Headquarters

Unitronics (1989) (R"G) Ltd.
Unitronics Building, Airport City
P.O.B. 300, Ben Gurion Airport, Israel 70100
Tel: + 972 (3) 977 8888   Fax: + 972 (3) 977 8877

Communication with the Vision™ PLC

Document Version: 20/9/04

Contents

Introduction

This document is intended to enable developers to create applications that communicate data between Unitronics' PLCs and other applications.

The Vision™ series offers two formats of data exchange between the PLC and the calling application:

    ASCII format,

    Binary format.

M90/91 controllers support ASCII format only. Binary format is not supported.

While usage of the ASCII format enables the user to exchange homogeneous data types with the PLC, the binary format enables him to either read or write heterogeneous data types from / to it, within a single request. Hence, if the application requires that different type values be included in a single request (such as MB and DW), a binary format must be used.

This document describes the format of the data requests that the calling application can send to the PLC and the PLC message response format.

Communicating with the Vision™ PLC

An application can communicate with the PLC either by serial communication or by TCP. To communicate via TCP (Ethernet), you must add a header to each command as described in Appendix 3: PCOM via Ethernet.

The format of the data exchange is independent of the communication protocol; in either case ASCII and binary formats may be implemented.

Notes
1. Either ASCII or Binary message format may be used; there is no connection between the two types.
2. PLCs of types M90, M91 and Vision 120 do not support the TCP communication protocol.
3. Vision 230, 260, and 280 controllers must have an installed Ethernet port to implement TCP.

ASCII Message Format

ASCII message format enables values to be both read from and written to the PLC.

In this format:

Frames sent to the PLC should start with a slash character '/' (ASCII 47) and should be terminated by <CR> (ASCII 13)

Frames received from the PLC start with '/A' (slash and the letter A) and are terminated by <CR>.

Note   In this document, the combined '/A', is referred to as STX1.

The format of a command sent to the PLC is: <STX><UnitID><CommandCode><Command parameters><CheckSum><ETX>

STX="/"

The UnitID should be two characters long (pad with zeroes if needed).

Note   Note that UnitID ='00' refers to a directly connected PLC.

The checksum (CRC) should be two characters long (pad with zeroes if needed). See Appendix 1 regarding checksum calculation.

Command parameters are command-dependent.

ETX = CR (Carriage Return, ASCII value: 13).

The values to and from the PLC are represented in hexadecimal, (aka 'Hex as ASCII'), and thus enable an easy debug process. For example, the PLC will respond with 06D9 when queried for a Memory Integer register containing a decimal value of 1753.

Note   In the examples below:
CRC represents the check sum value.
Spaces have been added just for clarity. A real space is indicated by <SPACE>.

RE, RA, RB, GS, RT, RM - Read bits

To PLC:

<STX><CC><ADDRESS><LENGTH><CRC><ETX>

From PLC:

<STX1><CC><VALUES><CRC><ETX>

CC (Command Code) values:

"RE" - Read inputs

"RA" - Read outputs

"RB" - Read memory bits

"GS" - Read system bits

"RT" - Read timer scan bits

"RM" - Read counter scan bits

Example (read 5 memory bits from address 32 to 36):

PC: /   01 RB 0020 05 CRC <CR>

PLC: / A 01 RB 10010 CRC <CR>

01 means PLC with UnitID=01

0020 means start Read operation at address 32 (0020 hex)

05 means read 5 values, beginning from the start address.

10010 means bits 32 (20 hex) to 36 (24 hex)

are 1, 0,0,1,0 respectively.

SA, SB, SS - Set bits

To PLC:

<STX><CC><ADDRESS><LENGTH><CRC><ETX>

From PLC:

<STX1><CC><VALUES><CRC><ETX>

CC (Command Code) values:

"SA" - Set outputs

"SB" - Set memory bits

"SS" - Set system bits

Example (set 5 memory bits from address 32 to 36):

PC: /   01 SB 0020 05 10010 CRC <CR>

PLC: / A 01 SB CRC <CR>

01 means PLC with UnitID=01

0020 means start at address 32 (0020 hex).

05 means length of 5 memory bits.

10010 means set bits 32 (20 hex) to 36 (24 hex).

are set to 1,0,0,1,0 respectively.

RW, RNL/D/H/J/F, GF, GT, GP,GX, GY - Read integers (16 and 32 bits)

To PLC:

 <STX><CC><ADDRESS><LENGTH><CRC><ETX>

From PLC:

<STX1><CC><VALUES><CRC><ETX>

> **Note**   When Command Codes are longer than 2 characters, only the first two characters will be included in the response from the PLC.

CC (Command Code) values:

"RW" - Read MIs

"RNL" - Read MLs (*)

"RND" - Read MDWs (*)

"GF" - Read SIs

"RNH" - Read SLs (*)

"RNJ" - Read SDWs (*)

"RNF" – Read MFs (memory floats) (See note about floats) (*)

"GT" - Read Timer's current value

"GP" - Read Timer's preset value

"GX" - Read Counter's current value

"GY" - Read Counter's preset value

* The Command Code from PLC will be "RN"

Example (read 2 memory integers from address 32 to 33):

PC: /   01 RW 0020 02 CRC <CR>

PLC: / A 01 RW 01020304 CRC <CR>

01 means PLC with UnitID=01

0020 means start address of 32 (0020 hex)

02 means length of 2

01020304 means:

Value in addresses 32 (20 hex) is 0102 (hex).

Value in address 33 (21 hex) is 0304

Example (read 2 memory longs from address 32 to 33):

PC: /   01 RNL 0020 02 CRC <CR>

PLC: / A 01 RN 1122334405060708 CRC <CR>

01 means PLC with UnitID=01

0020 means starting address is 32 (0020 hex)

02 means a read vector length of 2

1122334405060708 means:

Value in address 32 (20 hex) is 11223344 (hex).

Value in address 33 (21 hex) is 05060708 (hex)

Example (read a float value from MF (15)

PC: / 01 RNF 000F 01 CRC <CR>

PLC: /A 01 RN E6AE4640 CRC <CR>

01 means PLC with UnitID=01

000F means starting address is 15 (000F hex)

RNF is the command code.

01 means a read vector length of 1

E6AE4640 means:

The return value

The return value represents the float value of 12345.67

> **Note**   The bits are represented (according to IEEE 754 standard) 4640E6AE hence, in order to derive the actual value of the words, they should be transposed. (E6AE 4640 → 4640 E6AE).

SW, SNL/D/H/J, SF - Write integers (16 and 32 bits)

To PLC:

<STX><CC><ADDRESS><LENGTH><CRC><ETX>

From PLC:

<STX1><CC><VALUES><CRC><ETX>

> **Note**   When Command Codes are longer than 2 characters, only the first two characters will be included in the response from the PLC..

CC (Command Code) values:

"SW" - Write MIs

"SNL" - Write MLs (*)

"SND" - Write MDWs (*)

"SNF" – write MFs (*)

"SF" - Write SIs

"SNH" - Write SLs (*)

"SNJ" - Write SDWs (*)

* Command Code from PLC will be "SN"

Example (set values of two memory integers) :

PC: /   01 SW 0020 02 01020304 CRC <CR> // 16 bits

PLC: / A 01 SW <CR>

01 means PLC with UnitID=01

0020 means first register address is 32 (20hex).

02 means the write vector length is 2.

01020304 means that:

value 258 (0102 hex) will be written into address 32 (20 hex).

value 772 (0304 hex) will be written into addresses 33 (21 hex)

Example (set values of two memory longs):

        PC: / 01 SNL 0020 02 1122334405060708 CRC <CR> // 32 bits

        PLC: / A 01 SN <CR>

01 means PLC with UnitID=01

0020 means the start address is 32 (20hex).

02 means write into two registers.

1122334405060708 means:

value of 287454020 (11223344 hex) will be written to address 32 (20 hex)

value of 84281096 (5060708 hex) will be written to address 33 (21 hex)

Note: Data for each register should be indicated explicitly.

RC - Read RTC

To PLC:

 <STX><CC<CRC><ETX>

From PLC:

<STX1><CC><VALUE><CRC><ETX>

CC (Command Code) values:

        "RC" – Read Real Time Clock

Example Read Real Time Clock

        PC: / 01 RC CRC <CR>

        PLC: / A 01 RC 00 30 15 01 29 04 01 CRC <CR>

        The time is: 15:30:00, Sunday (01), 29/04/2001

SC - Set RTC

To PLC:

<STX><CC<CRC><ETX>

From PLC:

<STX1><CC><VALUE><CRC><ETX>

CC (Command Code) values:

" SC " – Set Real Time Clock

Example: Set Real Time Clock

PC: /   01 SC 00 30 15 01 29 04 01 CRC <CR>

PLC: / A 01 SC <CR>

Set time to 15:30:00, Sunday (01), 29/04/2001

Binary Message Format

This section shows the binary format used for all PC data requests and PLC responses.

Each Binary message contains the following sections:

- Header

- Details

- Footer

The format of each section is identical for both data requests and responses; the contents of the sections vary according to the command number and the data that is contained by the message.

Header

The header size is 24 bytes.

Byte	Purpose	Remarks
0 – 5	STX	"/_OPLC"
6	Destination CANbus network ID and RS485	See Note 2
7	Reserved	Should always be 254 (FE hex)
8	Reserved	Should always be 1
9-11	Reserved	Should always be 0
12	Command Number	
13	Reserved	Should always be 0
14-19	Specific for each Command Number	See each command for details
20 - 21	Data length	Number of bytes in the Details section (excluding header and footer).
22 – 23	Header 16-bit checksum	Calculated for bytes 0 to 21 (see Appendix 2: Checksum calculation

Details

This section contains the data that is communicated between the PC and PLC, and therefore varies from command to command.

It is not required by all commands, and may be empty.

Footer

The footer size is 3 bytes.

Byte	Purpose	Remarks
0 – 1	Details section:16 bit checksum	Calculated for Details Section (see Appendix 2: Checksum calculation
2	ETX	"\"

Notes
1. The PLC cannot accept or deliver messages that are longer than 500 bytes.
2. CANbus and RS485 If the PLC functions as a bridge to another PLC in a CANbus network, the destination within that network should be identified with the correct ID in byte #6.
If the PLC is a member of a RS485 multi-drop network, the ID of the destination PLC should reside in byte #6.

PLC Response

The PLC Response contains the same 3 sections as the Request, (header, details and footer).

Notes
1. Within the PLC response bytes 6 and 7 are transposed. See examples for more details.
2. When the PLC responds, it acknowledges the request by adding 80(hex) to the command number (byte 12 within the Header).

Binary Commands

There are 3 possible commands that may be issued using the Binary Format:

- Read Operands
- Read Data Tables
- Write Data Tables

Read Operands: Command # 77

A single message can, via this command, read data from operands. The command number, 77, is contained in byte #12 in the message header; the data requests are contained in the Details section.

Within the Details section, you can include multiple data requests. The data requests may be for different types of operands; the operands may be adjacent to each other (vectorial) or not (non-vectorial).

However, note that a single data request cannot request data for more than one operand type.

Within the Details section, the data requests must be organized as follows:

3. Bit data (MBs, Inputs, etc.)
4. Integer (16-bit)
5. Long/DW/Float (32-bit)

Data Request Structure

Byte	Purpose	Remarks
0 – 1	Number of addresses	Number of operands to be read
2	Operand type	See table below
3	Reserved	Should be 0xFF
4 – 5	1st address	Each 2 bytes comprises a single address
6 – 7	2nd address	
8 – 9	3rd, etc…	

Bytes 18-19 hold the number of data requests in the message.

To read the values of non-vectorial operands, you must include the address of each operand to be read.

To read the values of vectorial operands, you must indicate the address of the first element.

The value of the operand type is in byte #2. The following table shows the values addressed for non-vectorial operands. The size of the returned value for the operand is indicated as well.

To indicate that the operands are vectorial, add 128 (80 hex) to the operand type

Operand Types

Operand Type	Operand size (in bits)	Byte value
MB	1	1
SB	1	2
MI	16	3
SI	16	4
ML	32	5
SL	32	6
MF	32	7
SF	32	8
Input	1	9
Output	1	10
Timer Run Bit	1	11
Counter Run Bit	1	12
DW	32	16
SDW	32	17
Counter Current	16	18
Counter Preset	16	19
Timer Current	32	20
Timer Preset	32	21

Example: Read Operands Data Request

The table below shows how operands may be grouped into data requests.

Bits (non-vectorial)	MB2, MB9, MB25, MB26, MB27
Integers (non-vectorial)	MI0, MI2, MI45, MI56
Longs (non-vectorial)	ML56
Integers (vectorial)	9 MIs starting at MI198

The operands are arranged in data requests according to the following principals:

- One request for MB2, MB9, MB25, MB26, MB27 (note that although the last three MBs do form a vector, they have been requested as single entities and thus are grouped as individual MBs.

- One request for ML56.

- One request for MI0, MI2, MI45, MI56.

- One request for the vector starting from MI198 with a length of 9.

Note that if another vector of MIs is requested, this new vector will constitute an **additional** request, thus increasing the number of requests to 5.

PLC response to the binary request

The details section delivers the data according to the data order within the request: bit, integer, double integer.

> **Note**   the PLC will pad the response with 0's, in order to adjust the results into integer boundaries.

Example 1

Request to Read MB(1)

Message header: 2F 5F 4F 50 4C 43 00 FE 01 00 00 00 4D 00

00 00 00 00 01 00 06 00 F1 FC

Message details: 01 00 01 FF 01 00

Message footer: FE FE 5C

Example 2

Request to Read MB(1-7), MF(15), and MI(1-3)

The operands contain the following values:
MBs 1-7:1,0,1,0,1,1,0

MF15:12345.67

MIs 1-3:123,124,125

Message header:

2F 5F 4F 50 4C 43 00 FE 01 00 00 00 4D 00 00 00 00 00 03 00 22 00 D3 FC

Message details:

07 00 01 FF 01 00 02 00 03 00 04 00 05 00 06 00 07 00 03 00 03 FF 01 00 02 00 03 00 01 00 07 FF 0F 00

Message footer:

BC FC 5C

Response header: 2F 5F 4F 50 4C 43 FE 00 01 01 00 00 CD 00 00 00 00 00 03 00 0C 00 68 FC

Response details: 35 00 7B 00 7C 00 7D 00 40 46 AE E6

Response explanation:

35 00 7B 00 7C 00 7D 00 40 46 AE E6

Byte	Value/s	Interpretation
0	35	110101 bits values
1	00	Padding (for alignment)
2-3	7B 00	Bytes of first integer (=123)
4-5	7C 00	Bytes of second integer (=124)
6-7	7D 00	Bytes of third integer (=125)
8-11	40 46 AE E6	4 bytes of float value

Result footer: 3D FC 5C

The floating point numbers are represented in the PLC as four bytes in the IEEE 754 format

In order to reconstruct the float value, the bytes should be reordered (by transposing each integer's bytes (40 46 AE E6 → 46 40 E6 AE) and then packed into a 4 byte floating point variable.

Example 3

Request to read MB(1),MB(2),MB(3),MB(4),MB(5),MB(6),MI(6),MI(7),MI(8)

Message header: 2F 5F 4F 50 4C 43 00 FE 01 00 00 00 4D 00 00 00 00 00

02 00 1A 00 DB FC

Message details: 06 00 01 FF 01 00 02 00 03 00 04 00 05 00 06 00 03 00

03 FF 06 00 07 00 08   00

Message footer: CB FD 5C

Response header: 2F 5F 4F 50 4C 43 FE 00 01 00 00 00 CD

00 00 00 00 00 02 00 08 00 6D FC

Response details: 40 00 CE EB CE EB CE EB

Result footer: 95 FA 5C

In the example, the bit's value is zero, while the integers are 52971

On the other hand, if values in the PLC are:

1,0,1,0,11   for the MBs, and 1234,4567,12567 for MIs, the result would be:

Response header: 2F 5F 4F 50 4C 43 FE 00 01 00 00 00 CD

00 00 00 00 00 02 00 08 00 6D FC

Response details: 75 00 D2 04 D7 11 17 131

Result footer: 85 FD 5C

Example 4

Request to read continuous region MI13 to MI18

Message header: 2F 5F 4F 50 4C 43 00 FE 01 01 00 00 4D 00 00 00 00 00 01 00

06 00 F0 FC

Message details: 06 00 83 FF 0D 00

Message footer: 6B FE 5C

Response header: 47 5F 4F 50 4C 43 FE 00 01 01 00 00 CD 00 00 00 00 00 01 00

0C 00 6A FC

Response details: 71 00 72 00 73 00 74 00 75 00 76 00

Result footer: 4B FD 5C

Note the operand type (3 or 80h = 83 hex)

Reading and writing from/to Data Tables: Commands #4 & #68

This section describes how to use the binary message format to access the PLC Data Tables. The binary format enables data read / write from/to the Data Tables.

Accessing Data Tables using ASCII format is not supported.

Memory structure of Data Tables

The bytes of the first Data Table are allocated first; the bytes of the next table reside next to them in the memory and so on till the last Data Table.

The data of each Data Table is row-oriented. This means that the data of the first row is allocated and then the data of the last row and so on till the last row.

Setting parameters in the binary message format

The binary format enables access to any portion of a Data Table. The retrieved (or sent) data is in vectorial format (even if the data segments are not continuous addresses in the memory). The following data should be embedded in the binary message:

- Address of the first element to read (within the Data Table address space).

- Number of rows to read.

- Number of bytes to read in each row.

- Number of bytes in the entire row.

Read/Write from Data Table:

Within the message header, the value you set in byte number 12 determines the command; 4=read, 68 =write (44 hex).

The address of the first element (to read from or to write into) must be arranged as a 4 byte value and set into bytes 14,15,16,17 of the message header (byte 14 is the least significant byte).

The address of the first element must be calculated while the first element of the first table resides in address 0.

The other three parameters (number of rows to read, number of bytes to read from a row, table's row length) should be embedded in the message details section.

Address	Purpose	Remarks
0-1	Number bytes to read form each table row.	
2-3	Number of rows to read from Data Table.	
4-7	Number of bytes in the entire row of the Data Table.	A 4 byte representation
8-31	Reserved	Value of 0

If the purpose of the request is to write data into the Data Table, then this data should start at the 32nd address.

To calculate length, use the following table, which shows the number of bytes each Data Table column requires.

Column type	Number bytes required
Boolean	1 byte per 8 elements
Byte	1 byte per element
Integer	2 bytes per element
Unsigned integer	2 bytes per element
Long	4 bytes per element
Unsigned long	4 bytes per element
String	1 byte per character

Example: Accessing Data Tables

An example of request construction is shown in this section.

Read request definition

Assume two tables are defined in the PLC:
Table1:

- 10 rows

- 3 columns (integer, integer, 20 char string)

- the second column is defined as 'part of the project' and hence does not reside in the RAM space.

Table2:

- 7 rows

- 3 columns (array of 4 bytes, integer, long)

In the example, we want to read a portion of Table 2; 6 cells from columns 1,2 and rows 4,5,6. The following table illustrates Table 2 and the desired cells' data.

Row Number	Col #0 Contains array of 4 bytes (4 bytes/cell)	Col #1 contains an integer value (2 bytes/cell)	Col #2 contains a long value (4 bytes/cell)
0			
1			
2			
3			
4		X	X
5		X	X
6		X	X

In the example, we want to read the data of the cells marked with X.

The first stage is to calculate the address of the first element to be read.

This address is calculated by adding the table's address to the offset of the element within the table.

Table1 precedes our table, and spans over 220 bytes (22 bytes per row X 10 rows). Remember that Table 1 contains a column that has been defined as 'part of the project' and it is excluded from our calculations.

Table 2 is composed of rows that occupy 10 bytes/row. There are 4 whole rows above our element (= 40 bytes), and 4 bytes preceding it within the row. This means that the first element that we have to read is located at an offset of 44 bytes from the beginning of Table 2. The address of the first element in the 'Data Tables' address space is 240 +44 = 264.

The other three arguments are derived easily from Table 2:

The number of rows to read is clearly 3.

Number of bytes to read from each row is 6.

Number of bytes in the entire row is 10.

In order to read the indicated elements from the Data Table, the binary request should be arranged as following:

Message header section:

bytes	Values
0-5	"/_OPLC"
6	ID (CANbus or RS485)
7	254
8	1
9-11	0
12	(4 for read, 68 for write)
13	0
14	8 (lower byte of address)
15	1 (2nd byte of address)
16	0 (3rd byte of address)
17	0 (4th byte of address)
18-19	0
20	32 (length of detail section 1st byte)
21	0
22	Checksum low byte
23	Checksum high byte

Note that compared to the 'ordinary' binary request, the changes are minor:

(bytes 12, 14, 15, 16, 17) and the value of byte 20 is always 32 (since in the read request it is fixed length of 32). In our example the value of byte 12 will be 4 (read request).

The request detail section will be as following:

Bytes	Value
0-1	6
2-3	3
4-7	10
8-31	Reserved should be 0

The structure of the footer section is not changed.

The result: the delivered result will contain an exact copy of the appropriate section of the Data Table address space.

<u>Write request definition</u>

The write request is very similar to the read request. Of course, the data to write should be set and appended to the details section (starting at address 32).

The header section should be changed in two respects:

The value of byte 12 should be set to 68, and the value of bytes 20-21 in the header should be adjusted to the details length.

The data that will be appended to the details section have to be exactly as it would be in the PLC, since the PLC performs a simple data copy.

Appendix 1: Binary Message Format – a complex example

The following shows how to read data from:

MB78

MB34 to MB39

Inputs 7 to 15

Input 65

Timer 34

Timers 42 to 47

Timer 129

MI17

MI19

MI1022 to MI1027

ML13

DW12 to DW14

Counter15

Counter6 to Counter9

MF6

MF9 to MF12

The requests for the timers and the counters will contain their bit values, as well as the current and preset values.

The request header should be:

2F 5F 4F 50 4C 43 00 FE 01 00 00 00 4D 00 00 00 00 00 16 00 8C 00 55 FC

The request details and footer should be (note that for the sake of clarity, the stream is broken and explanations are embedded):

01 00 (one address)

01   (MB operand type)

FF   (Reserved)

4E 00 (address 78)

06 00 (six addresses)

81   (MB operand type continuous)

FF   (Reserved)

22 00 (address 34)

09 00 (nine addresses)

89   (Input type operand continuous)

FF   (Reserved)

07 00 (address 7)

01 00 (one operand)

09   (Input type operand)

FF   (Reserved)

41 00 (address 65)

02 00 (2 operands)

0B   (Timer operand type)

FF   (Reserved)

22 00 (Address 34)

81 00 (address 129)

06 00 (six operands)

8B   (Timer operand type Continues)

FF   (Reserved)

2A 00   (address 42)

01 00 (one address)

0C   (Counter type operand)

FF   (Reserved)

0F 00 (address 15)

04 00 (four operands)

8C   (Counter type operand continuous)

FF   (Reserved)

06 00 (address 6)

02 00 (two operands)

03   (MI operand type)

FF   (Reserved)

11 00 (address 17)

13 00 (address 19)

06 00 (six operands)

83   (MI operand type continues)

FF   (Reserved)

FE 03   (address 1022)

02 00 (two operands)

14   (Timer current value operand type)

FF   (Reserved)

22 00 (address 34)

81 00 (address 129)

02 00 (two operands)

15   (Timer operand type)

FF   (Reserved)

22 00 (address 34)

81 00 (address 129)

06 00 (two operands)

94   (Timer current operand type continues)

FF   (Reserved)

2A 00   (address 42)

06 00 (six operands)

95   (Timer Preset value continues)

FF   (Reserved)

2A 00   (address 42)

01 00 (one operand)

05   (ML operand type)

FF   (Reserved)

0D 00   (address 13)

03 00 (three operands)

90   (DW operand type)

FF   (Reserved)

0C 00   (address 12)

01 00 (one operand)

12   (Counter current operand type)

FF   (Reserved)

0F 00 (address 15)

01 00 (one operand)

13   (Counter preset operand)

FF   (Reserved)

0F 00 (address 15)

04 00 (four operands)

92   (Counter current value operand continues)

FF   (Reserved)

06 00 (address 6)

04 00 (four operands)

93   (Counter preset value operand continues)

FF   (Reserved)

06 00 (Address 6)

01 00 (One operand)

07   (MF operand type)

FF   (Reserved)

06 00 (address 6)

04 00 (four operands)

87   (MF operand type continues)

FF   (reserved)

09 00 (address 9)

9B DE 5C (CRC and ETX)

This request will deliver the following results:

24 byte of the Response header:

5F 4F 50 4C 43 FE 00 01 01 00 00 CD 00 00 00 00 00 16 00 8C 00 D5 FB

The Response details (note that for the sake of clarity, the stream is broken and explanations are embedded). The result footer is appended to the details.

6B 00 00 00 (30 bit values) (The bit request is 30 bits, packed according to the request order)

F9 03 (MI17 value=1017)

FB 03 (MI19 value=1019)

FE 03 (Six MI values from MI1022)

FF 03

00 04

01 04

EA 07

EB 07

32 01 00 00 (T34 Current value)

32 01 00 00 (T129 current value)

F4 01 00 00 (T34 preset value)

F4 01 00 00 (T129 preset value)

32 01 00 00 (six T Current values from T42)

32 01 00 00

32 01 00 00

32 01 00 00

32 01 00 00

32 01 00 00

F4 01 00 00 (six T preset values from T42)

F4 01 00 00

F4 01 00 00

F4 01 00 00

F4 01 00 00

F4 01 00 00

49 F4 10 00 (ML13 value = 1111113)

70 00 00 00 (three DW values from DW12)

71 00 00 00 (DW13 value)

CA 35 3A 42 (DW14 value=11111114)

07 A9 (counter value)

0A 00 (counter preset)

07 A9 (four counter values

07 A9

07 A9

07 A9

0A 00 (four counter preset value)

00 00

00 00

00 00

85 42 33 33 (MF6 value=66.6)

6A C5 3D 82 (four MF from MF9) (value= -3752.14)

5D 47 4D D5 (value=56789.3)

48 40 C3 F5 (value=3.14)

F6 45 33 97 (value=7890.9)

19 E0 5C (result footer CRC + ETX)

Appendix 2: Checksum calculation

A. Generating a checksum for an ASCII request

The checksum is generated by accumulating the ASCII values of the entire message and calculating the 256 modulo value

For example, the message "/01RE002005" will be appended with a CRC value, in which the checksum is calculated **without** the STX (/).

48+49+82+69+48+48+50+48+48+53=543

543 mod 256=31

31 in hexadecimal representation is 1F.

Hence the CRC is "1F" and the entire message will be:

"/01RE0020051F" + <CR>

B. Generating a checksum for a binary request

Calculate the accumulated sum of the bytes in the vector.

Get the value of the sum modulo 65536 (10000 hex)

The CRC is the 2 complement of the result.

If lSum is a variable holding the accumulated sum, then

CRC = (Not (lSum Mod &H10000)) + 1   (VB syntax).

CRC = (~(lSum % 0x10000)) + 1   (C/C++ syntax).

Note that the CRC is a two-byte value (while in ASCII format it is a one-byte value).

Appendix 3: PCOM via Ethernet

In order to communicate via TCP (Ethernet), you must add a 6-byte long header to each command. The header is always in binary format, whether or not the command itself is given in ASCII or Binary Format.

Ethernet Header format

Bytes 0& 1: Transaction identifier.
> This number is assigned by the user. The number is used by the PCOM master to identify which slave is answering a command request.

Byte 2: Select Protocol.
> Identifies the protocol used for the command request. Enter:
> 101 (DEC) for ASCCI
> 102 (DEC) for Binary

Byte 3: Reserved. Leave blank.

Bytes 4&5: Length of transaction.
> Enter the number of bytes required for the command request.

Notes	1.	PLCs of types M90, M91 and Vision 120 do not support the TCP communication protocol.
	2.	Vision 230, 260, and 280 controllers must have an installed Ethernet port
	3.	The header must be added after message has been configured according to the message format. The Ethernet header is like an envelope, and does not affect the message. In addition, the checksums contained in the message and the length declared within the message, do not include the Ethernet header.

The Ethernet header is also irrelevant to the message contained within the PLC response. To check the response, note whether bytes 0-3 in the Ethernet header are identical to the header in the data request. After this has been checked, strip the header away to process the serial message.

Example

Message "/01RE0020051F" + <CR>

Byte	Decimal	ASCII
0	47	/
1	48	0
2	49	1
3	82	R
4	69	E
5	48	0
6	48	0
7	50	2
8	48	0
9	48	0
10	53	5
11	49	1
12	70	F
13	13	<CR.

If the size of this ASCII message is 14 bytes, the Transaction identifier is 1234 (DEC), the Ethernet header should be as follows:

Byte	Decimal	Comment
0	210	Low byte of 1234
1	4	High byte of 1234
2	101	Protocol: ASCII
3	0	Reserved
4	14	Low byte of Length of Transaction
5	0	High byte of Length of Transaction