

SSCP - Shark Slave Communication Protocol

Protocol description

Table of context

1. Purpose	7
2. Definitions	7
3. Document history	8
4. Transport	9
4.1. General notes	9
4.2. TCP/IP	9
4.3. Serial line	9
4.4. Other physical layers	9
4.5. Tunnelling over Linux	9
5. Protocol frames	10
5.1. Basic structure	10
5.2. Error responses	10
5.2.1. Insufficient rights	10
5.2.2. Invalid function	10
5.2.3. Invalid protocol version	11
5.2.4. Command specific error codes	11
5.3. Table of error codes	11
5.3.1. Optional data	13
5.4. Device management and autodetection	13
5.4.1. Get Basic Info	14
5.4.1.1. Request frame format	14
5.4.1.2. Positive response format	14
5.4.1.3. Negative response format	15
5.5. Access privileges	16
5.5.1. Access levels	16
5.5.2. Privileges for commands	16

5.5.3. Access privileges for file access	17
5.5.4. Login	17
5.5.4.1. Request frame format	17
5.5.4.2. Positive response frame format	18
5.5.4.3. Negative response frame format	18
5.5.5. Logout	18
5.5.5.1. Request frame format	18
5.5.5.2. Positive or negative response frame format	19
5.6. File/large binary data transfer	19
5.6.1. Send data	19
5.6.1.1. Initiate send data transmission	19
5.6.1.1.1. Request frame format	20
5.6.1.1.2. Positive response frame format	20
5.6.1.1.3. Negative response frame format	20
5.6.1.2. Send chunk	20
5.6.1.2.1. Request frame format	20
5.6.1.2.2. Positive response frame format	20
5.6.1.2.3. Negative response frame format	21
5.6.1.3. Finish send data transmission	21
5.6.1.3.1. Request frame format	21
5.6.1.3.2. Positive response frame format	21
5.6.1.3.3. Negative response frame format	21
5.6.2. Receive data	21
5.6.2.1. Initiate receive data transmission	22
5.6.2.1.1. Request frame format	22
5.6.2.1.2. Positive response frame format	22
5.6.2.1.3. Negative response frame format	22
5.6.2.2. ENQ - Receive chunk	23

5.6.2.2.1. Request frame format	23
5.6.2.2.2. Positive response frame format	23
5.6.3. Virtual file system structure	23
5.7. Device statistics	24
5.7.1. Get PLC statistics	24
5.7.1.1. Request frame format	24
5.7.1.2. Positive response frame format	24
5.7.2. Get Task statistics	27
5.7.2.1. Request frame format	27
5.7.2.2. Positive response frame format	28
5.7.2.3. Negative response frame format	28
5.7.3. Get Channel statistics	28
5.7.3.1. Request frame format	28
5.7.3.2. Positive response frame format	28
5.7.3.2.1. Endpoint statistics	29
5.7.4. Negative response frame format	29
5.8. Data communication	29
5.8.1. Direct mode	30
5.8.1.1. Read variables directly	30
5.8.1.1.1. Request frame format	30
5.8.1.1.2. Positive response frame format	31
5.8.1.1.3. Negative response frame format	31
5.8.1.2. Write variables directly	31
5.8.1.2.1. Request frame format	31
5.8.1.2.2. Positive response frame format	32
5.8.1.2.3. Negative response frame format	32
5.8.2. File mode	32
5.8.2.1. Read variables directly	32

5.8.2.2. Write variables directly	32
5.8.3. Data format	33
5.8.3.1. Get all values	33
5.8.3.2. Get all changed values	33
5.8.3.3. Set all values request format	33
5.8.3.4. Set some values request format	33
5.9. Time functions	33
5.9.1. Time setup	33
5.9.1.1. Request frame format	33
5.9.1.2. Positive response frame format	33
5.9.1.3. Negative response frame format	34
5.9.2. Time setup extended	34
5.9.2.1. Request frame format	34
5.9.2.2. Positive response frame format	34
5.9.2.3. Negative response frame format	35
6. Appendix	35
6.1. Examples	35
6.1.1. Get Basic Info	35
6.1.2. Login	37
6.1.3. Logout	38
6.1.4. Large binary transfer (send)	38
6.1.5. Large binary transfer (read)	41
6.1.6. Get PLC statistics	43
6.1.7. Get Task statistics	46
6.1.8. Get channel statistics	47
6.1.9. Read variables	48
6.1.9.1. Direct mode	48
6.1.9.2. File mode	49

Shark Slave Communication Protocol 1.0

6.1.10. Write variables	50
6.1.10.1. Direct mode	50
6.1.10.2. File mode	51
6.1.11. Time setup Extended	52
6.2. Device identification numbers	53

1. Purpose

The purpose of this standard is to define data communication services and protocols for computer equipment used for monitoring and control of Domat control system devices.

2. Definitions

SSCP - Shark Slave Communication Protocol

MD5 - Hash algorithm (<https://en.wikipedia.org/wiki/MD5>)

FNV1 hash – 32 bit Hash algorithm
(https://en.wikipedia.org/wiki/Fowler%E2%80%93Noll%E2%80%93Vo_hash_function)

Runtime – A process for measuring, controlling and regulating.

Device – A piece of hardware with operating system and running runtime process.

Task – One running program in runtime.

Channel – Communication channel with defined protocol.

RAW data – Undefined structure data. It is usually an array of bytes.

DATE_TIME format – (100ns from 1.1.0001, see
[https://msdn.microsoft.com/en-us/library/system.datetime.ticks\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.datetime.ticks(v=vs.110).aspx))

3. Document history

<i>Description</i>	<i>Version</i>	<i>Date</i>
Document version	1	20.1.2017
Protocol version	7	20.1.2017

4. Transport

4.1. General notes

Byte ordering is “network” (Big endian). Layout and alignment of all variables correspond to VM’s format.

Generic telegram format is in following form:

Layer header	Slave address	SSCP telegram	Layer footer
--------------	---------------	---------------	--------------

Layer header and **footer** is transport layer specific and its meaning is specified in following paragraphs.

Slave address is 8-bit value which specifies unique address of controller on network. It is always present except broadcast messages.

SSCP telegram - telegram described in this document

4.2. TCP/IP

Connection uses TCP protocol. Layer header and footer is absent on this transport layer.

4.3. Serial line

Connection uses standard serial line specified by RS-232C or RS-485 standards. For RS-485 only one master is supported. Character format is 8-N-1, baud rate is selectable.

Layer header is absent, footer is 16-bit CRC calculated by polynomial used in Modbus protocol.

4.4. Other physical layers

It is not planned to implement the SSCP over media other than "Ethernet" and serial line (232, 485). But it is straightforward to do it with simple extensions (like CRC checksums for serial line).

For autodetection and TCP/IP property settings, there is a UDP service. See further in the specification.

4.5. Tunnelling over Linux

Linux offers advanced tools for data transmission and security. Support for tunnel the protocol over such channels will be available in later protocol version.

5. Protocol frames

5.1. Basic structure

Function ID (2 bytes)	Data length (2 bytes)	Data (0..65535 bytes)
-----------------------	-----------------------	-----------------------

Maximal data length will be 65535 bytes. The real maximum is device dependent (cheap/simple devices can have smaller buffers for example) and it is a part of device's capability. Maximal data length is returned in response for login function, too.

Communication runs in a request - response manner. There are no features like windowing or several requests pending.

General rules:

- All function numbers of client requests have bits 6 and 7 equal to zero.
- All OK responses have bit 6 equal to zero and bit 7 equal to one. The rest of the function number is copied.
- All ERROR responses have bits 6 and 7 equal to one
- There are special ERROR responses. These can be sent anytime (when appropriate).

Note: the bits are numbered from 0 (zero).

5.2. Error responses

5.2.1. Insufficient rights

A server responds with insufficient rights response when a client sends a message that does require higher permissions. For example, when somebody with read-only permissions wants to set a value.

The connection will not be broken and the server is ready to accept other messages.

Function	2 B	0xFF 0xFF
Data length	2 B	0x00 0x00

5.2.2. Invalid function

Specified function is unknown and server cannot respond.

Function	2 B	0xFF 0xFE
Data length	2 B	0x00 0x00

5.2.3. Invalid protocol version

Requested version is not supported. This error can be returned for login function and for system command functions.

Function	2 B	0xFF 0xFD
Data length	2 B	0x00 0x00

5.2.4. Command specific error codes

All commands can have an error response, in this case response follow general rules with telegram constructed as follow

Function	2 B	(0xC0 0x00) Function code
Data length	2 B	(0x00 0x04) + Optional data length
Error code	4 B	Error codes from following table
Optional data	? B	Description below error codes table

5.3. Table of error codes

Error	Code	Description
NoError	0x0000	
NoResponse	0x0001	Not sent by controller, used internally
FailedToConnect	0x0002	Not sent by controller, used internally
NotImplemented	0x0003	Not sent by controller, used internally
InvalidFunctionReceived	0x0004	Not sent by controller, used internally
WrongLogin	0x0101	
NoSuchFile	0x0102	
NoSuchVariable	0x0103	
NoSuchTask	0x0104	
WrongOrder	0x0105	

WrongParameter	0x0106	
InvalidGroupId	0x0107	
TransmissionInProgress	0x0108	
NotRegistered	0x0109	
WriteFailed	0x010A	
NotAllDataReceived	0x010B	
InvalidCrc	0x010C	
DataTooLong	0x010D	
TooLongUseFileTransfer	0x010E	
FileNameTooLong	0x010F	
VariableCountLimitExceed	0x0110	
OutOfBounds	0x0111	
SizeMismatch	0x0112	
OperationDenied	0x0113	
NotLogged	0x0114	
InvalidState	0x0115	
UnknownChannel	0x0116	
DriverCommandTimeout	0x0117	
UnknownDriverCommand	0x0118	
NoResourcesAvailable	0x0119	
ChunkReadFailed	0x011A	
ChunkWriteFailed	0x011B	

NoSuchMetadata	0x011C	In RT rev <= 44924 this is reported as ChunkReadFailed
Async	0x011D	
SysCmd_NewImage	0x0801	
SysCmd_InvalidImageArea	0x0802	
SysCmd_CreateBootImage	0x0803	
SysCmd_WarmReboot	0x0804	
SysCmd_ColdReboot	0x0805	
SysCmd_StartPlc	0x0806	
SysCmd_StopPlc	0x0807	
SysCmd_SetMacAddress	0x0808	
SysCmd_Timeout	0x0809	
SysCmdRequestActive	0x080C	
SysCmdWaitTimeout	0x080D	
AlreadyRunning	0x080A	never sent at the moment
AlreadyStopped	0x080B	never sent at the moment

5.3.1. Optional data

Available for TransmissionInProgress, NoSuchVariable, WriteFailed, SizeMismatch, InvalidState and OperationDenied error codes at the moment. For these codes, data is 64 bit integer, in which every bit correspond to one variable in request for which this error code is valid. This is used for example if ReadVariableDirectly is used for read multiple variables in which more than one variable shouldn't exists.

5.4. Device management and autodetection

This feature is a complement to the SSCP and serves to facilitate to start to communicate with unknown devices or devices that are in different IP address range. Frames are sent over UDP broadcast and does not contain layer header, footer and SSCP address fields.

UDP is chosen because it is possible to broadcast to all devices on a network. This broadcast works in case of incompatible IP address settings too. UDP service listens on port 8010 by default.

All commands can be sent also as standard command. In this case the SSCP address is present ([4.1](#))

5.4.1. Get Basic Info

Retrieves the basic information from a device. All the data about the HW configuration are encoded into a binary XML file. A reference implementation can be found here

<http://blogs.microsoft.co.il/blogs/tamir/> under a WBXMLDocument. Description of the document and token ID assignments are present in the Appendix.

5.4.1.1. Request frame format

Function	2 B	0x00 0x00
Data length	2 B	
Version	1 B	Version 1 - Send the <ul style="list-style-type: none"> • Serial number • TCP/IP properties • application version • HW identification
Serial number	1 + ? B	Serial number length (can be 0 if missing) followed by its value
Username	1 + ? B	Length (1 byte), UTF-8 encoded string
Password	1 + ? B	Length (1 byte), MD5 hash
Start offset	2 B	Offset into config memory region
Requested size	2 B	Size of transferred block (0xffff for all bytes, have to fill into frame) - can be 0 for only detecting devices in the network

5.4.1.2. Positive response format

Function	2 B	0x80 0x00
Data length	2 B	
Size	2 B	0x00 (Read basic settings)
Serial number	1 + ? B	Serial number length, followed by serial number
Endianness	1 B	Target endianness (1 - big, 0 - little)

Platform ID	4 B	Identification of platform (6.2)
Runtime version length	1 B	0x04
Runtime version	4 B	31 .. 29 - Major version 28 .. 26 - Minor version 25 .. 21 - Release day 20 .. 17 - Release month 16 .. 0 - SVN revision
Information		Device basic information

Information data are structured data enclosed in begin (0x3E) and end (0x3F) tag. Every information starts with definition tag followed by its data:

<i>Value</i>	<i>Data</i>	<i>Description</i>
0x3E	-	Open tag
0x01	Device name in UTF-16 ended by value 0x0000	Device name tag. Maximum length is 64 Bytes
0x02	Slave address (1 Byte)	SSCP slave address tag
0x04	TCP port (2 Bytes)	SSCP TCP slave port tag
0x05	SSL TCP port (2 Bytes)	SSCP SSL slave port tag
0x3F	-	Close tag

5.4.1.3. Negative response format

Follow common response format ([5.2](#)).

Note: If 'Requested size' is zero then positive response is returned regardless of wrong credentials - because it is used for autodetect PLCs in the network

5.5. Access privileges

5.5.1. Access levels

Access levels are divided into three categories mentioned before, so for recapitulation:

- Read only
- Full control
- Engineering

These levels are represented by 8bit unsigned integer which in max can give 256 access levels, higher number means higher privileges, default assignment of levels is as follows:

Category	Level
Read only	0x10 (16)
Full control	0x80 (128)
Engineering	0xFF (255)

5.5.2. Privileges for commands

Command	Minimal required privilege
GetBasicInfo (5.4.1)	Engineering, if RequestedSize is equal to zero no valid login is required
Login (5.5.4)	not applicable
Logout (5.5.5)	not applicable
InitiateDataSend (5.6.1)	Engineering, for file access see below
SendDataChunk (5.6.1)	Engineering, for file access see below
FinishDataSend (5.6.1)	Engineering, for file access see below
InitiateDataReceive (5.6.2)	Engineering, for file access see below
ReceiveDataChunk (5.6.2)	Engineering, for file access see below
GetPlcStatistics (5.7.1)	Read only
GetTaskStatistics (5.7.2)	Read only

GetChannelStatistics (5.7.3)	Read only
ReadVariablesDirectly (5.8.1.1)	Read only
WriteVariableDirectly (5.8.1.2)	Full control
TimeSetupExtended (5.9.2)	Read only or Full control for set

5.5.3. Access privileges for file access

<i>File name</i>	<i>Minimal required privilege for read</i>	<i>Minimal required privilege for write</i>
/var/direct	Read only	Full control
/log	Engineering	not writable
/d/*	Engineering	Engineering

5.5.4. Login

Login is expected as the first command after connection establishment. Any other function will close the connection immediately (except of broadcast messages ([5.4](#)) with corresponding access rights).

There are 3 login levels (see [5.5.1](#)). The PLC will deny (insufficient rights) execution of functions that are not available at the level of current login ([5.2.1](#)).

5.5.4.1. Request frame format

Function	2 B	0x01 0x00
Data length	2 B	
Requested version	1 B	Protocol version requested by client (Currently 7)
[Client's] Max data size in telegrams	2 B	Client indicates the maximal buffer size to the server. Server must not create frames that are bigger than this.
User name length	1 B	Encoded user name is max 255 chars.
User name	? B	Short string UTF-8 encoded.
Password hash length	1 B	
Password hash	? B	MD5

ProxyID length	1 B	ProxyID Length - from version 2 of the protocol. Max 255 chars.
ProxyID	? B	ProxyID String UTF8 Encoded - from version 2 of the protocol.

5.5.4.2. Positive response frame format

Function	2 B	0x81 0x00
Data length	2 B	(0x00 0x14) + Optional data length
Protocol version	1 B	0x07 (current version)
[Server's] Max data size in telegrams	2 B	Server indicates the maximal buffer size to the client. Client must not create frames that are bigger than this.
Right group	1 B	See 5.5.1
Image GUID	16 B	Image identification
Optional data	2+ B	<p>Tag-Value structure, this data is enclosed in 0x3E and 0x3F tags</p> <p>known tags</p> <ul style="list-style-type: none"> 1 - device name, string 2 - SSCP address, 8bit number 3 - unique image build id, 32bit number 4 - SSCP TCP port, 16bit number 5 - SSCP SSL port, 16bit number

5.5.4.3. Negative response frame format

PLC disconnects immediately and does not send any data, exception is connection via proxy server which can report error before disconnecting.

5.5.5. Logout

Function immediately closes the connection with the device. No response is received.

5.5.5.1. Request frame format

Function	2 B	0x01 0x01
Data length	2 B	0x00 0x00

5.5.5.2. Positive or negative response frame format

There is no response to this message. The connection is closed by the PLC immediately.

5.6. File/large binary data transfer

By large data used in next chapters we mean data which cannot be transferred in one frame. These data have to be split to more frames. The method for sending and receiving such data is described below.

5.6.1. Send data

A typical scenario for transfer is shown in the next table:

<i>Client</i>	<i>Server (PLC)</i>
Initiate send data transmission (data identification, length, ...)	
	isACK -> continue NAK -> terminate transmission
Send chunk	
	sACK (offset) NAK -> terminate transmission
Send chunk....	
Finish send data (CRC)	OK ERROR

The transmission terminates when:

- Server or client stops it - server can issue error message (insufficient rights,...)
- All data has been sent

5.6.1.1. Initiate send data transmission

Initialize file transfer

5.6.1.1.1. Request frame format

Function	2 B	0x02 0x00
Data length	2 B	
Data identification length	1 B	Data identification == file name.
Data identification	? B	Encoded file name is max 64 chars.
Data size	4 B	Size of the data
Timestamp	8 B	DATE_TIME format

5.6.1.1.2. Positive response frame format

Function	2 B	0x82 0x00
Data length	2 B	0x00 0x00

Receiving this message indicates that transfer can continue with next data.

5.6.1.1.3. Negative response frame format

Deny the transfer for whatever reason.

Follow common response format ([5.2](#)).

5.6.1.2. Send chunk

Send pack of file data from specified offset with defined length.

5.6.1.2.1. Request frame format

Function	2 B	0x02 0x01
Data length	2 B	(4 + data length)
Data offset	4 B	The data that are being sent are from this offset
Data	? B	Data itself

5.6.1.2.2. Positive response frame format

Function	2 B	0x82 0x01
----------	-----	-----------

Data length	2 B	0x00 0x04
Offset	4 B	Offset being acknowledged

5.6.1.2.3. Negative response frame format

Follow common response format (5.2). (OutOfBound, WriteFailed, WriteChunkFailed)

5.6.1.3. Finish send data transmission

All data send. With this function gets checksum 32 bits CRC.

5.6.1.3.1. Request frame format

Function	2 B	0x02 0x02
Data length	2 B	0x00 0x02
CRC	2 B	CRC of the data.

5.6.1.3.2. Positive response frame format

Function	2 B	0x82 0x02
Data length	2 B	0x00 0x00

5.6.1.3.3. Negative response frame format

Follow common response format (5.2). (NotAllDataReceived, WriteFailed)

5.6.2. Receive data

A typical scenario for transfer is shown in the next table:

<i>Client</i>	<i>Server (PLC)</i>
Initiate receive data transmission (data identification)	
	irACK total size, CRC -> continue NAK -> terminate transmission
ENQ -> (offset)	

NAK -> terminate transmission	
	Send chunk
ENQ (offset) NAK -> terminate transmission	
	Send chunk....

5.6.2.1. Initiate receive data transmission

Initialize reading data. Send to device file name for download from it.

5.6.2.1.1. Request frame format

Function	2 B	0x02 0x10
Data length	2 B	
Data identification length	1 B	Data identification == file name
Data identification	? B	Encoded file name is max 64 chars.

5.6.2.1.2. Positive response frame format

Function	2 B	0x82 0x10
Data length	2 B	0x00 0x0E
Total size	4 B	Size of the data.
Timestamp	8 B	DATE_TIME format
CRC	2 B	CRC of the data

Receiving this response indicates that the transfer can continue.

5.6.2.1.3. Negative response frame format

Deny the transfer.

Follow common response format ([5.2](#)). (NoSuchFile, DataTooLong, FilenameTooLong,

TransmissionInProgress)

5.6.2.2. ENQ - Receive chunk

Confirm read data and ask for more data, if available

5.6.2.2.1. Request frame format

Function	2 B	0x02 0x11
Data length	2 B	0x00 0x04
Offset	4 B	Offset to retrieve

5.6.2.2.2. Positive response frame format

Function	2 B	0x82 0x11
Data length	2 B	(4 + data length)
Data offset	4 B	The data that are being sent are from this offset
Data	? B	Data itself

5.6.3. Virtual file system structure

Path	Description	Comment
/sys/sexm	In memory RT's image.	According to capabilities (usually 2MB)
/sys/rt	Area for upgrade of RT	
/sys/sex1	Area for the first shark image.	According to capabilities (usually 2MB)
/sys/sex2	Area for the second shark image.	According to capabilities (usually 2MB)
/sys/caps	PLC capabilities.	See appendix for file format. This file describes the PLC as much as possible. Usually, it will be used by IDE.
/var/direct	Variable value/descriptions - direct query	

/var/group/[groupid]/[flags]	Variable value/description for registered group	
/log	PLC log	PLC Log.
/drv/result	Response for driver command longer than one packet	
/d/...	Mountpoint for general filesystem	Web,...

5.7. Device statistics

Functions for reading basic statistics from the device. It is possible to read statistics from:

- Device
- Task
- Channel

5.7.1. Get PLC statistics

Gets the default statistics from the device.

5.7.1.1. Request frame format

Function	2 B	0x03 0x00
Data length	2 B	0x00 0x00

5.7.1.2. Positive response frame format

Function	2 B	0x83 0x00
Data length	2 B	
Statistics version	1 B	Following description describes the version 4
Block type	1 B	0 = Runtime statistics
Block version	1 B	1
Block length	2 B	0x00 0x1C
Normal Tasks count	1 B	Corresponds with the max number of threads (i.e. tasks that run

		concurrently in normal operation).
Max Task ID	1 B	Max task ID.
Evaluator state	1 B	<ul style="list-style-type: none"> ● Stopped = 0 ● RunningNormalTasks = 1 ● StoppingExecution = 2 ● RunningExceptionStateTask = 3 ● ExceptionStateTaskFailed = 4 ● NoExceptionStateTaskDefined = 5 ● Commissioning = 6 ● InvalidImage = 7 ● NoImage = 8 ● WaitingForDebugger = 9 ● PreparedForStart = 10
Run mode	1 B	<ul style="list-style-type: none"> ● FullRun = 0 ● CommunicationOnly = 1 ● EvaluationOnly = 2 ● Commissioning = 3 ● CommunicationsWithTransform = 4 ● PrepareOnly = 5 ● StartDisabledBySwitch = 32 ● InvalidImageVersion = 33 ● NoMemoryForImage = 34
UpTime	8 B	TIME
Running tasks	8 B	Bit mask of running task (bit position with 1 corresponds to task's ID that runs)
Tasks with exception	8 B	Bit mask of tasks that encountered an error.
Block type	1 B	1 = Memory statistics
Block version	1 B	1
Block length	2 B	0x00 0x10
Total heap	2 B	Total heap memory available for runtime
Free heap before load	2 B	Free heap after runtime startup (memory used for runtime itself)

Free heap	2 B	Free heap after image load and process
Total code space	2 B	Total space available for image
Free code space	2 B	Free space after image save
Retain size	2 B	Retain area size
Allocator total size	2 B	
Allocator free space	2 B	
Block type	1 B	2 = Sections statistics
Block version	1 B	1
Block length	2 B	0x00 0x06
VMEX section	2 B	Heap memory consumed by VMEX section (VM image)
RTCM section	2 B	Heap memory consumed by communication section
Other sections	2 B	Heap memory consumed by other sections
Block type	1 B	3 = RCware DB statistics
Block version	1 B	1
Block length	2 B	0x00 0x15
Client status	1 B	<ul style="list-style-type: none"> ● Disabled = 0 ● NotUsed = 1 ● Idle = 2 ● Connected = 3 ● Unathorized = 4 ● NotAvailable = 5 ● FailedToConnect = 6 ● HostNotFound = 7 ● Connecting = 8 ● PageNotFound = 9 ● DbError = 10
Records saved	4 B	

Last save time	8 B	
Last request time	8 B	
Block type	1 B	4 = Proxy statistics
Block version	1 B	1
Block length	2 B	0x00 0x17
Proxy status	1 B	<ul style="list-style-type: none"> ● Disabled = 0 ● NotUsed = 1 ● Idle = 2 ● Connected = 3 ● Unathorized = 4 ● NotAvailable = 5 ● FailedToConnect = 6 ● HostNotFound = 7 ● Connecting = 8 ● PageNotFound = 9 ● DbError = 10
Proxy ID	20 B	
Slots total	1 B	
Slots free	1 B	

5.7.2. Get Task statistics

Get basic statistics from the task.

5.7.2.1. Request frame format

Function	2 B	0x03 0x01
Data length	2 B	0x00 0x01
Task ID	1 B	

5.7.2.2. Positive response frame format

Function	2 B	0x83 0x01
Data length	2 B	
Statistics version	1 B	Following description describes the version 1
Cycle count	8 B	UINT 64
Last cycle duration	8 B	TIME
Average cycle duration	8 B	TIME
Minimal cycle duration	8 B	TIME
Maximal cycle duration	8 B	TIME
Version 2 adds following:		
Waiting for debugger	1 B	bool
Debugger actual UID	4 B	
Debugger actual offset	4 B	

5.7.2.3. Negative response frame format

Follow common response format ([5.2](#)). (NoSuchTask)

5.7.3. Get Channel statistics

Get base statistic from the channel.

5.7.3.1. Request frame format

Function	2 B	0x03 0x10
Data length	2 B	0x00 0x04
Channel ID	4 B	FNV1 Hash of channel name

5.7.3.2. Positive response frame format

Function	2 B	0x83 0x10
----------	-----	-----------

Data length	2 B	
Statistics version	1 B	Following description describes the version 1
Sent packets	4 B	Total number of packets sent through this channel/driver
Received packets	4 B	Total number of packets received for this channel/driver
Wrong packets	4 B	Total number of wrong packets
Sent bytes	4 B	Total number of bytes sent through this channel/driver
Received bytes	4 B	Total number of bytes sent through this channel/driver
Endpoints count	2 B	Number of consecutive endpoints statistics
Endpoints	n * 12 B	

5.7.3.2.1. Endpoint statistics

Average cycle time	4 B	Average communication cycle time in milliseconds
Maximal cycle time	4 B	Maximal comm. cycle time in milliseconds
Minimal cycle time	4 B	Minimal comm. cycle time in milliseconds

5.7.4. Negative response frame format

Follow common response format ([5.2](#)). (UnknownChannel)

5.8. Data communication

Every variable is assigned a ID number (4 bytes unsigned int) that does not change with program changes. Once assigned, the ID can only change when a specific action is taken (manual reordering or data type changes).

Clients must be aware of data types and sizes of all variables and layouts of complex variables (strings, structs, FBs, arrays). All information needed can be retrieved from variable metadata 'value structure'.

There are two methods to read variables:

- Direct mode - It is used to read values which fit into one telegram. PLC responses with values for requested variables.
- File mode - It is used to read values which are larger than one telegram.

Both modes start with initialization command. The mode is detected by the response.

5.8.1. Direct mode
5.8.1.1. Read variables directly

Read data from variables specified with their UID and optionally with their offset and length.

5.8.1.1.1. Request frame format

Function	2 B	0x05 0x00
Data length	2 B	
Flags	1 B	B7 (0x80) 0 - Offset and Length fields are missing 1 - Offset and Length fields are present B6 UID Type 0 - Communication UID 1 - VM UID B4 Task ID 0 - Task ID not present 1 - Task ID present; next byte contains task ID B0..B2 (response format) 0 - Get All Values
Task ID	1 B	Used to access local variables. Present only if appropriate flag is set.
UID	4 B	Variable unique identification
Offset	4 B	Offset in value (with Length field this is handy for accessing just a part of a struct/FB/array). Present only if appropriate flag is set.
Length	4 B	Length of the data to copy as value. Present only if appropriate flag is set.

The UID, Offset and Length fields repeat for every requested variable, the maximum protocol supported amount is 64 variables in one request.

5.8.1.1.2. Positive response frame format

Function	2 B	0x85 0x00
Data length	2 B	
Data	? B	Variable data in order according to the order of the definitions.

5.8.1.1.3. Negative response frame format

Follow common response format ([5.2](#)). (WrongParameter, DataTooLong, NoSuchVariables, TooLongUseFileTransfer = File mode used)

5.8.1.2. Write variables directly

Direct write to variables specified by their UID and optionally with their offset and length. All writes that fit into the maximal packet size can be set directly without accessing the file. Other values are written by file mode.

5.8.1.2.1. Request frame format

Function	2 B	0x05 0x10
Data length	2 B	
Flags	1 B	B7 (0x80) 0 - Offset and Length fields are missing 1 - Offset and Length fields are present B6 UID Type 0 - Communication UID 1 - VM UID B5 (0x20) 0 - Direct mode 1 - File mode B4 Task ID 0 - Task ID not present 1 - Task ID present; next byte contains task ID

Task ID	1 B	Used to access local variables. Present only if appropriate flag is set.
Number of variables	1 B	This field is only present if file mode is not requested (Flags.B5 == 0)
UID	4 B	Variable unique identification
Offset	4 B	Offset in value (with Length field this is handy for accessing just a part of a struct/FB/array). Present only if appropriate flag is set.
Length	4 B	Length of the data to copy as value. Present only if appropriate flag is set.
The UID, Offset and Length fields repeat for every requested variable, the maximum protocol supported amount is 64 variables in one request.		
Data	? B	Variable data in order according to the order of the definitions. Present only in the direct mode. In the file mode, these data must be written to <code>/var/direct</code> before this request.

5.8.1.2.2. Positive response frame format

Function	2 B	0x85 0x10
Data length	2 B	0x00 0x00

5.8.1.2.3. Negative response frame format

Follow the common response format ([5.2](#)). (NoSuchVariable, TooLongUseFileTransfer = File mode)

5.8.2. File mode

If the variable data cannot fit to a single packet, they must be transferred using the file transfer. (see [5.6](#)).

5.8.2.1. Read variables directly

Client sends request to read variables directly. Server responds with error **TooLongUseFileTransfer** and puts the data to the file `/var/direct` instead of the positive response data field. Then client reads the file.

5.8.2.2. Write variables directly

Client recognizes, that data cannot fit to a single packet and sends the data to file `/var/direct`. Then it sends request to write variables directly with flag B5 set and only with variables definitions, without the data.

5.8.3. Data format

5.8.3.1. Get all values

The response is just a subsequent sequence (in the order of request / registration) of values.

```
| value of var 1 | value of var 2 | ... | value of the last var |
```

5.8.3.2. Get all changed values

The response is a sequence of pairs {variable index; variable value}. Variable index is the index of the variable (and offset and length) during the registration.

```
| index 1 | value of var of index 1 | index 2 | value of var of index 2  
| ... | index n | value of var of index n |
```

5.8.3.3. Set all values request format

The request is just a subsequent sequence (in the order of registration) of registered values.

```
| value of var 1 | value of var 2 | ... | value of the last var |
```

5.8.3.4. Set some values request format

The request is a sequence of pairs {variable index; variable value}. Variable index is the index of the variable (and offset and length) during the registration.

```
| index 1 | index 2 | ... | index n | value of var of index 1 | value of  
var of index 2 | ... | value of var of index n |
```

Note: values can be packed into bytes.

5.9. Time functions

5.9.1. Time setup

Used for query actual RTC time in UTC format, if timestamp is provided (data length is equal to eight and timestamp presented) then RTC is preset to this value. **From protocol version 7 this command is removed, use time setup extended command instead.**

5.9.1.1. Request frame format

Function	2 B	0x06 0x02
Data length	2 B	0/8 bytes
Timestamp	0/8 B	Optional 64bit C# timestamp in UTC format for time preset

5.9.1.2. Positive response frame format

Function	2 B	0x86 0x02
----------	-----	-----------

Data length	2 B	0x00 0x08
Current timestamp	8 B	Actual PLC time in UTC format

5.9.1.3. Negative response frame format

Follow common response format ([5.2](#)). (WriteFailed)

5.9.2. Time setup extended

Used for querying of actual RTC time in UTC or local (from controller's side) format. This command is available from protocol version 7. If no *Timestamp* value is present, the function returns current date and time in the requested format.

5.9.2.1. Request frame format

Function	2 B	0x06 0x04
Data length	2 B	2/10 bytes
Command	1 B	0x01 - Get RTC in UTC format 0x02 - Get RTC in local format 0x10 - Set RTC from UTC format (timestamp following after flags) 0x11 - Set RTC from local format 0x20 - Get timezone offset (returned as timestamp) 0x21 - Get daylight saving offset (returned as timestamp)
Flags	1 B	Not used in current version - set to zero
Timestamp	0/8 B	Optional 64bit C# timestamp in format specified by command for time preset

5.9.2.2. Positive response frame format

Function	2 B	0x86 0x04
Data length	2 B	0/8 bytes
Current timestamp	0/8 B	Optional returned timestamp (UTC/local/offsets/...)

5.9.2.3. Negative response frame format

Follow common response format (5.2). (WrongParameter, WriteFailed)

6. Appendix

6.1. Examples

6.1.1. Get Basic Info

Send broadcast detection command

0x0000	Function Get Basic Info
0x001D	Data length (29 Bytes)
0x01	Version
0x00	Serial number is not known
0x05	User name length
0X61646D696E	User name in utf-8 ("admin")
0x10	Password length (16 Bytes)
0X038C0DC8A988FFEA13AF047228FB696 0	MD5 hash for password
0x0000	Offset in memoryregion
0x0000	Size of transfered block (0 = detect only)

Respond

0x8000	Respond to Get Basic Info function
--------	------------------------------------

0x0028	Data length (40 bytes)
0x043D	Size of whole config block (1085 Bytes)
0x08	Serial number length (8 Bytes)
0X0000000A14BE14B0	Serial number
0x00	Target endianness (little)
0x00030007	Platform ID (uPLC - mark150/485s)
0x04	Runtime version length (4 Bytes)
0X22F2C002	Runtime version (1.0.2309.49154)
	0010 0010 1111 0010 1100 0000 0000 0010b =>
	0 1100 0000 0000 0010b (0x0c002) - Revision (49154)
	1001 (0x9) - Month (9)
	1 0111 (0x17) - Day (23)
	000 (0x0) - Minor version (0)
	001 (0x1) - Major version
0x3E	Open tag
0x01	Device name tag
0x0050004C00430000	“PLC”
0x02	SSCP slave address tag
0x01	1

0x04	SSCP TCP slave port tag
0x303A	12346
0x05	SSCP SSL slave port tag
0x0000	0
0x3F	Close tag

6.1.2. Login

Send Login

0x01	SSCP address
0x0100	Login function
0x001B	Data length (27 Bytes)
0x07	Requested version (7)
0x2800	Max data size in telegrams (10240 B)
0x05	User name length (5 Bytes)
0X61646D696E	User name ("admin") - UTF-8
0x10	Password length (16 Bytes)
0X038C0DC81258FFEA11BF047244FB6960	MD5 password
0x00	ProxyID length (0 Bytes => no proxy)

Respond

0x01	SSCP address
------	--------------

Shark Slave Communication Protocol 1.0

0x8100	Login function success respond
0x001B	Data length (27 Bytes)
0x07	Protocol version (7)
0x00E4	Max data size in telegrams (228 Bytes)
0xFF	Right group (Engineering) see 5.5 Access privileges
0XF02A9D0B2A377544B6AF282105A2CA00	Image GUID
0x3E	Open tag
0x03	Unique image build ID
0X584544F8	Image build ID
0x3F	Close tag

6.1.3. Logout

Send Log out

0x01	SSCP address
0x0101	Logout function
0x0000	Data length (0 Bytes)

Respond

No response sent

6.1.4. Large binary transfer (send)

Send Initiate send data transmission

Shark Slave Communication Protocol 1.0

0x01	SSCP address
0x0200	Initiate send data transmission function
0x0B	File name length (11 Bytes)
0x2F7661722F646972656374	File name (“/var/direct” -> variable value data) - UTF-8
0x00000170	File size (368 B)
0x08D43FBA4CFDD0D3	DATE_TIME (18.1.2017 15:54:35.425)

Respond

0x01	SSCP Address
0x8200	Initiate send data transmission respond
0x0000	Data length

Send chunk

0x01	SSCP Address
0x0201	Send chunk function
0x00E4	Data length (228 Bytes)
0x00000000	Data offset
0x...	Raw data. Length = 228 - 4

Respond

0x01	SSCP Address
0x8201	Send chunk function respond

0x0004	Data length
0x00000000	Offset being acknowledged

Continue Sending

0x01	SSCP Address
0x0201	Send chunk function
0x00E4	Data length (228 Bytes)
0x000000E0	Data offset
0x...	Raw data. Length = 228 - 4

Respond

0x01	SSCP Address
0x8201	Send chunk function respond
0x0004	Data length
0x000000E0	Offset being acknowledged

Continue until all data in file are sent
Send finish

0x01	SSCP address
0x0202	Finish send data transmission function
0x0002	Data length
0x660E	CRC

Respond

0x01	SSCP Address
0x8202	Finish send data transmission respond
0x0000	Data length

6.1.5. Large binary transfer (read)
Send Initiate receive data transmission function

0x01	SSCP Address
0x0210	Initiate receive data transmission function
0x000C	Data length (12 Bytes)
0x0B	File name length (11 Bytes)
0X2F7661722F646972656374	File name (“/var/direct” -> read variable value data) - UTF-8

Receive

0x01	SSCP Address
0x8210	Initiate receive data transmission response
0x000E	Data length (14 Bytes)
0x00000548	Total file (data) size (1352 Bytes)
0x0000000000000000	Timestamp (0)
0X4FFA	File CRC

Send ENQ

Shark Slave Communication Protocol 1.0

0x01	SSCP Address
0x0211	ENQ function
0x0004	Data length (4 Byte)
0x00000000	Offset to retrieve

Receive

0x01	SSCP Address
0x8211	ENQ function response
0x00E4	Data length (228 Bytes)
0x00000000	Data offset
0x...	Raw data. Length = 228 - 4

Send ENQ

0x01	SSCP Address
0x0211	ENQ function
0x0004	Data length (4 Byte)
0x000000E0	Offset to retrieve

Receive

0x01	SSCP Address
0x8211	ENQ function response
0x00E4	Data length (228 Bytes)

0x000000E0 Data offset

0x... Raw data. Length = 228 - 4

Continue for all data reading (1352 Bytes). Increase Offset to retrieve next data. Finally, calculate CRC and compare with the received CRC in the Initiate receive data transmission response

6.1.6. Get PLC statistics

Send Get PLC statistics

0x01 SSCP Address

0x0300 Get PLC statistics function

0x0000 Data length

Respond

0x01 SSCP Address

0x8300 Get Statistics response

0x0073 Data length (115 Bytes)

0x04 Statistics version

0x00 Runtime statistics block ID

0x01 Block version

0x001C Block length (28 Bytes)

0x01 Normal tasks count

Shark Slave Communication Protocol 1.0

0x00	Max task ID
0x01	Evaluator state (Running normal tasks)
0x00	Run mode (Full run)
0x000000004f6d4080	Uptime
0x0000000000000001	Running tasks mask (ID = 0)
0x0000000000000000	Tasks with exception mask (none)
0x01	Memory statistics block ID
0x01	Block version
0x0010	Block length (16 Bytes)
0x208f	Total heap size (8335 kB)
0x1e5f	Free heap after runtime startup (7775 kB)
0x1d73	Free heap after image load and process (7539 kB)
0x01ff	Total space available for image (511 kB)
0x0123	Free space after image save (291 kB)
0x0001	Retain size (1 kB)
0x0200	Allocator total size (512 kB)
0x0200	Allocator free space (512 kB)
0x02	Sections statistics block ID

Shark Slave Communication Protocol 1.0

0x01	Block version
0x0006	Block size
0x008E	VMEX section used (142 kB)
0x0040	RTCM section used (64 kB)
0x000D	Other sections used (13 kB)
0x03	RCware DB statistics block ID
0x01	Block version
0x0015	Block size (21 Bytes)
0x00	Client status (Disabled)
0x00000000	Records saved
0x0000000000000000	Last save time
0x0000000000000000	Last request time
0x04	Proxy statistics block ID
0x01	Block version
0x0017	Block size
0x00	Proxy status (Disabled)
0x00... (20 Bytes)	Proxy ID
0x00	Slots total

0x00 Slots free

6.1.7. Get Task statistics

Send Get Task statistics

0x01 SSCP address

0x0301 Get Task statistics function

0x0001 Data length

0x00 Task ID

Respond

0x01 SSCP address

0x8301 Get Task statistics response

0x0032 Data length (50 Bytes)

0x02 Data version

0x00000000000044707 Cycle count (280327)

0X000000000001ADB0 Last cycle duration (110 000 ns)

0X000000000001C1E5 Avarage cycle duration (115 173 ns)

0X000000000001ADB0 Minimal cycle duration (110 000 ns)

0x0000000000003A980 Maximal cycle duration (240 000 ns)

0x00 Waiting for debugger (false)

0x00000000 Debugger actual UID

0x00000000 Debugger actual offset

6.1.8. Get channel statistics

Send Get Channel statistics

0x01 SSCP address

0x0310 Get Channel statistics function

0x0004 Data length

0XD712906A Channel name FNV1 hash ("channel")

Respond

0x01 SSCP address

0x8310 Get Channel statistics response

0x0023 Data length (35 Bytes)

0x01 Data version

0x00000000 Sent packets

0x00000000 Received packets

0x00000000 Wrong packets

0x00000000 Sent bytes

0x00000000 Received bytes

0x0001 Endpoints count

0x00000000	Endpoint 1 average cycle time (0 ms)
0x00000000	Endpoint 1 maximal cycle time (0 ms)
0x00000000	Endpoint 1 minimal cycle time (0 ms)

6.1.9. Read variables

6.1.9.1. Direct mode

Send Read variables directly

0x01	SSCP address
0x0500	Read variables directly function
0x0025	Data length (37 Bytes)
0x80	Flags: 1 - Offset and Length fields are present 0 - Communication UID 0 - Task ID not present 0 - Get All Values
0x000022BE	Value 1 UID
0x000000D9	Value 1 Offset
0x00000001	Value 1 Length (1 Byte)
0x000022C0	Value 2 UID
0x000000DA	Value 2 Offset
0x00000002	Value 2 Length (2 Bytes)

0x000022BF	Value 3 UID
0x00000184	Value 3 Offset
0x00000004	Value 3 Length (4 Bytes)

Response

0x01	SSCP Address
0x8500	Read variables directly response
0x0007	Data length (7 Bytes)
0x00	Value 1 raw data
0x0002	Value 2 raw data
0x42480000	Value 3 raw data

6.1.9.2. File mode
Send Read variables directly

0x01	SSCP address
0x0500	Read variables directly function
0x0011	Data len (17 Bytes)
0x01	Flags: 0 - Offset and Length fields are missing 0 - Communication UID 1 - Get All Values With Metadata
0x00000001	UID = 1

0x000022BE UID = 8894

0x000022BF UID = 8895

0x000022C0 UID = 8896

Response

0x01 SSCP address

0xC500 Read variables directly error

0x0004 Data length (4 Bytes)

0x0000010E TooLongUseFileTransfer error

Next packets follow [6.1.5](#).

6.1.10. Write variables

6.1.10.1. Direct mode

Send Write variables directly

0x01 SSCP address

0x0510 Write variables directly function

0x001D Data length (15 Bytes)

0x80 Flags:
 1 - Offset and Length fields are present
 0 - Communication UID
 0 - Direct mode
 0 - Task ID not present

0x02 Variables count (2 variable)

0x00000001	Variable 1 UID (1)
0x00000000	Variable 1 offset (0)
0x00000001	Variable 1 length (1)
0x00000002	Variable 2 UID (1)
0x00000000	Variable 2 offset (0)
0x00000002	Variable 2 length (1)
0x01	Variable 1 RAW data
0x0235	Variable 2 RAW data

Response

0x01	SSCP address
0x8510	Write variables directly response
0x0000	Data length (0)

6.1.10.2. File mode

Send data to file /var/direct (see [6.1.4](#))

Send Write variables directly

0x01	SSCP address
0x0510	Write variables directly function
0x000D	Data length (15 Bytes)
0xA0	Flags:

1 - Offset and Length fields are present

0 - Communication UID

1 - File mode

0 - Task ID not present

0x000022BE Variable UID

0x00000000 Variable offset

0x00000170 Variable length (368 Bytes)

Response

0x01 SSCP address

0x8510 Write variables directly response

0x0000 Data length (0)

6.1.11. Time setup Extended

Send Time setup Extended

0x01 SSCP address

0x0604 Time setup Extended function

0x0002 Data length (2 Bytes)

0x01 Get runtime date in UTC format

Response

0x01 SSCP address

0x8604 Time setup Extended response

0x0008

Data length

0x08D4407E9341C9AA

Current date time

6.2. Device identification numbers

Platform	Platform ID	Device ID	Device name
Windows	0x00010000	0	Generic windows
Linux	0x00020000	0	Generic linux
		1	iPLC 510
		2	markMX
		3	iPLC P-100
		4	mark220
		5	mark320
		6	iPLC 520
		7	RaspberryPi
		8	esg001
		9	mark325
		10	UniPi
		11	UniPi (RPi2)
ARM / uPLC	0x00030000	0	uPLC100
		1	M007
		2	HT-1
		3	mark150s
		4	imio100

Shark Slave Communication Protocol 1.0

	5	mark120
	6	mark125
	7	mark150/485s
	8	mark150
	9	mark150/485
	10	mark100
	11	imio105
	12	icio200
	13	icio205
	xxx	xxx

Note: Red background means that device is obsolete