IB-11B041

Emerald Series Controller FEBRUARY 2004

Emerald Controller Serial Communications

Application Note

Industrial Indexing Systems, Inc.

Revision: D
Approved By:

REVISION HISTORY

07/23/02 – REV C:

1) Added note to operations code 4,5,6,and 7 that they have been preceded by operation codes 27,28,29, and 30. sam

02/27/04 – REVD:

1) Added serial packets RDERRPRLOG (opcode 35) and CLRFIXEDMEMORY (opcode 36) and corrected the size of the Scan Mode field for packet RDEVENTINFO (opcode 31). lac

Emerald Series Controller (ESC) Serial Commands

This document defines the communications protocol of the Emerald Series Controller.

The Emerald Series Controller can be interfaced to a host computer system using serial communications. The protocol is designed to handle point-to-point and multidrop communications hardware. The host can control the operation of the ESC in the following ways:

1. Read and write integer (2 or 4 byte) and floating point (8 byte) variables.

2. Read the Emerald Controller "Status Word".

3. Read flags and I/O.

4. Set flags and I/O.

5. Trace program execution and trap the value of up to two variables at each line traced.

6. Download an executable program with associated configuration data.

7. Reset the program and variable areas of the Emerald Controller.

8. Stop program execution.

9. Start program execution.

Serial commands are implemented using a **Packet** concept. A **Packet** consists of:

1. The **Packet Header (12 bytes).** This area is defined by a **packet prefix** (3 bytes), the **controller type** (1 byte)**,** the **controller ID** (1 byte), a **reserved area** (3 bytes), the **packet body length** (2 bytes) and **operation code** (2 bytes).

2. The **Packet Body ('N' bytes).** This variable length area contains ASCII characters and binary data representing addresses, values and other parameters.

3. The **Packet Checksum (1 byte).** Each byte of the **Packet** beginning with the **controller type** byte to and including the last byte of the **Packet Body** are summed. The **Packet Checksum** is equal to '0' minus the least significant byte of this value.

The host must initiate all serial communications as if the ESC is a passive device with respect to Packet Protocol.

The **Packet Header**, **Packet Body** and **Packet Checksum** are generated using the protocol described in the remainder of this document.

Data Transfer Protocol

This protocol allows data to be sent between the host computer and the ESC.

In data communications, the protocol defines the rules for the electrical, physical and functional characteristics of the communication link. The protocol contains procedures required to ensure an orderly exchange of information through the link, to and from the executing programs.

The ESC is initially a "passive" device with respect to **Packet** transmission. It will wait for **Packet** requests from other devices and then respond to those requests where appropriate. It is the responsibility of the host to initiate communications with the ESC.

When the ESC receives a **Packet**, it responds to the host with either an acknowledgement *character* (ACK) or a negative acknowledgement *code* (NAK). An ACK *character* is an indication that the **Packet** received was valid. A NAK *code* is an indication that the **Packet** received was invalid or a timeout occurred.

It is the responsibility of the host device to retransmit a **Packet** in the event that it receives a NAK *code* from the ESC.

Communications Error	NAK Code (hex)	Code Description
Address Out of Range	0x10	Tried to access a memory address out-of-range.
Receive Data Length	0x11	Tried to send a packet that's to large.
Bad Data / Flag Type	0x12	Tried to read/write an invalid type.
Request Exceeded Maximum Packet Length	0x13	Tried to request too much data.
Data is Read-Only	0x14	Tried to change a read-only data type.
Bad Checksum	0x15	Error in checksum.
Timeout	0x16	Packet Transmission Stopped Midstream
Bad Header	0x17	Incorrect Packet Header Format
Bad Opcode	0x18	Tried to execute an opcode that is not supported.
Program Load Error	0x1f	Program loaded out of sequence.

Data Transfer Sequence

The following describes a typical sequence of events for the transfer of **Packets** between devices:

1. The host device creates and sends a **Packet** to the ESC.

2. The ESC receives the **Packet**. The **Packet** is examined for a valid **Packet Header**, **Packet Body** and **Packet Checksum**.

3. The ESC will respond with an ACK *character* if the **Packet Header**, **Packet Body** and **Packet Checksum** are correct.

   The ESC will respond with a NAK *code* if a problem was found with the **Packet**. For example, an incorrect **Packet Checksum** was received or a packet data length that exceeds the input buffer size.

4. If the **Packet** from the host is requesting information, the ESC will then retrieve the information, build a return **Packet** and transmit this **Packet** back to the host device.

NOTE

The ESC will timeout and reset the incoming packet protocol state machine in the event the host transmission of a packet stops for more than 2 seconds.

ESC Memory Partitions

ESC memory is partitioned in the following manner:

Program Memory

This is the area containing the users application program. The physical memory type of this area is Flash ROM, which is booted into SDRAM at startup.

The *Program Memory* size is 128 K bytes.

Configuration Memory

This is the area containing the users application configuration. The physical memory type of this area is Flash ROM, which is loaded into SDRAM at startup.

The *Configuration Memory* size is 64 K bytes.

Data Variable Memory – Text / SHORT Integer / LONG Integer / FLOATING Point

This is the area containing the variables used by the application program. All Integer and Floating Point variables are stored here. The physical memory type of this area is NOVRAM.

The *Data Variable Memory* size is 64 K bytes.

Data Constants Memory - SHORT Integer / LONG Integer / FLOATING Point

This is the area containing the constants used by the application program. All Integer and Floating Point constants are stored here. The physical memory type of this area is Flash ROM, which is booted into SDRAM at startup.

The *Data Constants Memory* size is 16 K bytes.

Extended Data Memory - SHORT Integer / LONG Integer

This is the area containing the cam values generated as a result of one or more **calc_cam** commands. The physical memory type of this area is SDRAM.

The *Extended Data Memory* size is 512 K bytes.

Fixed Variable Memory – SHORT Integer / LONG Integer / FLOATING Point

This is another area containing variables used by the application program. This area of memory is not affected by a program RESET or DOWNLOAD. Only Long Variables, Short Variables and Floating Point variables are stored here. This area cannot hold the data defined by the ***begin_data / data / end_data*** construct. The physical memory type of this area is NOVRAM.

The Fixed *Variable Memory* size is 32 K bytes.

Data Memory Addressing

Each *Data Memory* area can be addressed directly using a 32-bit absolute pointer.

Packet Data Structure

Packet Header	Packet Body	Packet Checksum
(12 bytes)	(Variable Length - 496 bytes maximum)	(1 byte)

The Packet Data Structure is comprised of three main sections; the **Header**, the **Packet Body** and the **Checksum**. The maximum size of a packet must not exceed 509 bytes.

Packet Header

"ESC"	Type	ID	Reserved	Packet Body Length	Operation Code
(3 bytes)	(1 byte)	(1 byte)	(3 bytes)	(2 bytes)	(2 bytes)

The first 3 bytes are the **packet prefix** character string. It will always be the 3-byte character string **"ESC"** and is used to signal the beginning of a **Packet**.

The next byte is the **controller type**. This is a fixed value of [01] for the ESC controller.

The next byte is the **controller id.** This is used to identify a controller on the multi-drop RS485 interface. This value must be in the range of 1 to 255. (The ID is ignored on RS232 Transmissions).

The next 3 bytes are **reserved** for future use.

The next 2 bytes define the **packet body length** (the **Packet Header** and the **Packet Checksum** are not included). The length is given in bytes and may be an even or odd value. The most significant byte of the length is transmitted first.

The next 2 bytes define the **operation code**. Valid **operation codes** are listed below.

Operation Codes

Code	Operation	Passed	Returned
01	Read Block Data (RDBLKDATA)	Data Type, Starting Address, # of Values to Read	NAK *code* or ACK *char* followed by Data
02	Write Block Data (WRBLKDATA)	Data Type, Starting Address, Values to Write	NAK *code* or ACK *char*
03	Request ESC Status (RQESCSTAT)	None	NAK *code* or ACK *char* followed by Data
04**	Set Flag (SETFLAG)	Flag Type, Logical Flag #	NAK *code* or ACK *char*
05**	Clear Flag (CLRFLAG)	Flag Type, Logical Flag #	NAK *code* or ACK *char*
06**	Read Flag (RDFLAG)	Flag Type, Logical Flag #	NAK *code* or ACK *char* followed by Data
07**	Read Flag Group (RDFLAGGROUP)	Flag Type	NAK *code* or ACK *char* followed by Data
08	AutoTune Device (AUTOTUNE)		TBD
09	Jog (JOG)		TBD
10	Index (INDEX)		TBD
11	Stop Motion (STOPMOTION)		TBD
12	Write IDN (WRIDN)		TBD
13	Read IDN (RDIDN)		TBD
14	Read Device Information (RDDEVINFO)		TBD
15	Stop Program (STOPPRG)	None	NAK *code* or ACK *char*
16	Reset Program (RESETPRG)	None	NAK *code* or ACK *char*
17	Start Program (STARTPRG)	None	NAK *code* or ACK *char*
18	Set AutoStart (SETAUTOSTART)	None	NAK *code* or ACK *char*
19	Clear AutoStart (CLRAUTOSTART)	None	NAK *code* or ACK *char*
20	Read Program Information (RDPRGINFO)		NAK *code* or ACK *char* followed by Data
21	Read Controller Info. (RDCNTRLINFO)		NAK *code* or ACK *char* followed by Data
22	DeviceNet Status (DNET_STATUS)		TBD
23	O Scope (O_SCOPE)		TBD
24	Bring Loop Up (LOOP_UP)		TBD

Code	Operation	Passed	Returned
25	Bring Loop Down (LOOP_DOWN)		TBD
26	Read Watch Data (RDWATCHDATA)		NAK *code* or ACK *char* followed by Data
27	Set Flag Addressable (SET_FLAGA)	Device Number, Flag Type, Physical Flag Number	NAK *code* or ACK *char*
28	Clear Flag Addressable (CLR_FLAGA)	Device Number, Flag Type, Physical Flag Number	NAK *code* or ACK *char*
29	Read Flag Addressable (RD_FLAGA)	Device Number, Flag Type, Physical Flag Number	NAK *code* or ACK *char* followed by Data
30	Read Flag Group Addressable (RD_FLAGA_GROUP)	Device Number, Flag Type, Number of Bytes	NAK *code* or ACK *char* followed by Data
31	Read Event Information (RDEVENTINFO)	None	NAK *code* or ACK *char* followed by Data
32	Read Nodes Status (RDNODESSTAT)	None	NAK *code* or ACK *char* followed by Data
33	Read PMC Option Config	Slot Number	NAK *code* or ACK *char* followed by Data
34	Write PMC Option Config	Slot Number, Config Data	NAK *code* or ACK *char*
35	Read Error Log (RDERRORLOG)	None	NAK *code* or ACK *char* followed by Data
36	Clear Fixed Memory (RDFIXEDMEM)	None	NAK *code* or ACK *char*
66	Write Program Information (WRPRGINFO)		NAK *code* or ACK *char*
68	Request Trace Status (RQTRCSTAT)	(Reserved Value)	NAK *code* or ACK *char* followed by Data
69	Start Trace (STARTTRC)	(Reserved Value)	NAK *code* or ACK *char*
70	Stop Trace (STOPTRC)	(Reserved Value)	NAK *code* or ACK *char*
71	Write Trace Setup (WRTRCSETUP)	Mode, Trace Enable, Trace Trigger, Lines to Trace, Var 1 Addr, Var 2 Addr, Var 1 Type, Var 2 Type	NAK *code* or ACK *char*
72	Read Trace Lines (RDTRCLINES)	(Reserved Value)	NAK *code* or ACK *char* followed by Data
73	Read Trace Variable 1 (RDTRCVAR1)	(Reserved Value)	NAK *code* or ACK *char* followed by Data
74	Read Trace Variable 2 (RDTRCVAR2)	(Reserved Value)	NAK *code* or ACK *char* followed by Data

** These operation codes have been replaced.

Controller Data / Flag Types

Type	Description	R/W	Device Address
1.	Emerald Controller Status	read only	no
2.	Sercos Device Status	read only	yes
3.	Sercos Device Input	read only	yes
4.	Sercos Device Output	read / write	yes
5.	Timer	read only	no
6.	User Flag	read / write	no
7.	Global S/W PLS	read only	no
8.	Global H/W PLS	read only	no
9.	Local H/W PLS	read only	yes
10 - 49.	Not used		
50.	Data – SHORT	read / write	no
51.	Data – LONG	read / write	no
52.	Data – FLOAT	read / write	no
53.	Data – Text	read / write	no
54.	Data – SHORT Extended Memory	read / write	no
55.	Data – LONG Extended Memory	read / write	no

Packet Body

Operation Data
(Variable Length)

The variable length **Packet Body** is next. The maximum length of the operation data is 496 bytes.

Packet Checksum

Checksum Value
(1 byte)

Following the **Packet Body**, there is a one-byte field, which contains the **Packet Checksum** for the **Packet**. Each byte of the **Packet** beginning with the **controller type** byte to and including the last byte of the **Packet Body** are summed. The **Packet Checksum** is equal to '0' minus the least significant byte of this value.

General Serial Command Format

The ESC will accept two basic types of serial commands:

1 - The first type of command is a simple transfer of data from a host device to the ESC. Once the ESC receives this type of **Packet**, it simply responds to the host with an ACK *character* or NAK *code*. Whatever data or command was received is processed by the ESC.

2 - The second type of command is a request for data from a host device to the ESC. Once the ESC receives this type of **Packet**, it responds to the host with an ACK *character* or NAK *code*.

After responding with an ACK *character*, the ESC will process the request, construct a **Packet** of information to be sent back to the host device, and then transmit that **Packet**.

Each serial command consists of an **Operation Code**, optionally followed by one or more parameters. The complete command must be formatted into a **Packet** by defining the **Packet Header**, followed by the **Packet Body** and the **Packet Checksum.**

If the command is one that requires a response from the ESC, the ESC will respond with a similar **Packet Header** along with the requested data.

In the format description for each **Packet**, [Header] represents the **Packet Header**, and [Checksum] represents the **Packet Checksum**. The brackets themselves are for purposes of clarity and are NOT part of the actual **Packet**.

All numbers in binary format are most significant byte first, followed by least significant byte(s).

The RDBLKDATA Command (Operation Code 01)

The RDBLKDATA serial command reads a group of data bytes from the ESC. Each data value is stored sequentially in the ESC Data Area beginning at the address specified in the RDBLKDATA command.

Packet sent to the ESC:

[Header][Address][#Bytes][Checksum]

Address: A 4-byte value indicating the absolute address within the Memory Data Area where the values will be read. If the region addressed (up to the end of the block) is outside the boundary of the configured data memory then a NAK is returned. The value for *Address* can be determined from the Data Area of the program Symbol File "<program name>.syf".

#Bytes: A 2-byte value representing the number of bytes to be read. Byte 1 will be read from *Address[0],* Byte 2 from *Address[1],* etc. The minimum number of bytes that can be read is 1 byte. The maximum number of bytes that can be read at one time is 494 bytes. If more than 494 bytes are needed, multiple RDBLKDATA commands can be used.

The ***Packet Body Length*** is 6 bytes.

***Packet returned by the ESC*:**

[Header][Byte 1][Byte 2]...[Byte N][Checksum]

Byte1 … N: These represent the byte values for the data to be read. Data bytes read from the ESC are ordered big endian, where the highest order byte is first.

The ***Packet Body Length*** is 'N' bytes. The maximum ***Packet Body Length*** is 494 bytes.

The WRBLKDATA Command (Operation Code 02)

The WRBLKDATA serial command transmits a group of data bytes to the ESC. Each data byte is stored sequentially in the ESC Data Area beginning at the address specified in the WRBLKDATA command.

Packet sent to the ESC:

[Header][Address][#Bytes][Byte 1][Byte 2]...[Byte N][Checksum]

Address: A 4-byte value indicating the absolute address within the Memory Data Area where the values will be written. If the region addressed (up to the end of the block) is outside the boundary of the configured data memory then a NAK is returned. The value for *Address* can be determined from the Data Area of the program Symbol File "<program name>.syf".

#Bytes: A 2-byte value representing the number of bytes to be written. Byte 1 will be stored at *Address[0]*, Byte 2 at *Address[1*], etc. The minimum number of bytes that can be written is 1 byte. The maximum number of bytes that can be written at one time is 490 bytes. If more than 490 bytes are needed, multiple WRBLKDATA commands can be used.

NOTE**:** The WRBLKDATA commands cannot write to the ***Constants Area.***

Byte1 … N: These represent the bytes of data to be written. The maximum number of bytes is 490. Data bytes written to the ESC should be ordered big endian, where the highest order byte is first.

The ***Packet Body Length*** is 6 + 'N' bytes. The maximum ***Packet Body Length*** is 496 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RQESCSTAT Command (Operation Code 03)

The RQESCSTAT serial command requests the ESC Controller to transmit its status bits as represented in the table below.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

[Header][Controller Status][Reserved 1][Checksum]

Controller Status: A 4-byte value representing the *Controller Status* bits. The *Controller Status* bits definition is shown below.

Reserved: A 10-byte area not currently used.

The ***Packet Body Length*** is 14 bytes.

Controller Status Bits

Bit	Definition
0	RING PHASE 1.
1	RING PHASE 2.
2	RING PHASE 3.
3	RING UP.
4	PROGRAM RUNNING.
5	SYSTEM RESET.
6	AUTOSTART ENABLED (PROGRAM STARTS ON POWER-UP).
7	BAD PROGRAM ARGUMENT DETECTED.
8	BAD PROGRAM ADDRESS DETECTED.
9	BAD FLASH MEMORY.
10	spare - not used.
11	LOADING PROGRAM.
12	LOADING PROGRAM ERROR.
13	BAD OPCODE (PROGRAM COMMAND) DETECTED.
14	STACK OVERFLOW ON SUBROUTINE CALLS.
15	STACK UNDERFLOW ON SUBROUTINE RETURN FROM CALL.
16	RING ENABLED.
17	CALCULATING.
18	ON ERROR ENABLED.
19	EVENTS ENABLED.
20	CAN BUS FAULT.
21	CAN BUS ENABLED.
22	CAN BUS POWER.
23	DEVICE NET SCANNER UP.

The SETFLAG Command (Operation Code 04)

This operation code has been preceded by SET_FLAGA(operation code 27).

The SETFLAG serial command can be used to turn on a specified input, output or to set a specified user flag.

Packet sent to the ESC:

[Header][Flag Type][Flag Number][Checksum]

Flag Type: A 2-byte value indicating the *Flag Type* to be set.

Flag Number: A 2-byte value indicating the logical *Flag Number* to be set.

The ***Packet Body Length*** is 4 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The CLRFLAG Command (Operation Code 05)

This operation code has been preceded by CLR_FLAGA(operation code 28).

The CLRFLAG serial command can be used to turn off a specified input, output or to clear a specified user flag.

Packet sent to the ESC:

[Header][Flag Type][Flag Number][Checksum]

Flag Type: A 2-byte value indicating the *Flag Type* to be set.

Flag Number: A 2-byte value indicating the logical *Flag Number* to be set.

The ***Packet Body Length*** is 4 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RDFLAG Command (Operation Code 06)

This operation code has been preceded by RD_FLAGA(operation code 29).

The RDFLAG serial command can be used to read the state of the controller status flags, device status flags, inputs, outputs, timers, user flags, global S/W PLS, global H/W PLS, and local H/W PLS flags.

Packet sent to the ESC:

[Header][Flag Type][Flag Number][Checksum]

Flag Type: A 2-byte value indicating the *Flag Type* to be set.

Flag Number: A 2-byte value indicating the logical *Flag Number* to be set.

The ***Packet Body Length*** is 4 bytes.

***Packet returned by the ESC*:**

[Header][Flag State][Checksum]

Flag State: A 2-byte value indicating the state of the *Flag Number* read. If the flag is SET/ON, *Flag State* will be 1. If the flag is CLEAR/OFF, *Flag State* will be 0.

The ***Packet Body Length*** is 2 bytes.

The RDFLAGGROUP Command (Operation Code 07)

This operation code has been preceded by RD_FLAGA_GROUP(operation code 30).

The RDFLAGGROUP serial command can be used to read the status of all ESC status flags, timers, user flags, global S/W PLS with a single instruction.

Packet sent to the ESC:

[Header][Flag Type][Checksum]

Flag Type: A 2-byte value indicating the *Flag Type* to be read.

The ***Packet Body Length*** is 2 bytes.

***Packet returned by the ESC*:**

[Header][Byte 1][Byte 2]...[Byte N][Checksum]

Byte 1 ... N: These are 8 bit unsigned values representing the state of the requested *Flag Types*. The flag states are returned as follows:

BYTE	BIT #	LOGICAL FLAG #
1	7 - 0	7 - 0
2	7 - 0	15 - 8
3	7 - 0	23 - 16
4	7 - 0	31 - 24
.	.	.
.	.	.
.	.	.
.	.	.
N	7 - 0	((N*8) – 1) – ((N*8) - 8)

The ***Packet Body Length*** is the length in bytes of the *Flag Type* requested.

Flag Types

Description	# Bytes
Emerald Controller Status	4
Timers	2
User Flags	8
Global S/W PLS	2

/****** Packet Structure Still To Be Determined *********/**
The AUTOTUNE Command (Operation Code 08)

/****** Packet Structure Still To Be Determined *********/**
The JOG Command (Operation Code 09)

/****** Packet Structure Still To Be Determined *********/**
The INDEX Command (Operation Code 10)

/****** Packet Structure Still To Be Determined *********/**
The STOPMOTION Command (Operation Code 11)

/****** Packet Structure Still To Be Determined *********/**
The WRIDN Command (Operation Code 12)

/****** Packet Structure Still To Be Determined *********/**
The RDIDN Command (Operation Code 13)

/****** Packet Structure Still To Be Determined *********/**
The RDDEVINFO Command (Operation Code 14)

The STOPPRG Command (Operation Code 15)

The STOPPRG serial command is used to stop the execution of the program currently running in the controller.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RESETPRG Command (Operation Code 16)

The RESETPRG serial command will erase the program currently stored in the controller memory. The program must be stopped before issuing the RESETPRG command.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The STARTPRG Command (Operation Code 17)

The STARTPRG serial command will cause the program currently loaded in the controller to begin executing.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The SETAUTOSTART Command (Operation Code 18)

The SETAUTOSTART serial command will set a status bit in the controller so that every time the power is cycled on the controller the program will automatically begin execution.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0 byte.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The CLRAUTOSTART Command (Operation Code 19)

The CLRAUTOSTART serial command will clear a status bit in the controller so that every time the power is cycled on the controller the program will not begin execution.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RDPRGINFO Command (Operation Code 20)

The RDPRGINFO serial command is used to retrieve the details of the Program Area, Data Areas and the System Configuration.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

[Header][Program Info Structure][Reserved][Checksum]

***Program Info Structure*:** A 240 byte String containing detailed information about the program.

Filename	-	32 Byte String
Date	-	12 Byte String
Time	-	12 Byte String
Version	-	16 Byte String
Reserved1	-	24 Byte String
Program Area Org.	-	Long Integer
Constant Area Org.	-	Long Integer
Variable Nonvolatile Ram Org	-	Long Integer
Extended Area Org	-	Long Integer
Configuration Area Org		Long Integer
Reserved2	-	40 Byte String
Actual Program Size	-	Long Integer
Program Size	-	Long Integer
SHORT Integer Equate Size	-	Long Integer
LONG Integer Equate Size	-	Long Integer
FLOAT Equate Size	-	Long Integer
SHORT Integer Constant Size	-	Long Integer
LONG Integer Constant Size	-	Long Integer
FLOAT Constant Size	-	Long Integer
SHORT Integer Variable Size	-	Long Integer
LONG Integer Variable Size	-	Long Integer
FLOAT Variable Size	-	Long Integer
Text Variable Size	-	Long Integer
SERCOS Cfg Area Size	-	Long Integer
DeviceNet Cfg Area Size	-	Long Integer
Prg Parameter Cfg Area Size	-	Long Integer
Reserved3	-	20 Byte String
Program Check Sum	-	Long Integer
Reserved	-	240 Byte String

The ***Packet Body Length*** is 480 bytes.

The RDCNTRLINFO Command (Operation Code 21)

The RDCNTRLINFO serial command is used to retrieve the details of the Controllers present hardware and software configuration.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

[Header][Controller Info Structure][Checksum]

***Controller Info Structure*:** Data structure of 248 bytes.

Firmware Number/Rev String	-	12 Byte String
DeviceNet Idenity (8 Byte Structure)		
DeviceNet Vendor ID	-	2 Byte Value
DeviceNet Product Type	-	2 Byte Value
DeviceNet Product Code	-	2 Byte Value
DeviceNet Product Rev Major	-	1 Byte Value
DeviceNet Product Rev Minor	-	1 Byte Value
Controller/DeviceNet Serial Number	-	4 Byte Value
Spare	-	128 Bytes
Powerup Count	-	4 Byte Value
Last Error Exception		
Status Register	-	8 Byte Value
Cause Register	-	8 Byte Value
Exception Program Counter	-	8 Byte Value
Bad Virtual Address Register	-	8 Byte Value
Error EPC Register	-	8 Byte Value
Cache Error Register	-	8 Byte Value
MacroProgram Status	-	8 Byte Value
Power Count	-	4 Byte Value
MacroProgram Instruction Ptr	-	4 Byte Value
Over Temperature Count	-	4 Byte Value
Reserved bytes	-	24 bytes

The ***Packet Body Length*** is 248 bytes.

/****** Packet Structure Still To Be Determined *********/**
The DNET_STATUS Command (Operation Code 22)

/****** Packet Structure Still To Be Determined *********/**
The O_SCOPE Command (Operation Code 23)

/****** Packet Structure Still To Be Determined *********/**
The LOOP_UP Command (Operation Code 24)

/****** Packet Structure Still To Be Determined *********/**
The LOOP_DOWN Command (Operation Code 25)

The RDWATCHDATA Command (Operation Code 26)

The RDWATCHDATA serial command reads a group of data bytes from the ESC. Each data value will be read from the ESC based on the type and the Address/Logical Number.

Packet sent to the ESC:

[Header][Type 1]…[Type 16][Address/Logical 1]…[Address/Logical 16][Checksum]

Type 1 … 16: 16 2-byte values indicating the type to be read. *Type 1* corresponds to *Address1*.

Addr 1 … 16: 16 4-byte values representing the address or logical number of the desired type. *Address 1* corresponds to *Type 1*.

The ***Packet Body Length*** is 96 bytes.

***Packet returned by the ESC*:**

[Header][Value 1]...[Value 16][Checksum]

Value 1 … 16: 16 8-byte values representing the data or flag state of the corresponding *Type*.

The ***Packet Body Length*** is 128 bytes.

The SET_FLAGA Command (Operation Code 27)

The SET_FLAGA serial command can be used to set (enable) all flags, includes network addressing capability; i.e. controllers SERCOS Devices.

Packet sent to the ESC:

[Header][Device Number][Flag Type][Physical Flag Number][Checksum]

Device Number: A 2-byte value indicating the logical device number of the device, zero if not needed for desired *Flag Type*.

Flag Type: A 2-byte value indicating the *Flag Type* to be set.

Physical Flag Number: A 2-byte value indicating the *Physical Flag Number* to be set.

The ***Packet Body Length*** is 6 bytes.

***Packet returned by the ESC*:**

Typical response is an ACK. Error response 0x65 will be returned for read only *Flag Types*.

The CLR_FLAGA Command (Operation Code 28)

The CLR_FLAGA serial command can be used to clear (disable) all flags, includes network addressing capability; i.e. controllers SERCOS Devices.

Packet sent to the ESC:

[Header][Device Number][Flag Type][Physical Flag Number][Checksum]

Device Number: A 2-byte value indicating the logical *Device Number* of the device, zero if not needed for desired *Flag Type*.

Flag Type: A 2-byte value indicating the *Flag Type* to be cleared.

Physical Flag Number: A 2-byte value indicating the *Physical Flag Number* to be cleared.

The ***Packet Body Length*** is 6 bytes.

***Packet returned by the ESC*:**

Typical response is an ACK. Error response 0x65 for read only *Flag Types*.

The RD_FLAGA Command (Operation Code 29)

The RD_FLAGA serial command can be used to read all flags, includes network addressing capability; i.e. controllers SERCOS Devices.

Packet sent to the ESC:

[Header][Device Number][Flag Type][Physical Flag Number][Checksum]

Device Number: A 2-byte value indicating the logical *Device Number* of the device, zero if not needed for desired *Flag Type*.

Flag Type: A 2-byte value indicating the *Flag Type* to be read.

Physical Flag Number: A 2-byte value indicating the *Physical Flag Number* to be read

The ***Packet Body Length*** is 6 bytes.

***Packet returned by the ESC*:**

[Header][Device Number][Flag Type][Physical Flag Number][Flag State][Checksum]

Device Number: A 2-byte value indicating the logical *Device Number* of the device, zero if not needed for desired *Flag Type*.

Flag Type: A 2-byte value indicating the *Flag Type* read.

Physical Flag Number: A 2-byte value indicating the *Physical Flag Number* read

Flag State: A 2-byte value indicating the state of the *Flag Number* read. If the flag is SET/ON, *Flag State* will be 1. If the flag is CLEAR/OFF, *Flag State* will be 0.

The ***Packet Body Length*** is 8 bytes.

The RD_FLAGA_GROUP Command (Operation Code 30)

The RD_FLAGA_GROUP serial command can be used to read all flags, includes network addressing capability; i.e. controllers SERCOS Devices.

Packet sent to the ESC:

[Header][Device Number][Flag Type][Number of Bytes][Checksum]

Device Number: A 2-byte value indicating the logical *Device Number* of the device whose status flags are to be read, zero if not needed for desired *Flag Type*.

Flag Type: A 2-byte value indicating the type of device flag to be read.

All Flag types VALID, however *Device Number* is only used for flag types 2, 3, and 4. For other flag types *Device Number* should be zero.

Number of Bytes: A 2-byte value indicating the *Number of Bytes* to read. This value is typically of a range from 1-16 bytes. The controller's maximum packet length of 512 total bytes should not be exceeded, if so the controller will respond with a error NAK value of 0x55. Requesting more than the available number of bytes in any *Flag Type* will return data from the adjacent flag type organized in the controller's memory.

The ***Packet Body Length*** is 6 bytes.

***Packet returned by the ESC*:**

[Header][Device Number][Flag Type][Number of Bytes][Byte 1][Byte 2]...[Byte N][Checksum]

Device Number: A 2-byte value indicating the logical *Device Number* of the device whose status flags are to be read.

Flag Type: A 2-byte value indicating the type of device flag to be read.

All flag types VALID, however device number is only used for flag types 2, 3, and 4. For other flag types, *Device Number* should be zero.

Number of Bytes: A 2-byte value indicating the *Number of Bytes* to read. This value is typically of a range from 1-16 bytes. The controller's maximum packet length of 512 total bytes should not be exceeded, if so the controller will respond with a error NAK value of 0x55. Requesting more than the available number of bytes in any *Flag Type* will return data from the adjacent flag type organized in the controller's memory.

Byte 1 ... N: These are 8-bit unsigned values representing the state of the requested flags. The flag states are returned as follows:

Byte 1 Bit 0 is flag 1……..…Bit 7 is flag 8
Byte 2 Bit 0 is flag 9……..Bit 7 is flag 16
Byte 3 Bit 0 is flag 17……Bit 7 is flag 24
.
.
Byte N Bit 0 is flag (((N-1) x 8) +1)…Bit 7 is flag (N x 8)

The ***Packet Body Length*** is 6 + 'N' bytes.

The RDEVENTINFO Command (Operation Code 31)

The RDEVENTINFO serial command is used to retrieve the status of the Events. Which events are enabled and the status of the event scan mode.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

[Header][Scan Mode][Event State 64 to 33][Event State 32 to 1][Reserved][Checksum]

Scan Mode: A 2-byte value indicating the Scan Mode. A 1 indicates the scan mode is on and 0 indicates the scan mode is off.

Event State 64 to 33: A 4-byte value indicating the state of events 64 to 33.

A 1 in the appropriate bit position indicates that the event is enabled and a 0 indicates that the corresponding event is disabled.

Event State 32 to 1: A 4-byte value indicating the state of events 32 to 1.

A 1 in the appropriate bit position indicates that the event is enabled and a 0 indicates that the corresponding event is disabled.

Reserved: A 8-byte value currently not used.

The ***Packet Body Length*** is 18 bytes.

The RDNODESSTAT Command (Operation Code 32)

The RDNODESSTAT serial command is used to retrieve the status of the DeviceNet Nodes.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

[Header][ESC Address][Reserved][Node 64 Status] … [Node 1 Status][Checksum]

ESC Address: A 1-byte value indicating the MACID of the ESC.

Reserved: 3 reserved bytes.

Node 64 Status to 1: 1-byte values indicating the status of the node.

The ***Packet Body Length*** is 68 bytes.

VALUE	**NAME**
0	SUCCESSS.
1	RESERVED.
2	RESOURCE UNAVAILABLE.
3	RESERVED.
4	PATH SEGMENT ERROR.
5	PATH DESTINATION UNKNOWN.
6	RESERVED.
7	CONNECTION LOST.
8	SERVICE NOT SUPPORTED.
9	INVALID ATTRIBUTE VALUE.
10	ATTRIBUTE LIST ERROR.
11	ALREADY IN REQUESTED MODE/STATE.
12	OBJECT STATE CONFLICT.
13	OBJECT ALREADY EXISTS.
14	ATTRIBUTE NOT SETTABLE.
15	PRIVILEGE VIOLATION.
16	DEVICE STATE CONFLICT.
17	REPLY DATA TO LARGE.
18	RESERVED.
19	NOT ENOUGH DATA.
20	ATTRIBUTE NOT SUPPORTED.
21	TOO MUCH DATA.
22	OBJECT DOES NOT EXIST.
23	RESERVED.
24	NO STORED ATTRIBUTE DATA.
25	STORE OPERATION FAILURE.
26	RESERVED.
27	RESERVED.
28	MISSING ATTRIBUTE LIST ENTRY DATA.
29	INVALID ATTRIBUTE VALUE LIST.
30	RESERVED.
31	VENDOR SPECIFIC ERROR.

32	INVALID PARAMETER.
33-38	RESERVED.
39	UNEXPECTED ATTRIBUTE IN LIST.
40	INVALID MEMBER ID.
41	MEMBER NOT SETTABLE.
42	GROUP 2 ONLY SERVER GENERAL FAILURE.
43-69	RESERVED.
70	DUP MAC FAILURE.
71	SCANNER CONFIG ERROR.
72	DEVICE COMM FAILURE.
73	WRONG DEVICE TYPE.
74	PORT OVERRUN ERROR.
75	NETWORK FAILURE.
76	NO MESSAGE FOR SCANNER.
77	WRONG SIZE DATA.
78	NO SUCH DEVICE.
79	TRANSMIT FAILURE.
80	IN IDLE MODE.
81	IN FAULT MODE.
82	FRAGMENTATION ERROR.
83	SLAVE INIT ERROR.
84	NOT YET INITIALIZED.
85	RECEIVE BUFFER OVERFLOW.
86	DEVICE WENT IDLE.
87	SHARED MASTER ERROR.
88	SHARED CHOICE ERROR.
89	KEEPER FAILED.
90	CAN PORT DISABLED.
91	PORT BUS OFF.
92	PORT POWER OFF.
93	RESERVED.
94	RESERVED.
95	FLASH UPDATE IN PROGRESS.
96	IN TEST MODE.
97	HALTED BY USER COMMAND.
98	FIRMWARE FAILURE.
99	SYTEM FAILURE.
100-207	RESERVED.
208-255	RESERVED FOR OBJECT CLASS AND SERVICE ERRORS.

The RDPMCCFG Command (Operation Code 33)

The RDPMCCFG serial command is used to retrieve the current configuration data for a given PMC slot.

Packet sent to the ESC:

[Header][PMC Slot #][Checksum]

PMC Slot #: 2-byte value indicating the hardware slot of the EMC that the configuration data is requested from.

(1) PMC Slot Number 1
(2) PMC Slot Number 2

The ***Packet Body Length*** is 2 byte

***Packet returned by the ESC*:**

[Header][DeviceID][VendorID][Config Data][Checksum]

PMC Slot #: 2-byte value indicating the hardware slot of the EMC of the returned data.

DeviceID: 2-byte value reflecting the device ID of the PMC Option Card.

VendorID: 2-byte value reflecting the Vendor ID of the PMC Option Card Manufacturer.

Config Data: 64- byte string representing the configuration data for the PMC Option Card. Each card has its own unique Structure for the config data.

The ***Packet Body Length*** is 70 bytes

PMC Option Card	**Variable**	**Description**
Ethernet	TCP/IP Address	{Byte0}.{Byte1}.{Byte2}.{Byte3}
	TCP/IP Subnet Mask	{Byte4}.{Byte5}.{Byte6}.{Byte7}
	TCP/IP Gateway	{Byte8}.{Byte9}.{Byte10}.{Byte11}
	Socket Port Number	2 byte integer Value
	Byte14 .. Byte63	Reserved
Modem	Baud Rate	4 byte integer value representing the baud rate. Valid settings: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400
	Parity	1 byte integer value representing the parity. Valid settings: 0 = None, 1 = Even, 2 = Odd
	Data Bits	1 byte integer value representing the number of data bits. Valid settings: 7 = 7 data bits, 8 = 8 data bits
	Stop Bits	1 byte integer value representing the number of stop bits. Valid settings: 1 = 1 stop bits, 2 = 2 stop bits
	Protocol	1 byte Reserved
	Byte8 .. Byte63	Reserved

The WRPMCCFG Command (Operation Code 34)

The WRPMCCFG serial command is used to write the configuration data for a given PMC slot.

Packet sent to the ESC:

[Header][PMC Slot #]][DeviceID][VendorID][Config Data][Checksum]

PMC Slot #: 2-byte value indicating the hardware slot of the EMC that the configuration data is requested from.

(1) PMC Slot Number 1
(2) PMC Slot Number 2

DeviceID: 2-byte value reflecting the device ID of the PMC Option Card.

VendorID: 2-byte value reflecting the Vendor ID of the PMC Option Card Manufacturer.

Config Data: 64- byte string representing the configuration data for the PMC Option Card. Each card has its own unique Structure for the config data. (See RDPMCCFG for data structures.)

The ***Packet Body Length*** is 70 bytes

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RDERRORLOG Command (Operation Code 35)

The RDERRORLOG serial command is used to retrieve the information stored for the last 8 program error conditions.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

[Header][Error 1 Info] … [Error 8 Info][Checksum]

Error 'N' Info: 2-byte value indicating the enabled status for that error
2-byte value indicating the error number
4-byte value indicating the address of the error routine to be executed
4-byte value indicating the address of the program command in error
2-byte value indicating the address of the program line in error
2-byte value indicating the index to be used with the power up counter
4-byte value indicating the power up counter
4-byte spare
4-byte spare

The returned ***Packet Body Length*** is 224 bytes.

The CLRFIXEDMEM Command (Operation Code 36)

The CLRFIXEDMEM serial command will erase the Fixed Variable Memory area stored in the controller NOV memory. The program must be stopped before issuing the CLRFIXEDMEM command. This command only works with the Emerald EMC-2000 Controller.

Packet sent to the ESC:

[Header][Checksum]

The ***Packet Body Length*** is 0.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The WRPRG Command (Operation Code 64)

The WRPRG serial command serves to initialize the transmission of the macro program and to transmit the first block of program or program data to the ESC Controller. Each data byte is stored sequentially in the ESC Data Area beginning at the address specified in the WRPRG command.

This command can only be used for the first program packet. Additional packets of the program should be transmitted with the WRMOREPRG Command (**Operation Code 65**). The end of program transmission is signified by the transmission of the WRPRGINFO (**Operation Code 66**).

Packet sent to the ESC:

[Header][Address][#Bytes][Byte 1][Byte 2]...[Byte N][Checksum]

Address: A 4-byte value indicating the absolute *Address* within the Memory Data Area where the values will be written. If the region addressed (up to the end of the block) is outside the boundaries of the configured data memory then a NAK is returned. The value for *Address* can be determined from the Data Area of the program Symbol File "<program name>.syf".

#Bytes: A 2-byte value representing the number of bytes to be written. Byte 1 will be stored at Address[0], Byte 2 at Address[1], etc. The minimum number of bytes that can be written is 1 byte. The maximum number of bytes that can be written at one time is 495 bytes. If more than 490 bytes are needed to transmit the program, additional WRMOREPRG commands can be used.

Byte 1 … N: These represent the bytes of data to be written.

The ***Packet Body Length*** is 6 + 'N' bytes. The maximum ***Packet Body Length*** is 496 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The WRMOREPRG Command (Operation Code 65)

The WRMOREPRG serial command serves to transmit additional blocks of program or program data to the ESC Controller. Each data byte is stored sequentially in the ESC Data Area beginning at the address specified in the WRMOREPRG command.

This command cannot be used for the first program packet. The end of program transmission is signified by the transmission of the WRPRGINFO **(Operation Code 66).**

Packet sent to the ESC:

[Header][Address][#Bytes][Byte 1][Byte 2]...[Byte N][Checksum]

#Bytes: A 2-byte value representing the number of *Bytes* to be written. Byte 1 will be stored at Address[0], Byte 2 at Address[1], etc. The minimum number of bytes that can be written is 1 byte. The maximum number of bytes that can be written at one time is 490 bytes. If more than 490 bytes are needed to transmit the program, additional WRITEMOREMACRO commands can be used.

Address: A 4-byte value indicating the absolute *Address* within the Memory Data Area where the values will be written. If the region addressed (up to the end of the block) is outside the boundaries of the configured data memory then a NAK is returned. The value for *Address* can be determined from the Data Area of the program Symbol File "<program name>.syf".

Byte 1 … N: These represent the bytes of data to be written.

The ***Packet Body Length*** is 6 + 'N' bytes. The maximum ***Packet Body Length*** is 496 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The WRPRGINFO Command (Operation Code 66)

The WRPRGINFO serial command is used to define the details of the Program Area, Data Areas and the System Configuration. It also is used to signify to the controller that this is the last packet to be down loaded for the program.

Packet sent to the ESC:

[Header][Program Info Structure][Reserved][Checksum]

***Program Info Structure*:** A 240 byte String containing detailed information about the program.

Filename	-	32 Byte String
Date	-	12 Byte String
Time	-	12 Byte String
Version	-	16 Byte String
Reserved1	-	24 Byte String
Program Area Org.	-	Long Integer
Constant Area Org.	-	Long Integer
Variable Nonvolatile Ram Org	-	Long Integer
Extended Area Org	-	Long Integer
Configuration Area Org		Long Integer
Reserved2	-	40 Byte String
Actual Program Size	-	Long Integer
Program Size	-	Long Integer
SHORT Integer Equate Size	-	Long Integer
LONG Integer Equate Size	-	Long Integer
FLOAT Equate Size	-	Long Integer
SHORT Integer Constant Size	-	Long Integer
LONG Integer Constant Size	-	Long Integer
FLOAT Constant Size	-	Long Integer
SHORT Integer Variable Size	-	Long Integer
LONG Integer Variable Size	-	Long Integer
FLOAT Variable Size	-	Long Integer
Text Variable Size	-	Long Integer
SERCOS Cfg Area Size	-	Long Integer
DeviceNet Cfg Area Size	-	Long Integer
Prg Parameter Cfg Size	-	Long Integer
Reserved3	-	20 Byte String
Program Check Sum	-	Long Integer
Reserved	-	240 Byte String

The ***Packet Body Length*** is 480 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RQTRCSTAT Command (Operation Code 68)

The RQTRCSTAT serial command returns the ESC controller trace status as a 2-byte value.

Packet sent to the ESC:

[Header][Reserved][Checksum]

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 2 bytes.

***Packet returned by the ESC*:**

[Header][Status][Reserved][Checksum]

Status: A 2-byte value defined as follows:
(0) The controller has stopped tracing or was never requested to start tracing.
(1) The trace is in progress but the Trace Enable line has not yet been detected.
(>1) The trace is in progress, the Trace Enable line has been detected but the Trace Trigger line has not been detected.

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 4 bytes.

The STARTTRC Command (Operation Code 69)

The STARTTRC serial command requests the ESC controller to begin tracing program execution as specified in a previous WRTRCSETUP serial command. Program execution (source line numbers) will be placed in the trace buffer until the WRTRCSETUP conditions are met or until a STOPTRC command is received.

Packet sent to the ESC:

[Header][Reserved][Checksum]

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 2 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The STOPTRC Command (Operation Code 70)

The STOPTRC serial command requests the ESC controller to disable the trace feature. Retrieving trace information from the trace buffer may not be reliable, as a result of issuing a STOPTRC command.

Packet sent to the ESC:

[Header][Reserved][Checksum]

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 2 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The WRTRCSETUP Command (Operation Code 71)

The WRTRCSETUP serial command requests the ESC controller to perform a real time trace of the instructions being executed based on the list of parameters below.

Packet sent to the ESC:

[Header][Mode][Trace Enable][Trace Trigger][Lines to Trace] [Var 1 Address] [Var 2 Address][Var 1 Type][Var 2 Type][Reserved][Checksum]

Mode: A 2-byte value defined as follows:
(2) Trace Before - store the line numbers of the instructions executed just ***before*** the Trace Trigger.
(3) Trace About - store the line numbers of the instructions executed just ***before*** and just ***after*** the Trace Trigger.
(4) Trace After - store the line numbers of the instructions executed just ***after*** the Trace Trigger.

Trace Enable: A 2-byte value indicating the line number which, when encountered, will cause the ESC controller to begin loading its' trace buffer and monitoring the Trace Trigger.

Trace Trigger: A 2-byte value indicating the line number which keys the actual trace buffer display based on before, after or about.

Lines to Trace: A 2-byte value indicating the number of lines to be traced. This value is based on the *Var 1 Type* and *Var 2 Type* entries:
240 - if there is no request to monitor *Var 1* or *Var 2*
60 - if the largest *Var Type* to monitor is type FLOAT
120 - if the largest *Var Type* to monitor is type LONG
240 - if the largest *Var Type* to monitor is type SHORT

Var 1 Address: A 4-byte value indicating the address of a variable to be monitored at each trapped trace line.

Var 2 Address: A 4-byte value indicating the address of a variable to be monitored at each trapped trace line.

Var 1 Type: A 1-byte value indicating the type of data (SHORT, LONG, FLOAT) to be monitored at each trapped trace line.

Var 2 Type: A 1-byte value indicating the type of data (SHORT, LONG, FLOAT) to be monitored at each trapped trace line.

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 20 bytes.

***Packet returned by the ESC*:**

***** None, ACK is only response *****

The RDTRCLINES Command (Operation Code 72)

During a trace operation, the ESC Controller stores the line number of each instruction executed into a trace buffer. Each line number corresponds to a line from the program Source File "<program name>.srf". Although the RDTRCLINES command returns a fixed length packet and buffer size, the number of lines in the buffer is dependent on whether a request was made to monitor variables as well as the size of the variables.

Note: The number of lines traced in the buffer can vary between 60, 120 and 240 based on the trace option to monitor variables during trace execution. Depending on the size of the largest data variable to be monitored, the number of traced program lines / variables can be reduced to as few as 60.

Packet sent to the ESC:

[Header][Reserved][Checksum]

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 2 bytes.

***Packet returned by the ESC*:**

[Header][Mode][Enable Line][Trigger Line][Mark][Trace Line 1]...[Trace Line N][Checksum]

Mode: A 2-byte value defined as follows:
(2) Trace Before - store the line numbers of the instructions executed just ***before*** the Trace Trigger.
(3) Trace About - store the line numbers of the instructions executed just ***before*** and just ***after*** the Trace Trigger.
(4) Trace After - store the line numbers of the instructions executed just ***after*** the Trace Trigger.

Enable Line: A 2-byte value indicating the line number which, when encountered, will cause the ESC controller to begin loading its' trace buffer and monitoring the Trace Trigger.

Trigger Line: A 2-byte value indicating the line number which keys the actual trace buffer display based on before, after or about.

Mark: A 2-byte value acting as a pointer into the trace buffer indicating the location of the Trigger Line.

Lines Traced: A 2-byte value indicating the actual number of lines in trace buffer. This value should match the number of Lines to Trace as indicated in the WRTRCSETUP command.

Trace Line 1 … N: This is a 480-byte area which will hold from 60 to 240 traced lines, depending on the requested number of Lines to Trace as indicated in the WRTRCSETUP command.

The ***Packet Body Length*** is 490 bytes.

The RDTRCVAR (1,2) Command (Operation Codes 73,74)

During a trace operation, the ESC Controller will store the value of up to two variables (Variable 1 and/or Variable 2 as defined in the WRTRCSETUP command) as each line is executed and stored in the trace buffer. This gives the user an opportunity to "monitor" the contents of two variables (SHORT, LONG or FLOAT) during program execution.

Note: This is a Trace option and is not mandatory. Up to 240 program lines / variables can be traced if this option is not invoked. Depending on the size of the largest data variable to be monitored, the number of traced program lines / variables can be reduced to as few as 60.

Packet sent to the ESC:

[Header][Reserved][Checksum]

Reserved: A 2-byte value currently not used. Default is 0x0000.

The ***Packet Body Length*** is 2 bytes. The ***Total Packet Length*** is 14 bytes.

***Packet returned by the ESC*:**

[Header][Var Addr][Reserved][Var Type][Data Count][Value 1] … [Value N][Checksum]

Var Addr: A 4-byte value indicating the address of the variable monitored at each trapped trace line.

Reserved: A 1-byte reserved value.

Var Type: A 1-byte value indicating the type of data (SHORT, LONG or FLOAT) monitored at each trapped trace line.

Data Count: A 2-byte value indicating the actual number of data values traced and thus the number of entries in the variable trace buffer. This value should match the number of *Lines to Trace* as indicated in the WRTRCSETUP command.

Value 1 … N: This is a 480-byte area which will hold from 60 to 240 trapped values, depending on the requested number of *Lines to Trace* as indicated in the WRTRCSETUP command.

The ***Packet Body Length*** is 488 bytes.