_{hex})

ADDR	FUNC	DATA start Addr HI	DATA start Addr LO	N. Register HI	N. Register LO	CRC HI	CRC LO
01	03	02	58	00	01	04	61

Response

The request refers to the content of the Register 601, which does not exist in the Drive slave. The slave answers with the error code “02” (ILLEGAL DATA ADDRESS) and goes back to the function codee 83hex (131).

Example:

Exception to the above mentioned request.

ADDR	FUNC	DATA Except. Code	CRC HI	CRC LO
01	83	02	C0	F1

1.5.1 Exception codes

This protocol implementation foresees only four exception codes:

Code	Name	Meaning
01	ILLEGAL FUNCTION	The received function code does not correspond to a function allowed on the addressed slave
02	ILLEGAL DATA ADDRESS	The address number, which the data field refers to, is not a register allowed on the addressed slave
03	ILLEGAL DATA VALUE	The value to be allocated, which the data field refers to, is not allowed for this register.
07	NAK - NEGATIVE	The function can not be performed with the present operating ACKNOWLEDGEMENT conditions or attempt to write a read-only parameter.

1.6 System configuration

Serial line configuration can be performed by setting the parameters indicated below.

23 – COMMUNICATION

23.1 – COMMUNICATION/RS485

The ADV200 drive is provided with a standard port (9 pole sub-D connector: XS) for connecting the RS485 serial line used for drive-PC point-to-point communication (via the GF-eXpress configuration software) or for the multidrop connection.

The RS485 serial line format is: 8 data bits, no parity and one stop bit.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
23.1.1	**3800**	**Drive address**		UINT16		1	1	255	ERW	FVS

Setting of the address to which the drive responds when connected to the RS485 serial line.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
23.1.2	**3802**	**Serial baud rate**		ENUM		38400	0	2	ERW	FVS

Setting of the RS485 serial communication speed (Baud Rate).

0 9600
1 19200
2 38400

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
23.1.3	**3810**	**Serial parameter**		ENUM		None,8,1	0	3	ERW	FVS

Setting of the format of the RS485 serial communication data.

0 None,8,1
1 None,8,2
2 Even,8,1
3 Odd,8,1

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
23.1.4	**3804**	**Serial protocol**		ENUM		Modbus	0	1	ERW	FVS

Setting of the serial communication protocol:

0 Modbus
1 Jbus

Setting to **0** selects the Modbus RTU (Remote Terminal Unit) serial communication protocol.

Setting to **1** selects the Jbus serial communication protocol. The Jbus protocol is functionally identical to the Modbus, except for the different numbering of addresses: in the Modbus these start from zero (0000 = 1st address) while in the JBUS they start from one (0001 = 1st address) and maintain this difference throughout numbering.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
23.1.5	**3806**	**Serial delay**	ms	UINT16		0	0	1000	ERW	FVS

Setting of the minimum delay between the drive receiving the last byte and starting its response. This delay avoids conflicts on the serial line when the RS485 interface that is used has not been pre-set for automatic Tx/Rx switching. The parameter only concerns the use of the standard RS485 serial line.

Example: if the delay in Tx/Rx switching on the master is a maximum of 20ms, the Ser answer delay parameter must be set to at least 20ms: 22ms

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
23.1.6	**3808**	**Serial swap data**		BIT		0	0	1	ERW	FVS

This parameter enables the exchange of the reading of the High and Low parts of the words for FLOAT, UINT32 and INT32 type parameters when using the Modbus or Jbus protocol.

1.6.2 ADL300

20 - COMMUNICATION

The ADL300 drive is provided with a standard port (9 pole sub-D connector: XS) for connecting the RS232 serial line used for drive-PC point-to-point communication (via the GF-eXpress configuration software).

20.1 - COMMUNICATION/RS232

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
20.1.1	**3800**	**Drive address**		UINT16		1	1	255	ERW	F__

Setting of the address to which the drive responds when connected to the RS232 serial line.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
20.1.2	**3802**	**Serial baudrate**		ENUM		38400	0	2	ERW	F__

Setting of the RS232 serial communication speed (Baud Rate).
0 9600
1 19200
2 38400

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
20.1.3	**3810**	**Serial parameter**		ENUM		None,8,1	0	3	ERW	F__

Setting of the format of the RS232 serial line.
0 None,8,1
1 None,8,2
2 Even,8,1
3 Odd,8,1

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
20.1.4	**3804**	**Serial protocol**		ENUM		Modbus	0	1	ERW	F__

Setting of the serial communication protocol:
0 Modbus
1 Jbus

Setting to **0** selects the Modbus RTU (Remote Terminal Unit) serial communication protocol.

Setting to **1** selects the Jbus serial communication protocol. The Jbus protocol is functionally identical to the Modbus, except for the different numbering of addresses: in the Modbus these start from zero (0000 = 1st address) while in the JBUS they start from one (0001 = 1st address) and maintain this difference throughout numbering.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
20.1.5	**3806**	**Serial delay**	ms	UINT16		0	0	1000	ERW	F__

Setting of the minimum delay between the drive receiving the last byte and starting its response. This delay avoids conflicts on the serial line when the RS232 interface that is used has not been pre-set for automatic Tx/Rx switching. The parameter only concerns the use of the standard RS232 serial line.

Example: if the delay in Tx/Rx switching on the master is a maximum of 20ms, the **Ser answer delay** parameter must be set to at least 20ms: 22ms

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc	Mod
20.1.6	**3808**	**Serial swap data**		BIT		0	0	1	ERW	F__

This parameter enables the exchange of the reading of the High and Low parts of the words for FLOAT, UINT32 and INT32 type parameters when using the Modbus protocol.

14 - COMMUNICATION

Il drive AFE200 è provvisto di serie di una porta (connettore a vaschetta 9 poli D-SUB: XS) per il collegamento della linea seriale RS485 utilizzata per la comunicazione punto-punto drive-PC (tramite il software di configurazione GF-eXpress) oppure per il collegamento multidrop.

Il formato della linea seriale RS485 è: 8 bits dati, nessuna parità ed un bit di stop.

14.1 - COMMUNICATION/RS485

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc
14.1.1	**3800**	**Drive address**		UINT16		1	1	255	ERW

Setting of the address to which the drive responds when connected to the RS485 serial line.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc
14.1.2	**3802**	**Serial baud rate**		ENUM		38400	0	2	ERW

Setting of the RS485 serial communication speed (Baud Rate).

0 9600
1 19200
2 38400

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc
14.1.3	**3810**	**Serial parameter**		ENUM		None,8,1	0	3	ERW

Setting of the format of the RS485 serial communication data.

0 None,8,1
1 None,8,2
2 Even,8,1
3 Odd,8,1

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc
14.1.4	**3804**	**Serial protocol**		ENUM		Modbus	0	1	ERW

Setting of the serial communication protocol:

0 Modbus
1 Jbus

Setting to **0** selects the Modbus RTU (Remote Terminal Unit) serial communication protocol.

Setting to **1** selects the Jbus serial communication protocol. The Jbus protocol is functionally identical to the Modbus, except for the different numbering of addresses: in the Modbus these start from zero (0000 = 1st address) while in the JBUS they start from one (0001 = 1st address) and maintain this difference throughout numbering.

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc
14.1.5	**3806**	**Serial delay**	ms	UINT16		0	0	1000	ERW

Setting of the minimum delay between the drive receiving the last byte and starting its response. This delay avoids conflicts on the serial line when the RS485 interface that is used has not been pre-set for automatic Tx/Rx switching. The parameter only concerns the use of the standard RS485 serial line.

Example: if the delay in Tx/Rx switching on the master is a maximum of 20ms, the Ser answer delay parameter must be set to at least 20ms: 22ms

Menu	PAR	Description	UM	Type	FB BIT	Def	Min	Max	Acc
14.1.6	**3808**	**Serial swap data**		BIT		0	0	1	ERW

This parameter enables the exchange of the reading of the High and Low parts of the words for FLOAT, UINT32 and INT32 type parameters when using the Modbus or Jbus protocol.

2. Appendix

2.1 Fault conditions

The ADV200, ADL and AFE200 drives have 16-bit and 32-bit parameters.

Only one Modbus Register is required to read and write 16-bit drive parameters.

Two Modbus Registers are necessary for reading and writing 32-bit drive parameters.

An even IPA is assigned to all system parameters in the ADV200, ADL and AFE200 drives. No system parameters are assigned to an odd IPA. This convention, for 32-bit parameters, makes it possible to use the Modbus Register associated with odd IPAs (IPA + 1) to read the upper part.

In the ADV200, ADL and AFE200 drives there are other gaps in parameter numbering besides those left by the odd IPAs.

In Modbus commands where several drive parameters can be read and written, a number of fault conditions may occur (inexistent parameters due to gaps, correct number of registers not specified, etc.) due to the conventions adopted to assign IPAs.

Some possible situations for different Modbus commands and their relative management are described below.

2.2 ReadHoldingRegisters (03 – 0x03)

This command enables reading of 16-bit Registers with which the drive parameters are associated.

The Starting Register, from which the starting IPA and number of Registers to read are obtained, is specified in the request message.

If inexistent parameters are read the value 0 is returned to the register.

Example 1. Registers to read = 1

This is normally the case when 16-bit drive parameters are read.

Condition	Result of operation
If a starting IPA of an existing 16-bit parameter is specified	Correct
If a starting IPA of an existing 32-bit parameter is specified	Error
If a starting IPA of an inexistent parameter is specified	Error
If an odd starting IPA is specified identifying the upper part of a 32-bit parameter	Error

Example 2. Registers to read = 2

This is normally the case when 32-bit drive parameters are read.

Condition	Result of operation
If a starting IPA of an existing 32-bit parameter is specified	Correct
If a starting IPA of a 16-bit parameter is specified	Correct
If a starting IPA of an inexistent parameter is specified	Error
If an odd starting IPA is specified associated with the upper part of a 32-bit parameter	Error
If a starting IPA of an existing parameter is specified or at least one of the subsequent registers is associated with an existing parameter but the last register is associated with a parameter that causes the requested number of registers to be exceeded	Error

Example 3. Registers to read = 3

This is normally the case when 16-bit or 32-bit drive parameters are read at the same time.

The command completes correctly if at least one IPA exists.

If a 32-bit IPA is only partially requested, either only the upper part if supplying a starting IPA associated with the upper part or only the lower part if an insufficient number of Registers are requested, the command completes with an error.

Condition	Result of operation
If a starting IPA of an inexistent parameter is specified and all the subsequent registers are also associated with inexistent parameters	Error
If a starting IPA of an inexistent parameter is specified but at least one of the subsequent registers is associated with an existing parameter	Correct
If a starting IPA of an existing parameter is specified but some of the subsequent registers are associated with inexistent parameters	Correct
If an odd starting IPA is specified, associated with the upper part of a 32-bit parameter	Error
If a starting IPA of an existing parameter is specified and the subsequent registers are associated with existing parameters	Correct
If a starting IPA of an existing parameter is specified or at least one of the subsequent registers is associated with an existing parameter but the last register is associated with a parameter that causes the requested number of registers to be exceeded	Error

Example 4. Registers to read = 4

This is normally the case when several 16-bit or 32-bit drive parameters are read at the same time.

The command completes correctly if at least one IPA exists.

If a 32-bit IPA is only partially requested, either only the upper part if supplying a starting IPA associated with the upper part or only the lower part if an insufficient number of Registers are requested, the command completes with an error.

Condition	Result of operation
If a starting IPA of an inexistent parameter is specified and all the subsequent registers are also associated with inexistent parameters	Error
If a starting IPA of an inexistent parameter is specified but at least one of the subsequent registers is associated with an existing parameter	Correct
If a starting IPA of an existing parameter is specified but some of the subsequent registers are associated with inexistent parameters	Correct
If an odd starting IPA is specified, associated with the upper part of a 32-bit parameter	Error
If a starting IPA of an existing parameter is specified and the subsequent registers are associated with existing parameters	Correct
If a starting IPA of an existing parameter is specified or at least one of the subsequent registers is associated with an existing parameter but the last register is associated with a parameter that causes the requested number of registers to be exceeded	Error

Example 5. Registers to read > 3 (odd number)

This case can be examined by applying the same criteria as for Example 3.

Example 6. Registers to read > 4 (even number)

This case can be examined by applying the same criteria as for Example 4.

2.3 Preset Single Register (06 – 0x06)

This command enables writing of a 16-bit Register with which a drive parameter is associated.

The request message specifies the Starting Register from which the starting IPA is obtained.

This is normally the case when writing 16-bit drive parameters.

Condition	Result of operation
If a starting IPA of an existing 16-bit parameter is specified	Correct
If a starting IPA of an existing 32-bit parameter is specified	Error
If a starting IPA of an inexistent parameter is specified	Error
If an odd starting IPA is specified associated with the upper part of a 32-bit parameter	Error

2.4 Preset Multiple Registers (16 – 0x10)

This command enables writing of 16-bit Registers with which the drive parameters are associated.

The Starting Register, from which the starting IPA, the number of Registers to write and number of data Bytes to write are obtained, is specified in the request message.

Example 1. Registers to write = 1

This is normally the case when writing 16-bit drive parameters.

If writing the Register with a value that is out of range the command completes with an error.

Condition	Result of operation
If a starting IPA of an existing 16-bit parameter is specified	Correct
If a starting IPA of an existing 32-bit parameter is specified	Error
If a starting IPA of an inexistent parameter is specified	Error
If an odd starting IPA is specified associated with the upper part of a 32-bit parameter	Error

Example 2. Registers to write = 2

This is normally the case when writing 32-bit drive parameters. If the error is intercepted when analysing Registers after the first the command completes with an error but the previous write operations are still executed.

If several registers are written and one has a value that is out of range the command completes with an error but the previous write operations are still executed.

Condition	Result of operation
If a starting IPA of an existing 32-bit parameter is specified	Correct
If a starting IPA of an existing 16-bit parameter is specified	Correct
If a starting IPA of an inexistent parameter is specified	Error
If an odd starting IPA is specified associated with the upper part of a 32-bit parameter	Error
If a starting IPA of an existing parameter is specified or at least one of the subsequent registers is associated with an existing parameter but the last register is associated with a parameter that causes the number of registers to be written to be exceeded	Error

Example 3. Registers to write = 3

This is normally the case when several 16-bit or 32-bit drive parameters are written at the same time. If the error is intercepted when analysing Registers after the first the command completes with an error but the previous write operations are still executed.

If several registers are written and one has a value that is out of range the command completes with an error but the previous write operations are still executed.

To adapt it to the read commands the write command completes correctly if at least one IPA exists.

If a 32-bit IPA is only partially written (only the upper part if supplying a starting IPA associated with the upper part of the parameter or only the lower part of the parameter because there are not a sufficient number of Registers), the command completes with an error.

Condition	Result of operation
If a starting IPA of an inexistent parameter is specified and all the subsequent registers are also associated with inexistent parameters	Error
If a starting IPA of an inexistent parameter is specified but at least one of the subsequent registers is associated with an existing parameter	Correct
If a starting IPA of an existing parameter is specified but some of the subsequent registers are associated with inexistent parameters	Correct
If an odd starting IPA is specified, associated with the upper part of a 32-bit parameter	Error
If a starting IPA of an existing parameter is specified and the subsequent registers are associated with existing parameters	Correct
If a starting IPA of an existing parameter is specified or at least one of the subsequent registers is associated with an existing parameter but the last register is associated with a parameter that causes the number of registers to be written to be exceeded	Error

Example 4. Registers to write = 4

This is normally the case when several 16-bit or 32-bit drive parameters are written at the same time. If the error is intercepted when analysing Registers after the first the command completes with an error but the previous write operations are still executed.

If several registers are written and one has a value that is out of range the command completes with an error but the previous write operations are still executed.

To adapt it to the read commands the write command completes correctly if at least one IPA exists.

If a 32-bit IPA is only partially written (only the upper part if supplying a starting IPA associated with the upper part of

the parameter or only the lower part of the parameter because there are not a sufficient number of Registers), the command completes with an error.

Condition	Result of operation
If a starting IPA of an inexistent parameter is specified and all the subsequent registers are also associated with inexistent parameters	Error
If a starting IPA of an inexistent parameter is specified but at least one of the subsequent registers is associated with an existing parameter	Correct
If a starting IPA of an existing parameter is specified but some of the subsequent registers are associated with inexistent parameters	Correct
If an odd starting IPA is specified, associated with the upper part of a 32-bit parameter	Error
If a starting IPA of an existing parameter is specified and the subsequent registers are associated with existing parameters	Correct
If a starting IPA of an existing parameter or at least one of the subsequent registers is associated with an existing parameter but the last register is associated with a parameter that causes the number of registers to be written to be exceeded	Error

Example 5. Registers to write > 3 (odd number)

This case can be examined by applying the same criteria as for Example 3.

Example 6. Registers to write > 4 (even number)

This case can be examined by applying the same criteria as for Example 4.

Instruction Manual
Series: Modbus
Revision: 0.2
Date: 12-1-2023
Code: 1S9H63

WEG Automation Europe S.r.l.
Via Giosuè Carducci, 24
21040 Gerenzano (VA) · Italy

Technical Assistance: technohelp@weg.net
Customer Service: salesmotion@weg.net