Serial Communication

Serial Communication Overview

The serial communication link enables communication between the drive and host (terminal, PC, or high-level controller) using ASCII-coded messages transmitted over an asynchronous, multi-drop line.

When the host and drive communicate through serial communication, a proprietary set of commands and variables, called **VarCom**, are used to configure, control and monitor the drive.

The communication interface can be a graphical software interface, such as ServoStudio, or a user-designed application, or a basic terminal.

This chapter describes the serial communication protocol used by the drive and its host.

Serial Communication Specifications

Communications port	RS232, USB
Baud rate	115200 bits per second (bps)
Start bits	1
Data bits	8
Stop bits	1
Parity	None
Hardware handshake	None
Software handshake	None
Character	ASCII code
Data error check	8-bit checksum

Control Code Definitions

Name	Symbol	Hex
Line feed	<LF>	0Ah
Carriage return	<CR>	0Dh
Space	<SP>	20h
Delay	<DLY>	Indicates delay due to internal drive processing of information

Communication Summary

Drive-to-Terminal Transmission	Terminal-to-Drive Transmission	Protocol Flags (Variables)
■ Character echoes ■ Prompts ■ Variable values ■ Error/fault messages	■ Commands ■ Variable values ■ Variable queries	ECHO MSGPROMPT CHECKSUM

Data Transmission Format

To enable proper serial communication between the drive and the host, they must both use the same data transmission format:

- Full-duplex
- 8 bits per character
- No parity
- 1 start bit
- 1 stop bit
- Baud rate: 115200 bps
- Hardware: RS232 or USB serial port

Drive Addressing

For more information, refer to the drive user manual.

CDHD: The drive can be addressed and controlled on a single-line RS232 (C7 interface), or on a daisy-chained RS232 (C8 interface), or a USB (C1 interface) line. The CDHD has two rotary switches for setting the drive address.

DDHD: The DDHD can be addressed and controlled on a daisy-chained RS232 line. The DDHD has a rotary switch for setting the node address.

Single-Line Configuration

In a single-line RS232 configuration, the drive is connected to the C7 interface, and assigned address 0 by setting both rotary switches to 0.

By default, the rotary switches are set to 0, and the drive assumes a single-line configuration.

Daisy-Chain (Multi-Drop) Configuration

In a daisy-chain RS232 configuration, all drives must be daisy-chained through the C8 interface. Each drive must have a unique address to enable its identification on the network. When configuring a daisy-chain, address 0 cannot be used.

You can communicate with any or all drives on the daisy-chain from any RS232 or USB port on any of the daisy-chained drives.

- To communicate with an individual drive in a daisy-chain, enter the following at the prompt:

  \x <Enter>

  Where **x** = the address setting of the drive.

- To communicate simultaneously with all drives on the chain, enter the following at the prompt:

  *** <Enter>**

  This is called global addressing. When using global addressing, no character echo to the terminal occurs.

Variables and Commands

When the host and drive are communicating through a serial connection, VarCom is used to configure, control and monitor the drive. The VarCom set of functions includes:

- **Commands**: instruct the drive to perform an operation.
- **Read-only variables**: calculated and/or set by the drive, and used to monitor the drive and its operational status.

  To read a variable, type the VarCom mnemonic followed by <Enter> (carriage return, CR). The drive returns the value of the variable.
- **Read/Write variables**: used to configure and monitor the drive.

  To set a variable value, type the VarCom mnemonic, a space (or =), the value, and then <Enter> (carriage return, CR).

Data Control

The drive can process approximately 16 characters per millisecond (at 115200 baud rate).

The operating system recognizes backspaces and resets upon receipt of an <Esc> character.

The following VarCom variables allow you to configure communication responses between drive and host.

ECHO	Enables/disables the serial port character echo. If echo is enabled, characters received via the serial port are echoed back to the serial port and displayed on the computer monitor. ECHO 0 = Serial port echo disabled ECHO 1 = Serial port echo enabled ECHO allows the host to check the validity of the information received by the drive.
MSGPROMPT	Defines whether asynchronous messages and the prompt from the drive are sent to the serial port (and to the host computer) 0 = Messages and prompt disabled 1 = Messages and prompt enabled
CHECKSUM	Enables/disables checksum protection on the message. 0 = Message checksum disabled (default) 1 = Message checksum enabled The checksum is an 8-bit value, displayed within brackets < >. For example, 0x1F checksum is displayed as <1F> at the end of the message before the carriage return.

Message Format

The message format is the structure by which the drive processes ASCII-coded messages. Messages from the host to the drive are used to send commands, to set variables, or to query the drive. Messages from the drive to the host contain the response to queries.

This message format has two main elements: **message unit** and **message termination**, as shown in the following figure.

The checksum utility is optional.

Note: *Start* has no significance; it simply represents the drive waiting for the host to send a message.

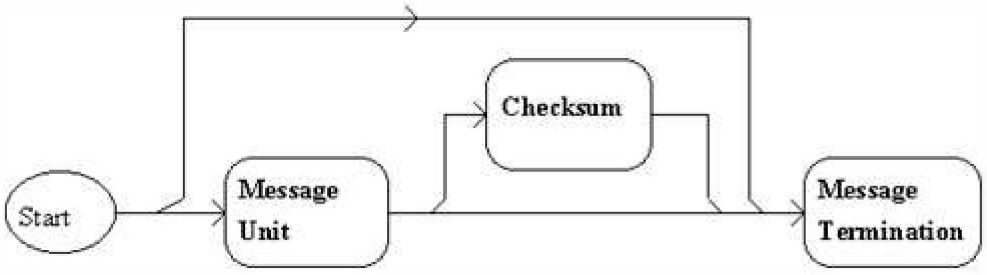

Figure 4-1. Message Format

Message Unit

A message unit is a block of information tha is transmitted on the communications link. The basic message unit is shown in the following figure.

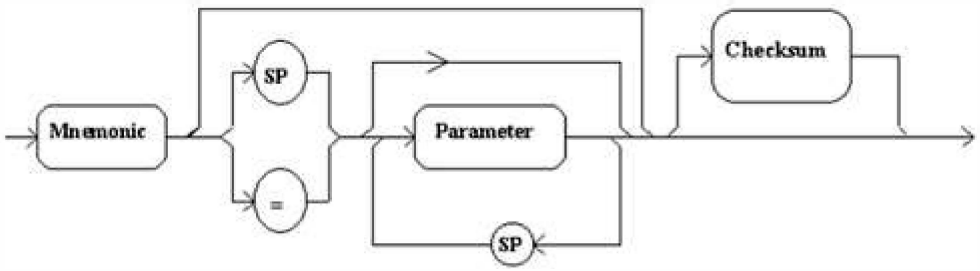

Figure 4-2. Message Unit

A message unit includes a header (VarCom mnemonic) with or ithout parameters. The header defines the context of the parameters that follow it. Messages sent from the host to the drives always have headers. Messages from the drive to the host do not generally include a header.

When used, parameters are separated from the mnemonic by either a space or an assignment operator. Parameters must be separated by spaces.

The drive can receive only a single message unit in a message format.

Message Termination

Message termination refers to the end of the message being sent.

Messages transmitted by the host are terminated by a carriage return (CR) – ASCII character 0DH.

Figure 4-3. Host Message Termination Format

Messages transmitted by the drive are teminated by a carriage return/line feed (CR/LF) combination – ASCII characters 0DH/0AH.

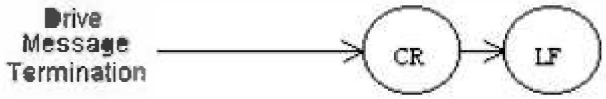

Figure 4-4. Drive Message Termination Format

The drive also accepts a message termintion sent without any additional information.

Complete Message Format with Checksum

The CHECKSUM block is used only when CHECKSUM command is set.

The drive is configured to accept incoming messages with or without checksum, and to append checksum to outgoing message according to the CHECKSUM variable.

Checksum is represented by two ASCII digits within brackets <> preceding the <CR>.

The complete message format is shown in the following figure.

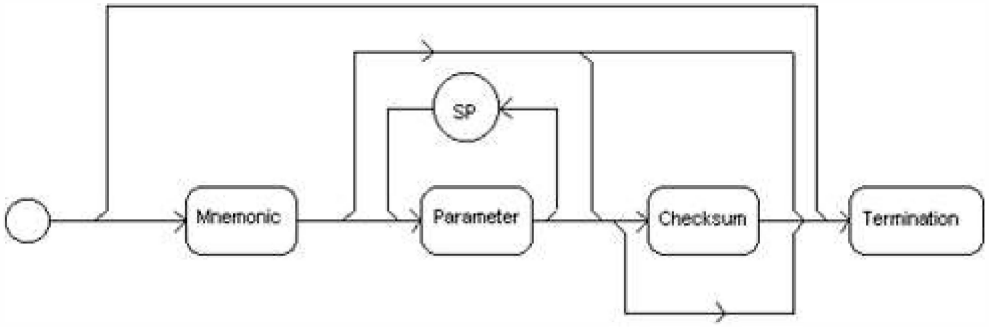

Figure 4-5. Complete Message Format

Units

Within a message or command, units are enlosed in brackets [ ]. For example:

- Message to drive: MPOLES
- Message from drive: 4 [poles]

When a command from the host contains units, the drive ignores the unit information.

Asynchronous Error Messages

The asynchronous error message function is enabled by the variable MSGPROMPT.

- If this function is enabled, and an error or fault occurs, the drive transmits a brief error message to the host.
- If the function is disabled, the error message is transmitted after a <CR> message termination is detected by the drive. This occurs whenever the host sends a message to the drive.

The variable MSGPROMPT also controls the prompt sent by the drive at the end of a message.

- If echoing is enabled, the characters in the message are all echoed before the error message is transmitted. Even though an error has occurred and its message returned to the host, the drive still accepts new incoming messages and attempts to execute them.
- If echoing is disabled, the error message is transmitted after the <CR> message termination is detected by the drive.

The drive must detect a new error or fault before transmitting an error message.

Examples of Serial Protocol

The following examples demonstrate serial protocol between the drive and a host.

Issuing a Command or Variable

In Examples 1 through 5, drive parameters are defined as:

```
ADDR 0
CHECKSUM 0
```

```
ECHO 1
MSGPROMPT 1
```

Example 1 – Command

EN (drive enable)

Sequence #	1	2	3	4	5	6	7	8	9	10	11
User Enters	**E**		**N**		**<CR>**						
Drive Returns		**E**		**N**		**<CR>**	**<LF>**	**<DLY>**	-	-	**>**

Displayed on terminal:

```
-->EN
-->
```

Example 2 – Command/Variable – Returns Multiple Lines of Values

This type of command typically has a longer delay due to the large amount of data that is output to the screen.

DUMP (return drive parameter values)

Sequence #	1	2	3	4	5	6	7	8	9	10
User Enters	**D**		**U**		**M**		**P**		**<CR>**	
Drive Returns		**D**		**U**		**M**		**P**		**<CR>**

Sequence #	11	12	13	14	15	16	17	18
User Enters								
Drive Returns	**<LF>**	**<DLY>**	**<VAR1>**	**<SP>**	**<VAL1>**	**<CR>**	**<LF>**	**<VAR2>**

Sequence #	19	20	21	22	23	24	25	26
User Enters								
Drive Returns	**<SP>**	**<VAL2>**	**<CR>**	**<LF>**	**<VAR*n*>**	**<SP>**	**<VAL*n*>**	**<LF>**

Sequence #	27	28	29	30
User Enters				
Drive Returns	**<CR>**	-	-	**>**

Displayed on terminal:

```
-->DUMP
-->var1 val1
-->var2 val2
-->varn valn
```

Example 3 – Command/Variable – Returns Multiple Values

J (jog)

Sequence #	1	2	3	4	5	6	7	8	9	10
User Enters	**J**		**<CR>**							
Drive Returns		**J**		**<CR>**	**<LF>**	**<DLY>**	**<VAL1>**	**<SP>**	**<VAL2>**	**<CR>**

Sequence #	11	12	13	14
User Enters				
Drive Returns	**<LF>**	**-**	**-**	**>**

Displayed on terminal:

```
-->J
-->nnnnn nnnnn
-->
```

Example 4 – Reading a Variable Value

MPOLES (single pole motor with value 2)

Sequence #	1	2	3	4	5	6	7	8	9	10
User Enters	**M**		**P**		**O**		**L**		**E**	
Drive Returns		**M**		**P**		**O**		**L**		**E**

Sequence #	11	12	13	14	15	16	17	18	19	20
User Enters	**S**		**<CR>**							
Drive Returns		**S**		**<CR>**	**<LF>**	**<DLY>**	**2**	**<SP>**	**[**	**p**

Sequence #	21	22	23	24	25	26	27	28	29	30	31
User Enters											
Drive Returns	**o**	**l**	**e**	**s**	**]**	**<CR>**	**<LF>**	**<DLY>**	**-**	**-**	**>**

Displayed on terminal:

```
-->MPOLES
2 [poles]
-->
```

Example 5 – Defining a Variable Value

ACC (acceleration with value 50000)

Sequence #	1	2	3	4	5	6	7	8	9	10
User Enters	**A**		**C**		**C**		**=**		**5**	
Drive Returns		**A**		**C**		**C**		**=**		**5**

Sequence #	11	12	13	14	15	16	17	18	19	20
User Enters	**0**		**0**		**0**		**0**		**<CR>**	
Drive Returns		**0**		**0**		**0**		**0**		**<CR>**

Sequence #	21	22	23	24	25
User Enters					
Drive Returns	**<LF>**	**<DLY>**	**-**	**-**	**>**

Displayed on terminal:

```
-->ACC=50000
-->
```

Multi-Drop Mode

In Examples 6 and 7, drive parameter values are defined as:

```
ADDR 3
ECHO 1
MSGPROMPT 1
```

Example 6 – Addressing a Daisy-Chained Drive

The range of values for ADDR is 0 to 99. A value other than 0 puts the system in Multi-drop mode, which results in a different prompt.

Sequence #	1	2	3	4	5	6	7	8	9	10	11
User Enters	****		**3**		**<CR>**						
Drive Returns		****		**3**		**<CR>**	**<LF>**	**<DLY>**	**3**	**-**	**>**

Displayed on terminal:

```
-->\3
3->
```

Example 7 – Reading a Variable

IMAX (drive current limit)

Sequence #	1	2	3	4	5	6	7	8	9	10
User Enters	**I**		**M**		**A**		**X**		**<CR>**	
Drive Returns		**I**		**M**		**A**		**X**		**<CR>**

Sequence #	11	12	13	14	15	16	17	18	19	20
User Enters										
Drive Returns	**<LF>**	**1**	**5**	**.**	**6**	**9**	**7**	**<CR>**	**<LF>**	**<DLY>**

Sequence #	21	22	23
User Enters			
Drive Returns	**3**	**-**	**>**

Displayed on terminal:

```
-->IMAX
3->15.697
3->
```

Serial Checksum

Example 8 – Variable

In this example, drive parameters are defined as:

```
ADDR 0
CHECKSUM 1
ECHO 1
MSGPROMPT 1
```

ACC (acceleration) with value 25000

Sequence #	1	2	3	4	5	6	7	8	9	10
User Enters	**A**		**C**		**C**		**=**		**2**	
Drive Returns		**A**		**C**		**C**		**=**		**2**

Sequence #	11	12	13	14	15	16	17	18	19	20
User Enters	**5**		**0**		**0**		**0**		**<**	
Drive Returns		**5**		**0**		**0**		**0**		**<**

Sequence #	21	22	23	24	25	26	27	28	29	30	31
User Enters	**F**		**B**		**>**		**<CR>**				
Drive Returns		**F**		**B**		**>**		**<CR>**	-	-	**>**

Character	Hex Value	ASCII Value
A	41	65
C	43	67
C	43	67
=	3D	61
2	32	50
5	35	53
0	30	48
0	30	48
0	30	48

```
Checksum=0xFF& (0x41+0x43+0x43+0x3d+0x32+0x35+0x30+0x30+0x30)
=0xFF & 0x01FB=0xFB
```

Note: Enter the last two characters of the HEX VALUE sum before the <CR>. Between brackets < >

Displayed on terminal:

```
//setting the checksum
-->CHECKSUM 1
//sending command to the drive with checksum appended
-->ACC=25000<FB>
//checking the actual value stored at the drive
-->ACC
//the reply is appended by checksum
25000.000[rpm/s]<7E>
-->
```