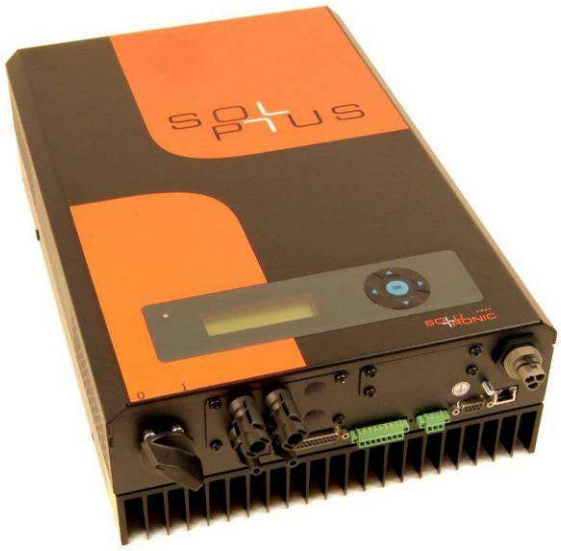
Specification
SOLUTRONIC Protocol



A new generation of inverters

SOLPLUS 25, SOLPLUS 35, SOLPLUS 50, SOLPLUS 55, SOLPLUS 100, SOLPLUS 120, SOLPLUS 300

Version B12

MOS /Date: 27 Jan. 2011/

Subject to change without prior notice

2.1 4-byte protocol

Date	Created	Revision	Comment
25 Aug. 2007	WS	A1	First version
16 July 2008	DS	B1	
10 Sep. 2008	DS	B2	
24 June 2009	DS	B3	
24 Nov. 2009	DS	B4	
18 Jan. 2010	TS	B5	
12 May 2010	DS	B7	
1 July 2010	TS	B8	
6 Aug. 2010	DS	B9	
27 Oct. 2010	TS	B10	File 15, Index 2,5,9,13h (parameters of variable length) Length=0 Act. value =0 saves ¼ space
17. Dec. 2010	TH	B11	Byteswap in binary files
27 Jan. 2011	MOS/DS	B12	File expansion 15/31 and S0-EYL for COMBOX, system EYL

Contents

Dear customer:

This specification provides details on the SOLUTRONIC protocol.

The SOLUTRONIC protocol lets SOLPLUS+ software and external devices communicate with SOLPLUS series solar inverters.

If a problem should arise or if you have questions, please contact us:

Solutronic AG

Küferstrasse 18

73257 Köngen

Germany

Tel.: +49 7024 96128-0

Fax: +49 7024 96128-50

info@solutronic.de

www.solutronic.de

1 Introduction

This specification describes the general protocol for controlling and monitoring SOLPLUS solar inverters.

2 Protocol structure

2.1 4-byte protocol

Description	Length in bytes	Byte number
STX (Start of text)	1	1
Recipient address (serial number)	2	2 (high byte) - 3 (low byte)
Sender address (serial number)	2	4 (high byte) - 5 (low byte)
Parameter numbers (with control bits)	2 (Bit 11 = 0)	6 (high byte) - 7 (low byte)
Parameter value	4	8 (HH byte) - 11 (LL byte)
Checksum (XOR of bytes 2-11)	1	12
ETX (End of text)	1	13

This protocol is used when bit 11 of the parameter number is 0.

2.2 128-byte protocol

Description	Length in bytes	Byte number
STX (Start of text)	1	1
Recipient address (serial number)	2	2 (high byte) - 3 (low byte)
Sender address (serial number)	2	4 (high byte) - 5 (low byte)
Parameter numbers (with control bits)	2 (Bit 11 = 1)	6 (high byte) - 7 (low byte)
Parameter value	128	8 (HH byte) – 135 (LL byte)
Checksum (XOR of bytes 2-135)	1	136
ETX (End of text)	1	137

This protocol is used when bit 11 of the parameter number is 1.

2.3 Error protocol

Description	Length in bytes	Byte number
STX (Start of text)	1	1
Recipient address (serial number)	2	2 (high byte) - 3 (low byte)
Sender address (serial number)	2	4 (high byte) - 5 (low byte)
Parameter numbers (with control bits)	2 (Bit 12 = 1)	6 (high byte) - 7 (low byte)
Parameter value	4	8 (HH byte) - 11 (LL byte)
Checksum (XOR of bytes 2-11)	1	12
ETX (End of text)	1	13

Bit 12 of the parameter number is defined as the error bit in the protocol. If an error occurs in reading or writing (such as non-existent parameter, limit value exceeded, password protection etc.), the error bit is set. The error number is written to the value of this protocol.

Error numbers:

1 = Parameter non-existent (for read and write commands)

2 = Parameter password-protected (only for write commands)

3 = Parameter value too large as transferred (higher than maximum) (for write commands only)

4 = Parameter value too small as transferred (lower than minimum) (for write commands only)

5 = Parameter cannot be changed (i.e. an actual value) (only for write commands)

6 = Access code 1 (parameter is protected by access code 1)

7 = Access code 2 (parameter is protected by access code 2)

8 = No valid command (for example, neither read nor write specified)

9 = Parameter value invalid (e.g. invalid date)

10 = No answer (to a protocol sent to another device)

11 = No valid firmware file

The following general limitations apply to bit combinations:

The answer telegram contains the bits Write, Read, Master, and Error Deleted, if there was no error.

If there is an error, the Error bit is set. The Write, Read and Master bits are deleted.

The Read and Write bits may never be set simultaneously!

The Error bit must not be set if a read or write command follows!

2.4 Master-Slave

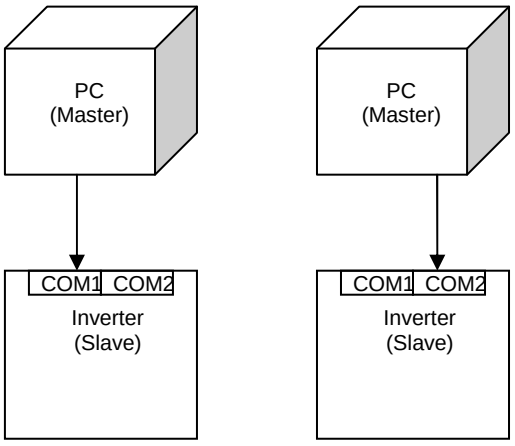
The inverter's Master function is use to network devices via RS485. This lets key data (power, error or status, yield) of connected Slave devices (which may not have displays) be shown on the display of the Master.

The Master is a kind of gateway. A PC or modem can be connected to its second serial RS232 interface to allow remote maintenance.

The inverter provides both Master and Slave functions.

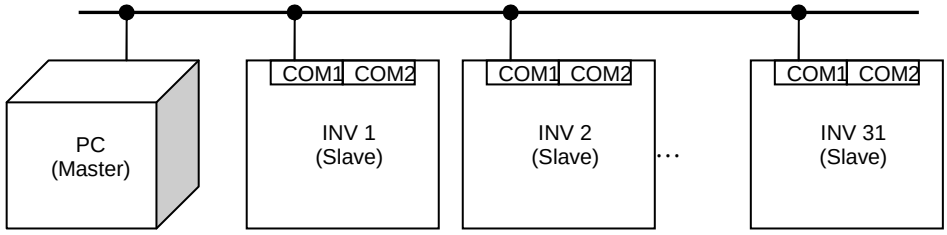
The Master function must be done on a PC, i.e. it has to set the Master bit. The PC's sender address is 0.

2.4.1 PC connection to an external inverter



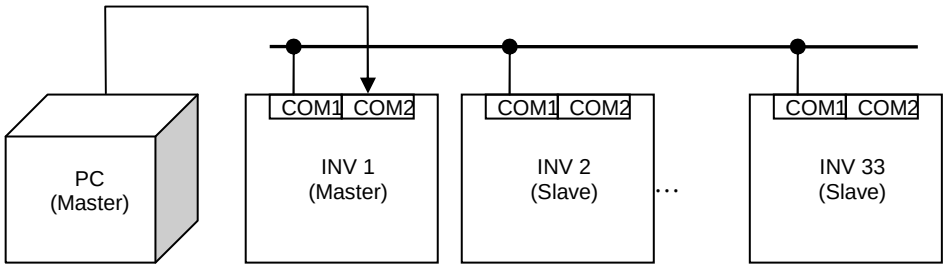
The PC can be connected to any interface (COM 1 or COM 2).

2.4.2 PC connection as Master in the RS485 bus



The PC is connected to the RS485 bus of interface COM1. **None** of the inverters may be configured as master.

2.4.3 PC connection in Master-slave network



The PC (or an external data capture unit) is connected to the master by the COM2 interface (RS232 or RS485 by option card).

A PC (or external data capture unit) may be connected to the system **only for the period of data transfer**.

Thereafter it must end the connection using the Master bit. This tells the master that it must transfer the request internally.

The inverter uses address spoofing to send the response to the PC.

2.5 Broadcast (low power parameter 84 PRED)

If the PRC function is activated (parameter 131, PRCF does not equal 0), cyclically every 10 seconds parameter 84 is written to the COMBOX RS485. RS485_MASTER must be set to 1 for this. The target address of the protocol is 0=Master, Master bit is not set, Write bit is set, source address is the COMBOX slave address. All slaves (SP50 from FW V2.53 SP120 from FW V1.8, SP300 from FW x.yy) read this parameter 84 and set it accordingly. No answer is sent.

2.6 VDEW monitoring

If the inverter detects a network error (U_AC or f_AC) it sets Bit 10 of the parameter number in all telegrams it sends.

If the Master receives this bit, it also sets it in all telegrams for a certain period of time.

A Slave that receives this bit from the Master then disconnects itself from the network.

2.7 Address discovery

So that a PC can recognize the Master's address, the PC sends a valid protocol to "any inverter" i.e. to its own address of 0, in which the Master bit is set, but the Read and Write bits both deleted.

Only the Master answers, i.e. sends a protocol with its own address to the PC.

Once the Master's address has thus been defined, the PC (or external data capture device) can read out parameters 165 to 196 of this master, to get the addresses of the linked Slaves.

3 Characteristics

3.1 Hardware interface

The protocol is designed for serial data transfer. The inverter has a RS232 and RS485 interface.

3.1.1 Interface settings

Baud rate: 19200 Baud

Stop bit: 1

Parity: None

Flow control: None

3.2 Addressability

The protocol has a recipient address and a sender address. This makes it possible to implement communication along multiple inverters as a Master-Slave data network. If a higher-level PC connects to this network, it can take over Master functions (for example for a limited time).

The address is the serial number of the inverter. This can be found on the device's rating plate and the display.

3.3 Security

The protocol includes a checksum (XOR link) for data security.

3.4 Parameter channel

There is just one parameter channel; there is no process data. Larger data volumes (data logger) are organised by free bits of the parameter number (see Section 4 File transfer).

Currently there are about 350 parameters. The protocol can be expanded to a maximum of 1023 parameters. At 16 bits, this leaves 6 bits free for the parameter number.

Bit 15: 0 = No Action; 1 = Write Parameter

Bit 14: 0 = No Action; 1 = Read Parameter

Bit 13: 0 = No Action; 1 = Master Bit (for indirect transfer)

Bit 12: 0 = No Action; 1 = Error Occurred

Bit 11: 0 = No Action; 1 = File Transfer 128 bytes useful data instead of 4 bytes

Bit 10: Function VDEW network monitoring, do not use for PC data connection, value is 0
If set by Slave, Slave will report a network error (UAC, f), PHASE_ERROR_BIT
If set by Master, Master will report to Slave: Switch off

Bits 9..0: Parameter numbers

3.5 Data length

For the sake of simplicity, data are always transferred 32 bits wide.

3.6 Protocol framework

"Start of Text" and "End of Text" as delimiters for transfer.

4 File transfer

The unit provides various lists or files that can be loaded by a connected PC.

For example, this allows transfer of the current parameters or data logger to the PC.

4.1 Parameters used

Number	Text	Description
251	FLS	File 15 block selection: Transferred from PC.
252	DDU	Date/time difference file: Transferred from PC.
260	FN	File number: Transferred from PC to the appropriate inverter.
261	FL	File length: Indicates the number of SP short protocols the application requires for transfer. For example, if a file has 100 bytes, the file length is 25, since a short protocol always consists of 4 bytes.
262	FLI	File read content: The file is transferred through repeated reading of P262. Example: The file contains 100 bytes, i.e. 25 long words. P262 has to be read 25 times.
263	FSI	File read content (used only by SP120 for firmware updates) The file is transferred through repeated reading of P263. Bit 11 of the parameter number (FILEBIT) must be set; the protocol has 128 bytes use data.
264	FI	File index: This parameter is entered by incremental reading from P262 (later: writing from P263). It can also be written directly from the PC. P264 is for monitoring reading (writing). It can be read at the end. After the nth read it is at n.
270	FLIL	File-read content long: In the case of P270, the protocol is not 4 bytes use data long, but 128 bytes. Bit 11 of the parameter number (FILEBIT) must be set in addition. Example: The file contains 1000 bytes, i.e. 250 long words. P270 must then be 250/32 = 7,8125, i.e. read 8 times.
297	DDD	Date difference file: Transferred from PC.

4.2 File read sequence

The application selects the file by writing from P260.

Once the device reads the new value from P260, it starts to assemble the file in its RAM. When it has done so, the value of P261 is available.

The application reads the value of P261 cyclically (1 x per second is enough). As soon as it has a value not equal to zero, file reading can begin.

The application reads the file. The PC reads parameter 262 incrementally (P261 times).

Finally it can check whether it read P261 times and so if the value of P261 is in P264.

Reread the same file: The application first writes parameter 260 to zero, waits at least 2 seconds, and then writes parameter 260 to the desired file number once more. This is because the device reacts only to changes of P260.

4.3 Error detection

The application detects how many blocks of 32 bits each can be read at P261.

The application then reads parameter 261 n times and notes how often it has been read.

At the kth read of P262 the application detects an error.

The application now writes value k to parameter 264.

The application then continues reading P262.

By writing to P264 the sequential counter in the device in corrected.

4.4 File-read content long:

Since reading files as described above is a very slow process, especially in modem downloads, there is also the option of reading 128 bytes at a time instead of 4. Bit 11 of the parameter number must be set for this. Now the application can read P270 instead of P262.

It takes the same action in the event of an error. However, P264 must then be set to the right value.

Example:
The mth protocol at P270 gives a read error.
Now P264 has to be set to the following value: P264 = m*32.

4.5 Difference files

To generate one of the difference files 23, 24, 25, 26 or 27, the parameter 297 (DDD) first has to be written to the inverter.

To generate the difference files the inverter needs a starting date for transferring the values.
It receives the date, in binary code: Year, Month, Day.
Parameter structure: Value = ((year - 2000) * 256 * 256) + (month * 256) + day

Example: The inverter is supposed to later create a file which will hold all yields from 25 Aug. 2007 through today.
To do this, parameter 297 is transferred with the value 460825 (7*65536 + 8*256 + 25).

Careful: Incorrect (invalid) values lead to errors in the difference file!!!
The value of parameter 297 is not stored secure from power outages.

To generate difference files 15, the parameter 252 (DDU) first has to be written to the inverter. The 32 bit parameter DDU contains the seconds since 2000-1-1 00:00:00, including leap years.
The value of parameter 252 is not stored secure from power outages.

4.6 File send sequence

The application selects the file by writing from P260.

The application sets P264 to 0.

The application writes the file length (long words: data length / 4) to P261.

The application now writes the file. The PC writes parameter 263 n times (note: the first answer to the parameter 263 write may take up to 20 seconds). The inverter automatically raises P264.

The application sets P261 to 0.

4.7 File overview

File number	Binary file	DDD file	Description	Length (binary file)
1			Error memory	
2			List of all parameters	
3			List of error codes	
4			List of error counters	
5			List of all actual values	
6			Data logger	
7			Energy year logger (EYL)	
8			Sensor energy year logger (SEYL)	
9			Sensor and energy year logger (SEYL+EYL)	
10			Configuration data logger	
11			Left unassigned for future expansions	
12	X		Complete EEPROM (i.e. includes error	32 Kbytes

File number	Binary file	DDD file	Description	Length (binary file)
			memory, warning memory, EYL, SEYL, data logger, all configurable parameters.	
13			Yield check list	
14			File warning	
15	X	DDU	Datablock universal file	
16			List of error codes, short	
17			List of warnings	
18	X		EYL and SEYL-packed format (useful if all you want to do is read EYL and SEYL quickly)	12 bytes + 366 days x 4 bytes = 1476 bytes
19			List of all parameters that deviate from standard values	
20			Virtual EYL/SEYL (like file 9 but with virtual values)	
21			Left unassigned for future expansions	
22	X		32 bit values of all parameters, binary	14 bytes + (340[1] pars x 4 bytes)
23	X	X	Binary EYL starting DDMMYY until binary today (energy year logger)	14 bytes + 3 bytes DD, MM, YY + n days x 2 bytes
24	X	X	Binary SEYL starting DDMMYY until binary today (sensor energy year logger)	14 bytes + 3 bytes DD, MM, YY + n days x 2 bytes
25	X	X	Binary STSP starting DDMMYY until binary today (error memory)	14 bytes + 3 bytes DD, MM, YY + n entries x 6 bytes
26	X	X	Binary WSP starting DDMMYY until binary today (warning memory)	14 bytes + 3 bytes DD, MM, YY + n entries x 7 bytes
27			Left unassigned for future expansions	
28	X		DCM logger, binary *(SP300 only)*	32 Kbytes
29			DCM logger (SP300 only)	
30			DCM error memory *(SP300 only)*	
31	X		Device status, binary	

4.8 Binary files

To reduce the data volume and thus accelerate transfer, binary files have been introduced. The files contain only the basic data in binary form, and so must be properly interpreted. They are machine-readable only.

[1] The number of parameters depends on the device class (SP25 – SP55 = 340; SP300 = 413)

4.8 **Binary** files

Binary files are available starting with firmware release 2.32 (SP300 1.02).

Before answer queries as to these files, SOLPLUS+ must know whether the firmware release is >= 2.32 (Parameter 149).

Note

Before data evaluation, all binary files must be swapped in the 4-byte block. The sequential bytes must thus be reordered from their 1,2,3,4 order to 4,3,2,1. Binary files always have a number of bytes divisible by 4.

4.8.1 Structure of file 12 (EEPROM binary)

Available on request from Solutronic AG.

4.8.2 Structure of File 15 (datablock universal file)

This file serves to give incoming data a consistent structure with difference date and time (via parameter 252 / DDU).

The file is structured in several blocks, divided by indexes and length information per block. This division into blocks with length information allows the blocks to be extended in the future.

In addition, File 15 offers the ability to select which blocks will be read via the 32 bit parameter 251 (FLS). Each set bit corresponds to a file block (Bit 1 (`b00000000000000000000000000000010`) → Block 0x01; Bit 0 is used).

All data words are transferred with the low byte first.

4.8.2.1 File structure

Index	Datablock length in bytes	Block name
—	15	Standard header block for binary files
0x00	8	File 15 header block
0x01	8 x number of entries	Total error memory
0x02		All parameters with variable length
0x05		Actual value
0x06		Data logger
0x07	4 + (2 x number of days)	EYL
0x08	4 + (2 x number of days)	SEYL
0x09	4 + (2 x number of days)	S0-EYL
0x0A	4 + (2 x number of days)	AEYL
0x0B		Parameter definition
0x0E	7 x number of entries	Warning memory
0x11		Reserved for internal flash
0x13		Non-standard values
0x16		All parameters in 32 bits
0x1A	5 + (5 x number of entries)	System energy logger
0x1B	5 + (5 x number of entries)	Sensor energy logger
0x1C	5 + (5 x number of entries)	S0-energy logger
0x1D		String logger
0x1E		Reserved for internal use

Before each block, except for the standard header block, there is a 4-byte block header with index and block length. The block length is the data length without block header, and can be 0 if no data is present:

Byte 1	Byte 2	Byte 3	Byte 4

Index number	Number of bytes, low	Number of bytes, mid	Number of bytes, high

4.8.2.1.1 Standard header block

Byte 4	Content	Parameter number
1	Day	93
2	Month	94
3	Year, low byte	95
4	Year, high byte	
5	Hour	92
6	Minute	91
7	Second	90
8	Firmware version, low	149
9	Firmware version, high	
10	Serial number, low	148
11	Serial number, high	
12	Device class, low	147
13	Device class, high	
14	File number	(260)
15	File version (reserve)	Current 0x02

The standard header block is always included.

4.8.2.1.2 File 15 header block (Index 0x00)

Byte 4	Content
1 … 4	Date (DDU) in seconds since 2000 (2000-1-1 00:00:00), starting from which the following data are contained.
5 … 8	Block selection (FLS)

The file 15 header block is always included and comes right after the standard header block.

4.8.2.1.3 Total error memory (Index 0x01)

Byte 4	Content
1 … 4	Date (seconds since 2000)
5	String number
6	Error code of an error memory entry
7 … 8	Error attribute
9 … 15	Next error memory entry
⋮	

Note:

The entire error memory block can occur multiple times.
String-independent errors have string number 0.

4.8.2.1.4 All parameters with length (Index 0x02)

Byte	Content
1 … 2	Parameter number (first parameter)
3	Parameter length in bytes (first parameter)
4	Value of first parameter, LL byte
5	Value of first parameter, LH byte
6	Value of first parameter, LH byte (not applicable for 16 bit parameters)
7	Value of first parameter, HH byte (not applicable for 16 bit parameters)
8 … 9	Parameter number (second parameter)
10	Parameter length in bytes (second parameter)
11. …	Value of second parameter
⋮	

Note:

The length is not the parameter length, but the length of the current value, so that only those bytes are transferred that are needed. Length 0 can be used for value = 0.

A length of 0 bytes is given for text parameters. The end of the text is indicated by a 0 byte.

4.8.2.1.5 Actual values (Index 0x05)

Structured like all parameters with length (see 4.8.2.1.4), but contains only the actual values.

4.8.2.1.6 Data logger (Index 0x06)

Byte	Content
1 … 4	Date (seconds since 2000), starting from which the following data logger values apply
5	Data logger interval in min
6	Number of parameters
7 … 8	Parameter number P1
9	Parameter length P1 in bytes
10 … 11	Parameter number P2
12	Parameter length P2 in bytes
⋮	
7 + X x 3	Date in (seconds / 2) since date in bytes 1 … 4 low byte
8 + X x 3	Date in (seconds / 2) since date in bytes 1 … 4 high byte
9 + X x 3	1st logger value, LL byte
10 + X x 3	1st logger value, LH byte
11 + X x 3	1st logger value, LH byte (not applicable for 16 bit parameters)

	1st logger value, HH byte (not applicable for 16 bit parameters)
	2nd logger value, low byte
	2nd logger value, high byte
⋮	

Note:

The data logger block occurs multiple times, at least once for each day. If the interval changes or there are other data logger parameters, there are multiple blocks for the same day.

4.8.2.1.7 EYL (Index 0x07)

Byte	Content
1 … 4	Date (seconds since 2000), starting from which the following values apply
5 … 6	Energy value on day X (byte 1), order low byte to high byte
7 … 8	Energy value on day X +1, order low byte to high byte
⋮	

There can be more than one EYL block if days are missing (the inverter was not in operation).

The values follow the dates in increasing order. The block contains only days that exist in the calendar, meaning that there is 29 February only for leap years.

The file also contains the current day (but its value will change over the course of the day).

The clock time in the DDU is not included in generation of this block.

The values have a decimal place, so that 359 is to be interpreted as 35.9 kWh.

4.8.2.1.8 SEYL (Index 0x08)

As the EYL block, but with sensor yield data. (See 4.8.2.1.7)

4.8.2.1.9 S0-EYL (Index 0x09)

As the EYL block, but with S0 sensor yield data. (See 4.8.2.1.7)

4.8.2.1.10 AEYL (Index 0x0A)

As the EYL block, but with system energy annual data. (See 4.8.2.1.7)

4.8.2.1.11 Parameter definition (Index 0x0B)

Byte	Content
1 … 2	Parameter number
3	Bit 8 … 6 3 x 0-bytes *(reserve)*

	Bit 5 0 = actual value; 1 = set value Bit 4 0 = unsigned; 1 = signed Bit 3 … 1 number of decimal places
4	Length in bytes (length 0 ≙ text)
5 … 6	Parameter number
⋮	

4.8.2.1.12 Warning memory (Index 0x0E)

Byte	Content
1 … 4	Date (seconds since 2000)
5	Warning code
6 … 7	Warning attribute
8 … 13	Next warning memory entry
⋮	

4.8.2.1.13 Non-standard values (Index 0x13)

Structured like all parameters with length (see 4.8.2.1.4). Contains all parameters (without actual values) that deviate from standard values.

4.8.2.1.14 All parameters in 32 bit (Index 0x16)

Contains all parameters (including actual values) and unused numbers as long as they are not at the end.

The unused parameter 0 is not transferred.

1 … 4	Value of parameter No. 1, sequence LL ... HH
5 … 8	Value of parameter No. 2, sequence LL ... HH
⋮	

4.8.2.1.15 System energy logger (Index 0x1A)

Byte	Content
1 … 4	Date (seconds since 2000), starting from which the following data logger values apply
5	Logger interval in min
6	Date in (seconds / 2) since date in bytes 1 … 4 low byte
7	Date in (seconds / 2) since date in bytes 1 … 4 high byte

8	1st data set energy/day system [Wh], low byte
9	1st data set energy/day system [Wh], mid byte
10	1st data set energy/day system [Wh], high byte
11	Date in (seconds / 2) since date in bytes 1 … 4 low byte
12	Date in (seconds / 2) since date in bytes 1 … 4 high byte
13	2nd data set energy/day system [Wh], low byte
14	2nd data set energy/day system [Wh], mid byte
15	2nd data set energy/day system [Wh], high byte
⋮	

Note:

The system energy logger block occurs multiple times, at least once for each day. If the interval changes there are multiple blocks for the same day.

4.8.2.1.16 Sensor energy logger (Index 0x1B)

Like the system energy logger block but with sensor energy data. (See 4.8.2.1.15)

4.8.2.1.17 S0 energy logger (Index 0x1C)

Like the system energy logger block but with S0 energy data. (See 4.8.2.1.15)

4.8.2.1.18 String logger (Index 0x1D)

Byte	Content
1 … 4	Date (seconds since 2000), starting from which the following data logger values apply
5	String logger interval in min
6	Number of strings
7	Number of the first string
8	Number of the second string
⋮	
7 + X	Date in (seconds / 2) since date in bytes 1 … 4 low byte
8 + X	Date in (seconds / 2) since date in bytes 1 … 4 high byte
9 + X	1st data set energy/day system [Wh], low byte
10 + X	1st data set energy/day system [Wh], mid byte
11 + X	1st data set energy/day system [Wh], high byte
12 + X	1st data set device temperature [°C] (signed)

13 + X	1st data set voltage [V] low byte
14 + X	1st data set voltage [V] high byte
15 + X	1st data set current [mA] (signed) low byte
16 + X	1st data set current [mA] (signed) high byte
17 + X	Voltage / current second string; first data set
⋮	
	Voltage / current first string; second data set
⋮	

Note:

The strong logger block occurs multiple times, at least once for each day. If the interval changes or there are other strings, there are multiple blocks for the same day.

4.8.3 Structure of file 18 (EYL/SEYL binary)

Byte	Content
1	Day
2	Month
3	Year, low byte
4	Year, high byte
5	Hour
6	Minute
7	Second
8	Firmware version, low byte
9	Firmware version, high byte
10	Serial number, low byte
11	Serial number, high byte
12	Device class (*low byte only*)
13	EYL value of 1.1., low byte
14	EYL value of 1.1., high byte
15	SEYL value of 1.1., low byte
16	SEYL value of 1.1., high byte
17	EYL value of 2.1., low byte
18	EYL value of 2.1., high byte
⋮	

Notes:

0 bytes can follow at the end.

The file also contains the current day (but its value will change over the course of the day ...).

The dates 30 Feb., 31 Feb., 31 April, 31 June, 31 Sep., and 31 Nov. are not transferred

For each year there is a 29 Feb. in the transfer, but the internal real-time clock ensures that the date is correct in non-leap years. This means that the evaluation software must ignore the value of 29 Feb. in non-leap years. In non-leap years the inverter reports a yield of 0 on 29 Feb.

The values have a decimal place, so that 359 is to be interpreted as 35.9 kWh.

4.8.4 Structure of file 22 (all parameters 32 bit)(/M)

Byte	Content
1	Day
2	Month
3	Year, low byte
4	Year, high byte
5	Hour
6	Minute
7	Second
8	Firmware version, low byte
9	Firmware version, high byte
10	Serial number, low byte
11	Serial number, high byte
12	Device class, low byte
13	Device class, high byte
14	File number, i.e. 22 binary (00010110)
15 … 18	Value of parameter 1, sequence LL ... HH
19 … 22	Value of parameter 2, sequence LL ... HH
⋮	

Notes:

Interpretation of the values: Information as to whether a parameter is 16 or 32 bits long, or should be interpreted as signed or unsigned, is in SOLPLUS+ and must be taken from there.

The introduction of file 22 reduces the number of bytes in file transfer from around 15688 to typically 1374 (parameters 1 through 340).[2]).

The number of file blocks is reduced from 123 to 11.

[2] The number of parameters depends on the device class (SP25 – SP55 = 340; SP300 = 413)

If it were necessary to transfer all parameters individually, it would mean transmitting 340 (short telegram) blocks.

4.8.5 Structure of file 23 (binary EYL starting from DDD) (/N)

Byte	Content
1	Day
2	Month
3	Year, low byte
4	Year, high byte
5	Hour
6	Minute
7	Second
8	Firmware version, low byte
9	Firmware version, high byte
10	Serial number, low byte
11	Serial number, high byte
12	Device class, low byte
13	Device class, high byte
14	File number, i.e. 23 binary (00010111)
15	Day starting from which the following EYL values come
16	Month starting from which the following EYL values come
17	Year starting from which the following EYL values come
18 … 19	Value of the energy year logger on day DDD, sequence low to high byte
20 … 21	Value of the energy year logger on day DDD+1, sequence low to high byte
⋮	

Notes:

0 bytes can follow at the end.

The file also contains the current day (but its value will change over the course of the day ...).

The dates 30 Feb., 31 Feb., 31 April, 31 June, 31 Sep., and 31 Nov. are not transferred.

For each year there is a 29 Feb. in the transfer. The internal real-time clock ensures that the date is correct in non-leap years. This means that the evaluation software must ignore the value of 29 Feb. in non-leap years. In non-leap years the inverter reports a yield of 0 on 29 Feb.

The values have a decimal place, so that 359 is to be interpreted as 35.9 kWh.

Example:

Today is 30 Aug. 2008.

DDD is set to 3 Sep. (2007).

The inverter stores values over 1 year in a ring buffer in its energy year logger. Today it contains values from 31 Aug. 2007 (!) to 30 Aug. 2008.

In this example, the first yield value to be transferred is that of 3 Sep. 2007. The last is that of 30 Aug. 2008.

4.8.6 Structure of file 24 (binary EYL starting from DDD) (/O)

Byte	Content
1	Day
2	Month
3	Year, low byte
4	Year, high byte
5	Hour
6	Minute
7	Second
8	Firmware version, low byte
9	Firmware version, high byte
10	Serial number, low byte
11	Serial number, high byte
12	Device class, low byte
13	Device class, high byte
14	File number, i.e. 24 binary (00011000)
15	Day starting from which the following SEYL values come
16	Month starting from which the following SEYL values come
17	Year starting from which the following SEYL values come
18 … 19	Value of the sensor energy year logger on day DDD, sequence low to high byte
20 … 21	Value of the sensor energy year logger on day DDD+1, sequence low to high byte
⋮	

Notes:

0 bytes can follow at the end.

The file also contains the current day (but its value will change over the course of the day ...).

The dates 30 Feb., 31 Feb., 31 April, 31 June, 31 Sep., and 31 Nov. are not transferred.

For each year there is a 29 Feb. in the transfer, but the internal real-time clock ensures that the date is correct in non-leap years. This means that the evaluation software must ignore the value of 29 Feb. in non-leap years. In non-leap years the inverter reports a sensor yield of 0 on 29 Feb.

The values have a decimal place, so that 359 is to be interpreted as 35.9 kWh.

The structure of files 23 and 24 is basically the same.

Example:

Today is 30 Aug. 2008.

DDD is set to 3 Sep. (2007).

The inverter stores values over 1 year in a ring buffer in its sensor energy year logger. Today it contains values from 31 Aug. 2007 (!) to 30 Aug. 2008.

In this example, the first yield value to be transferred is that of 3 Sep. 2007, and the last is the sensor yield of 30 Aug. 2008.

4.8.7 Structure of file 25 (binary error memory starting from DDD) (/P)

Byte	Content
1	Day
2	Month
3	Year, low byte
4	Year, high byte
5	Hour
6	Minute
7	Second
8	Firmware version, low byte
9	Firmware version, high byte
10	Serial number, low byte
11	Serial number, high byte
12	Device class, low byte
13	Device class, high byte
14	File number, i.e. 25 binary (00011001)
15	Day starting from which the following error memory values come
16	Month starting from which the following error memory values come
17	Year starting from which the following error memory values come
18 … 23	Day, month, (year-2000), hour, minute, error code of an error memory entry, starting with DDD
24 … 29	Next error memory entry
⋮	

Notes:

0 bytes can follow at the end.

If there are no error memory entries on day DDD, the values for the next day are sent, etc.

If there are multiple entries for the same day, all are transferred, starting with the oldest.

The sequence in file 25 is thus exactly opposite from that in file 1 (error memory).

The year is transferred without 2000, i.e. 7 = 2000.

4.8.8 Structure of file 26 (binary warning memory starting from DDD) (/Q)

Byte	Content
1	Day
2	Month
3	Year, low byte
4	Year, high byte
5	Hour
6	Minute

7	Second
8	Firmware version, low byte
9	Firmware version, high byte
10	Serial number, low byte
11	Serial number, high byte
12	Device class, low byte
13	Device class, high byte
14	File number, i.e. 26 binary (00011010)
15	Day starting from which the following warning memory values come
16	Month starting from which the following warning memory values come
17	Year starting from which the following warning memory values come
18 … 24	Day, month, (year-2000), hour, minute, error code of a warning memory entry, starting with DDD
25 … 31	Next warning memory entry
⋮	

Notes:

0 bytes can follow at the end.

If there are no warning memory entries on day DDD, the values for the next day are sent, etc.

If there are multiple entries for the same day, all are transferred, starting with the oldest.

The sequence in file 26 is thus exactly opposite from that in file 14 (error memory).

The year is transferred without “2000” i.e. 7 = 2000.

4.8.9 Structure of file 28 (binary DCM-logger) (/S)

Byte	Content
1 … 4	Time stamp: Day, month, year, hours, minutes (`DDDDDMMM MYYYYYYYhhhhhmmm mmm00000`)
5 … 7	DCM 1: Voltage (U) and current (I) (`UUUUUUUU UUIIIIII IIIIIIII`)
8 … 10	DCM 2: Voltage and current
11 … 13	DCM 3: Voltage and current
14 … 16	DCM 4: Voltage and current
17 … 19	DCM 5: Voltage and current
20 … 22	DCM 6: Voltage and current
23 … 25	DCM 7: Voltage and current
26 … 28	DCM 8: Voltage and current
29 … 31	DCM 9: Voltage and current
32 … 34	DCM 10: Voltage and current
35 … 37	DCM 11: Voltage and current
38 … 40	DCM 12: Voltage and current
41 … 43	DCM 13: Voltage and current

44 … 46	DCS (16): Voltage and current
47 … 50	Time stamp
51 … 53	DCM 1: Voltage and current
54 … 56	DCM 2: Voltage and current
⋮	
32750 … 32752	DCS (16): Voltage and current
32753 … 32768	Unused

Notes:

Careful: The time stamps are not sequentially sorted, since the inverter stores data in a ring buffer for the period of a year.

4.8.10 Structure of file31 (binary device status)

SOLPLUS+ provides a binary file that generates a list of all important device information, depending on the device model. This makes it possible to get a quick assessment of a system with SOLPLUS+. This file contains more information that with previous versions, and reduces waiting time for the information, especially when using a modem.

The file is structured in several blocks, divided by indexes and length information per block. This division into blocks with length information allows the blocks to be extended in the future.

All data words are transferred with the low byte first.

4.8.10.1 File structure

Index	Datablock length in bytes	Block name
—	15	Standard header block for binary files
0x01	27	General block
0x02	30	SP50
0x03	47	SP300
0x04	15	DCM block
0x05	74	SP120
0x06	26	System performance
0x07	26	Sensor
0x08	23	S0 energy meter
0x09	4	COMBOX

Before each block, except for the standard header block, there is a 3-byte block header with index and block length.

Byte 1	Byte 2	Byte 3
Index number	Number of bytes, low	Number of bytes, high

4.8.10.1.1 Standard header block

Byte	Content	Parameter number
1	Day	93
2	Month	94
3	Year, low byte	95

4	Year, high byte	
5	Hour	92
6	Minute	91
7	Second	90
8	Firmware version, low	149
9	Firmware version, high	
10	Serial number, low	148
11	Serial number, high	
12	Device class, low	147
13	Device class, high	
14	File number	(260)
15	File version (reserve)	Currently 0x01 for expansion of Index 2 (SP50) ,3 (SP300) ,5 (SP120) 0x03

4.8.10.1.2 General block (Index 0x01)

Byte	Content	Parameter number
1	Device status	32
2		
3	Status 2	31
4		
5	Status 1	30
6	"	
7	Energy/day	8
8	"	
9	"	
10	"	
11	Energy/week	9
12	"	
13	"	
14	"	
15	Energy/month	10
16	"	
17	"	
18	"	
19	Energy/year	11
20	"	
21	"	
22	"	
23	Energy total	12
24	"	
25	"	

26	"	
27	Final error	146

4.8.10.1.3 SP50 block (Index 0x02)

Byte	Content	Parameter number
1	Nominal DC output	272
2	"	
3	AC output	5
4	"	
5	Grid frequency	15
6	"	
7	UDC	2
8	"	
9	UAC	1
10	"	
11	IDC	4
12	"	
13	IAC	3
14	"	
15	Device temperature	16
16	"	
17	Grid impedance	13
18	"	
19	Fault current	24
20	"	
21	Irradiation	19
22	"	
23	Module temperature	17
24	"	
25	Sensor yield	217
26	"	
27	RS485 address SPP	89 ***(as of file version 0x02)***
28	"	
29	"	
30	"	

4.8.10.1.4 SP300 block (Index 0x03)

Byte	Content	Parameter number
1	Nominal DC output	272

2	"	
3	AC output	5
4	"	
5	Number of DC modules	287
6	UAC1	13
7	"	
8	UAC2	14
9	"	
10	UAC3	15
11	"	
12	UACpeak1	24
13	"	
14	UACpeak2	25
15	"	
16	UACpeak3	26
17	"	
18	PAC1	1
19	"	
20	PAC2	2
21	"	
22	PAC3	3
23	"	
24	fGrid1	77
25	"	
26	fGrid2	78
27	"	
28	fGrid3	79
29	"	
30	INV1 temperature	70
31	"	
32	INV2 temperature	71
33	"	
34	INV3 temperature	72
35	"	
36	DCM temperature	85
37	"	
38	FP temperature	88
39	"	
40	DC link voltage	73
41	"	

42	Deviation of DC link mean from N	74
43	"	
44	RS485 address SPP	89 ***(as of file version 0x02)***
45	"	
46	"	
47	"	

4.8.10.1.5 DCM block (Index 0x04)

Byte	Content	Parameter number
1	DCM address	-
2	UPV	DCM: 2 (233)
3	"	
4	IPV	DCM: 3 (234)
5	"	
6	Status	DCM: 7 (238)
7	Error	
8	DC module temperature	DCM: 5 (235)
9	"	
10	DC module yield	DCM: 15 (400)
11	"	
12	"	DCM: 14 (400)
13	"	
14	FW version	DCM: 20 (109)
15	"	

Note:

There is one DCM block for each DCM installed (only with SP300).

4.8.10.1.6 SP120 block (Index 0x05)

Byte	Content	Parameter number
1st	Nominal DC output	272
2	"	
3	AC output	5
4	"	
5	UAC1	13
6	"	
7	UAC2	14
8	"	
9	UAC3	15
10	"	
11	UACpeak1	24

12	"	
13	UACpeak2	25
14	"	
15	UACpeak3	26
16	"	
17	PAC1	1
18	"	
19	PAC2	2
20	"	
21	PAC3	3
22	"	
23	fGrid1	77
24	"	
25	fGrid2	78
26	"	
27	fGrid3	79
28	"	
29	INV1 temperature	70
30	"	
31	INV2 temperature	71
32	"	
33	INV3 temperature	72
34	"	
35	UDC1	231
36	"	
37	UDC2	232
38	"	
39	UDC3	233
40	"	
41	IDC1	234
42	"	
43	IDC2	235
44	"	
45	IDC2	236
46	"	
47	Error INV 1	237
48	Error INV 2	238
49	Error INV 3	239
50	Status INV1	240
51	Status INV2	241

52	Status INV3	242
53	Yield INV1	400
54	"	
55	"	
56	"	
57	Yield INV2	401
58	"	
59	"	
60	"	
61	Yield INV3	402
62	"	
63	"	
64	"	
65	RS485 address SPP	89 ***(as of file version 0x02)***
66	"	
67	"	
68	"	
69	Firmware version DC string 1 (Slave 3)	157
70	"	
71	Firmware version DC string 2 (Slave 2)	156
72	"	
73	Firmware version DC string 3 (Master)	149 ***(as of file version 0x03)***
74	"	

4.8.10.1.7 System output (Index 0x06)

Byte	Content
1st	System nominal DC output [W] ***(as of file version 0x03)***
2	
3	
4	Current system output [W]
5	
6	
7	System energy/day [Wh]
8	"
9	"
10	"
11	System energy/week [kWh]
12	"
13	"
14	"

15	System energy/month [kWh]
16	"
17	"
18	"
19	System energy/year [kWh]
20	"
21	"
22	"
23	System energy total [kWh]
24	"
25	"
26	"

4.8.10.1.8 Sensor (Index 0x07)

Byte	Content
1st	Nominal DC output [W] ***(as of file version 0x03)***
2	
3	
4	Current sensor performance [W]
5	
6	
7	Energy/day sensor [Wh]
8	"
9	"
10	"
11	Energy/week sensor [kWh]
12	"
13	"
14	"
15	Energy/month sensor [kWh]
16	"
17	"
18	"
19	Energy/year sensor [kWh]
20	"
21	"
22	"
23	Energy total sensor [kWh]
24	"
25	"

26	"

4.8.10.1.9 S0 energy meter (Index 0x08)

Byte	Content
1st	Current sensor output [W] ***(as of file version 0x03)***
2	
3	
4	Energy/day S0 [Wh]
5	"
6	"
7	"
8	Energy/week S0[kWh]
9	"
10	"
11	"
12	Energy/month S0 [kWh]
13	"
14	"
15	"
16	Energy/year S0 [kWh]
17	"
18	"
19	"
20	Energy total S0 [kWh]
21	"
22	"
23	"

4.8.10.1.10 COMBOX (Index 0x09)

1	RS485 address SPP	89 ***(as of file version 0x03)***
2	"	
3	"	
4	"	

5 Note for RS485

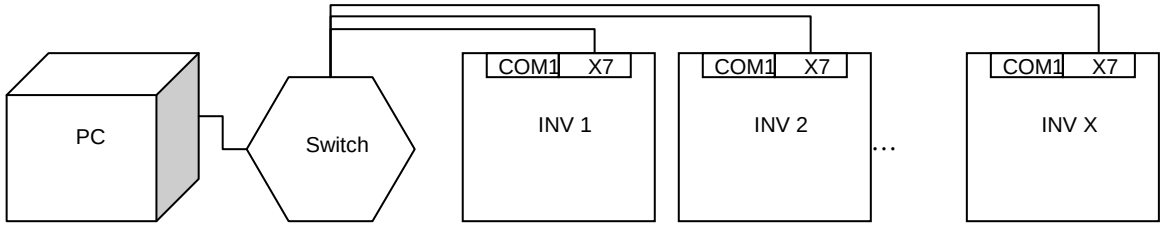
To implement the protocol on an RS485, deactivate the echo mode of the driver. Otherwise the echo will have to be treated.

6 Access to files and parameters via Ethernet

The Solutronic protocol (see also Section 2) is packaged in TCP for Ethernet communication.

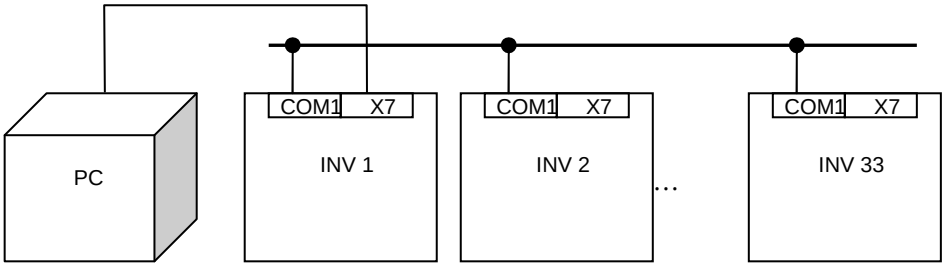
The IP address of the inverter can be set at parameters 100 through 113, and the port at parameter 208 (the standard port is 33330), the subnet mask at parameters 114 to 117, and the standard gateway at parameters 118 to 121.

6.1 PC connection as Master



All inverters are connected directly to the network by the Ethernet interface (X7).

6.2 PC connection in Master-Slave network



The PC (or an external data capture unit) is connected to an inverter in the data network by the Ethernet interface (X7).

The inverter the PC is connected to transfers all requests not meant for it to the other devices by the COM1 interface. The inverter uses address spoofing to send the response back to the PC.

6.3 COMBOX broadcast performance reduction

In order to supply all inverters in a local network with parameter PRED (P184), a UDP broadcast is sent to the IP broadcast address and port 33330 with the content SPP-PRM-SND.

Each inverter that receives such a UDP package at port 33330 (regardless of the port where it is configured with parameter 298), accepts the parameter.

[PARAMNR-H] = port high byte
[PARAMNR-L] = port low byte

[WERTHH] = value high high
[WERTHL] = value high low
[WERTLH] = value low high
[WERTLL] = value low low

[FILLER] = fillbyte for even number (until now set at 00h)

6.4 Inverter search by broadcast

In order to search all inverters in a local network, a UDP broadcast is sent to the IP broadcast address and port 33330 with the content SPP-ADP-REQ.

Each inverter that receives such a UDP package at port 33330 (regardless of the port where it is configured with parameter 298), then sends SPP-ADP-ACK [SN-H][SN-L] [FWV-H][FWV-L] [IPHH][IPHL][IPLH][IPLL] [PORT-H][PORT-L] [SPPHH][SPPHL][SPPLH][SPPLL] to the sender IP.

[SN-H] = serial number, high byte
[SN-L] = serial number, low byte
[FWV-H] = firmware version, high byte
[FWV-L] = firmware version, low byte
[IPHH] = IP address high high
[IPHL] = IP address high low

6.4 Inverter search by broadcast

[IPLH] = IP address low high
[IPLL] = IP address low low
[PORT-H] = port high byte
[PORT-L] = port low byte

[SPPHH] = RS485 address SPP high high
[SPPHL] = RS485 address SPP high low
[SPPLH] = RS485 address SPP low high
[SPPLL] = RS485 address SPP low low

[GK-H] = device class, high byte
[GK-L] = device class, low byte

[OEM-H] = OEM code, high byte
[OEM-L] = OEM code, low byte

Example: The answer from an inverter with serial number 1220, firmware version 2.45, IP 192.168.0.1 and port 33330 and RS485 address SPP 136, device class 1 = COMBOX, OEM code looks like this in hex notation:

```
53 50 50 2D 41 44 50 2D 41 43 4B  04 C4  00 F5  C0 A8 00 01  82 32  88 00 00 00 01 00 00 00
S  P  P  -  A  D  P  -  A  C  K   1220   245    192 168 0 1  33330  136         0001  0000
```