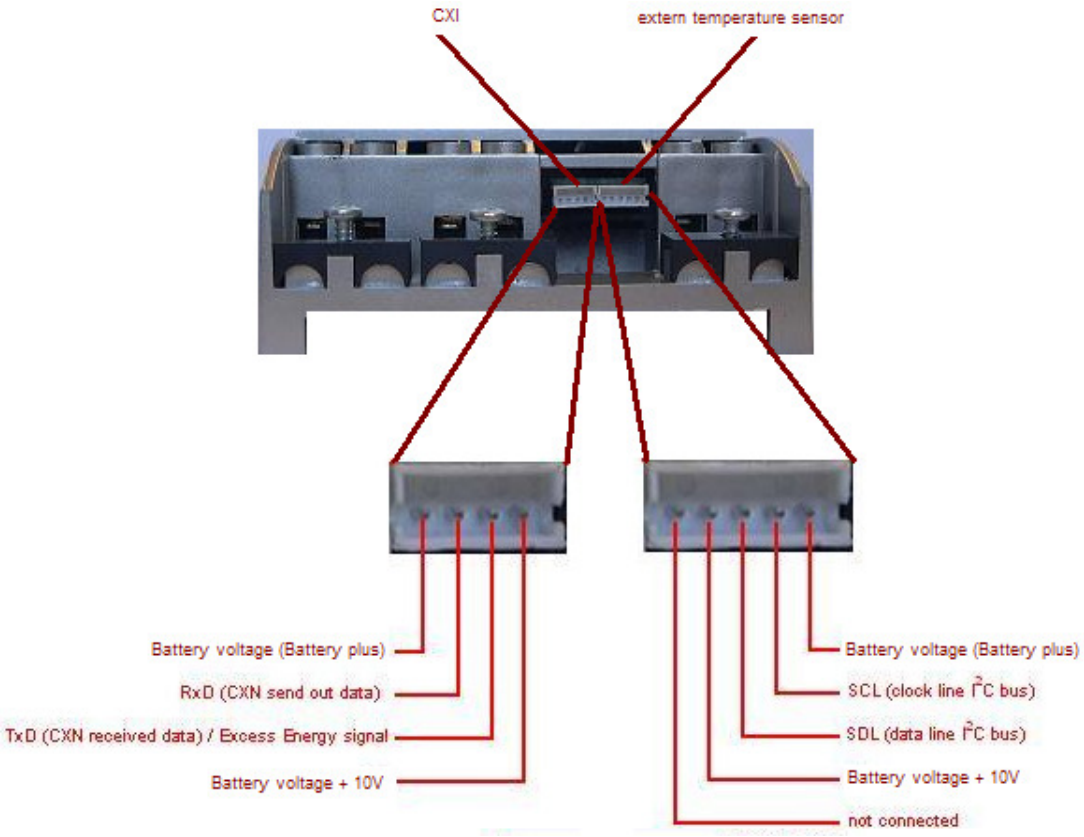
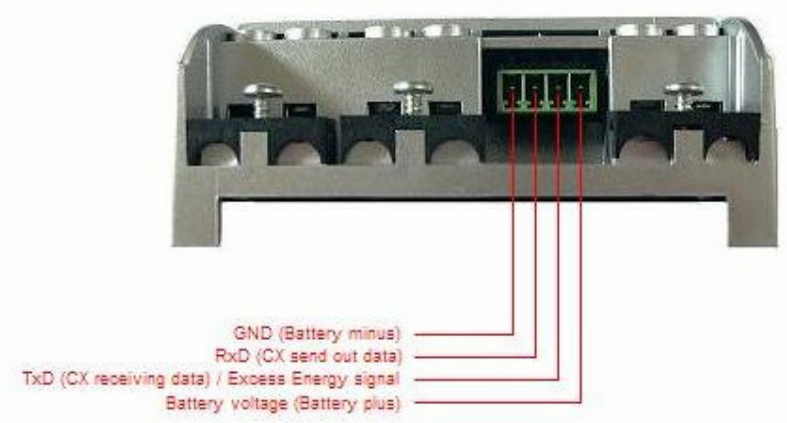
Explanation DATALOGGER and ACTUAL DATA of CXN and CX Controller

Interface CX / CXN Controller:



Picture 1:CXN Interface socket



Picture 2: CX Interface socket

Data Interface socket (the left socket 4 pins by CXN)

The left socket is for using the CXM, CXI/CXCOM or EEM (Excess Energy Management).

Using the left socket for data transmission CXI/CXCOM (or CXM only CXN)

The controller can receive and sending out asciicharacters with following transfer settings

- Transfer rate 9600 Baud (Bit/s)
- 8 Data bit
- 1 Stopp bit
- No parity bit

On the sending and receiving line where logical 0 and logical 1 transmitted as you can see as following:

- Logical 0 → CX=5V by CXN = battery voltage + 5V
- Logical 1 → CX=0V by CXN = battery voltage + 0V

Using the left socket for EEM (Excess Energy Management)

Set point 8.1 or 8.2 in the controller menu to use this function.

Frequenz of the excess energy signal = 12,5Hz

The excess energy signal is a PWM (pulse width modulated) signal between 0V and 5V. The average of this signal gives you the value of the excess energy. If the average is 5V, than you have 100% excess energy, if the average is 0V, than you have no excess energy.

At **CX controller** you can measure the signal between **TxD and GND** and at the **CXN controller** you can measure the signal between **TxD and battery voltage** line of the left socket, when you have set the controller on point 8.1 or 8.2 (see CXN manual)

Temperature sensor socket (the right socket with 5 pins only by CXN)

The left socket is used for the extern temperature sensor. The data where transmitted by the I^2C bus.

Data of the CXN and CX

If you send the ASCII character exclamation mark "!" (Asciicharacter 33) you get the **DATALOGGER DATA**, and if you send a Space " "(Asciicharacter 32) you get the **ACTUAL DATA**.

Please notice that the CX / CXN controller send back as echo the "!" respectively the space to confirm that communication worked.

The CX respectively CXN always send out Asciicharacters.

All Asciicharacters which the Controller send out respectively you need to program the Controller can you see on the following table:

Asciicharacter	In Byte	Asciinumber in decimal
0	0011 0000	48
1	0011 0001	49
2	0011 0010	50
3	0011 0011	51
4	0011 0100	52
5	0011 0101	53
6	0011 0110	54
7	0011 0111	55
8	0011 1000	56
9	0011 1001	57
A	0100 0001	65
B	0100 0010	66
C	0100 0011	67
D	0100 0100	68
E	0100 0101	69
F	0100 0110	70
+	0010 1011	43
-	0010 1101	45

Asciicharacter	In Byte	Asciinumber in decimal
Space	0010 0000	32
!	0010 0001	33
%	0010 0111	37
c	0110 0011	99
x	0111 1000	120
P	0101 0000	80
H	0100 1000	72
O	0100 1111	79
G	0100 0111	71
I	0100 1001	73
J	0100 1010	74
K	0100 1011	75
L	0100 1100	76
M	0100 1101	77
N	0100 1110	78
Q	0101 0001	81
A	0100 0001	65

table 1 ASCII table

In the whole document:

Rating of the bits:

Bit0 → 1
Bit1 → 2
Bit2 → 4
Bit3 → 8
Bit4 → 16
Bit5 → 32
Bit6 → 64
Bit7 → 128

Values

The **ACTUAL DATA** is composed always with Asciicharacters which represent decimal numbers (0,1,2,3,4,5,6,7,8,9)

The **DATALOGGER DATA** is composed always with Asciicharacters which represent hex numbers (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

The values always separated with Ascii character 32 (Space).

The Asciicharacter which is send first has always the highest rating. In the datalogger it can be that a values is composed by two, three or six parts. Then part1 has always the highest rating.

Data Read out with Hyper Terminal:

Example ACTUAL DATA CX:

```
064 023 004 003 146 160 000 000 004 000 120 000 +000 +000 229
```

Example ACTUAL DATA CXN:

```
020 004 005 107 170 161 000 000 255 015 120 021 -003 -003 000 0000 0000 001 000
```

Example DATALOGGER:

```
!F5 AF 6A 08 0B D7 00 01 1F 7D 0A 03 01 08 8E FF FF FF 28 12 00 07 00 00 04 0C 00
 02 A0 00 01 F0 00 5C 60 5E 00 00 05 00 00 22 02 5E 5D 00 00 05 00 00 22 02 5E 5E
 00 00 05 00 00 11 02 60 5E 00 00 05 00 00 22 02 62 61 00 00 05 00 00 44 02 5F 5E
 00 00 05 00 00 22 02 70 70 00 00 05 00 00 BB 02 62 61 00 00 05 00 00 44 02 77 72
 00 03 01 00 00 CD 02 92 6F 05 03 03 2D 0C CC 02 61 5E 00 00 04 00 00 44 03 A1 72
 0A 06 04 63 1C DD 02 92 6A 08 07 06 4B 14 AA 03 83 6C 03 02 02 1B 07 AA 03 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
 FF FF FF FF 40 3C 38 00 00 01 07 5C 02
```

ACTUAL DATA:

Send the Ascii character 32 (Space) and you will get the **ACTUAL DATA**.

Please notice that the CX / CXN controller send back as echo the Space to confirm that communication worked.

Value number	Valuename	Number
1	VERSIONNUMBER_CX_CXN	1
		2
		3
		4
2	SOC_CX_CXN	5
		6
		7
		8
3	DISCONADJ_CX_CXN	9
		10
		11
		12
4	BATTERY_VOLTAGE_CX_CXN	13
		14
		15
		16
5	BATTERY_END_CHARGE_CX_CXN	17
		18
		19
		20
6	STATUS_CX_CXN	21
		22
		23
		24
7	LOAD_AH_DAY_CXN	25
	LOADCURRENT_CX	26
		27
		28
8	VOLTAGE_BATTERY_WIRES_CX_CXN	29
		30
		31

		32
9	PWM_CX_CXN	33 34 35
		36
10	NIGHTHOURS_CX_CXN	37 38 39
		40
11	NIGHTHOURS_LAST_NIGHT_CX_CXN	41 42 43
		44
12	SPECBITS_CX_CXN	45 46 47
		48
13	TEMPERATURE_INTERN_CX_CXN	49 50 51 52
		53
14	TEMPERATURE_ENVIRONMENTAL_CX_CXN	54 55 56 57
		58
15	PV_AH_DAY_CXN PV_CURRENT_CX	59 60 61
		62
16	PV_CURRENT_CXN (Only CXN)	63 64 65 66
		67
17	LOAD_CURRENT_CXN (Only CXN)	68 69 70 71

		72
18	REASON_LOAD_OFF_CXN (Only CXN)	73 74 75
		76
19	EXCESS_AH_DAY_CXN (Only CXN)	77 78 79

table 2: ACTUAL DATA

The most **ACTUAL DATA** values are current values of the chargecontrollers.

The **ACTUAL DATA** values are all composed with several Ascii characters. The first one has always the highest rating.

A Space (Ascii character 32) will separate all the values.

The most of the **ACTUAL DATA** are decimal number system.

1. VERSIONNUMBER_CX_CXN

versionnumber of the microcontroller software from the controller
→ 3 Ascci characters

2. SOC_CX_CXN

Actual SOC (Battery State of charge)
Dimension: %
→ 3Ascii characters

If **SOC_CX_CXN** = 0 expect 0% *state of charge*

The Soc value depends on the battery protection setting. Look at the DATALOGGER value 14 (**MENUSTATE_CX_CXN**) there you can see the battery deep discharge setting.

- By setting: "Low voltage disconnect current compensated 11.4V-11.9V"
  SOC_CX_CXN = 30 accord → Battery *SOC* = 100%

- By setting: "Low voltage disconnect current compensated 11.0V–11.75V"
  SOC_CX_CXN = 35 accord → Battery *SOC* = 100%

- By setting:Low voltage disconnect current compensated/adaptive 11.0V-12.2V
  SOC_CX_CXN = 35 accord → Battery *SOC* = 100%

By setting load disconnect by a fixed voltage the value SOC_CX_CXN shows you the voltage difference
(**BATTERY_VOLTAGE_CX_CXN** – 11,0V respectively 11,5V)

Voltage_difference_load_disconnect = SOC_CX_CXN * 0,032V

To show it as battery state of charge (fully charged by 12,8V)

- By setting: Low voltage disconnect 11.5V
  SOC_CX_CXN = 41 accord → *SOC* = 100%
- By setting: Low voltage disconnect 11.0V
  SOC_CX_CXN = 57 accord → *SOC* = 100%

Example: → **SOC_CX_CXN** = 25
Battery protection setting: Low voltage disconnect 11.0V

→ *SOC* = 44%

3. DISCONADJ_CX_CXN

This value increase every day when battery is not fully charged (only by setting: Low voltage disconnect current compensated/adaptiv 11.0V-12.2V(datalogger value 13)
→ 3Ascii characters

If **DISCONADJ_CX_CXN** = **SOC_CX_CXN** the load will disconnected.

4. BATTERY_VOLTAGE_CX_CXN

Current battery voltage
Dimension = Volt
→3 Ascii characters

Calculation:

By 12V system:
Battery_voltage = (BATTERY_VOLTAGE_CX_CXN * 0,032) + 9

By 24V system:
Battery_voltage = 2 * [(BATTERY_VOLTAGE_CX_CXN * 0,032) + 9]

Look at value 6 (STATE_CX_CXN) there you can see if the controller work at a 12V or 24V system.

Example: battery_voltage = 114 → 12.6V (by 12V system)

5. END_CHARGE_CX_CXN

Voltage which the chargecontroller controll the battery voltage.
Dimension: Volt
→ 3Ascii characters

Calculation:

12V system:
Voltage_end_charge = (**END_CHARGE_CX_CXN** * 0,032) + 9

24V system:
Voltage_end_charge = 2 * [(**END_CHARGE_CX_CXN** * 0,032) + 9]

Look at value 6 (**STATUS_CX_CXN**) there you can see if the controller work at a 12V or 24V system.

Example: END_CHARGE_VOLTAGE = 151 → 13.8V (by 12V system)

6. STATUS_CX_CXN

Shows you some status bits
→3Ascii characters

Convert the decimal number into a binary number:
Bit0 = 0: BOOST Mode: off
Bit0 = 1: charging in the BOOST Mode
Bit1 = 0: EQUAL Mode: off
Bit1 = 1: charging in the EQUAL Mode
Bit2 = 0: 12V system
Bit2 = 1: 24V system
Only by CXN:
Bit7 = 0: external temperature sensor is connected
Bit7 = 1: external temperature sensor is not connected

BOOST Mode: battery charging by 14,5V
EQUAL Mode: battery charging by 14,8V

Example: STATE_CX_CXN = 160_{10} = 10100001_2 →
Bit0 = 1 → BOOST MODE: on
Bit1 = 0 → EQUAL MODE: off
Bit2 = 0 → 12V System
Bit7 = 1 → external temperature sensor is not connected (only CXN)

7. LOAD_AH_DAY_CXN / LOAD_CURRENT_CX

LOAD_AH_DAY_CXN: Load Amperehours of the actual day
Dimension: Amperehours (Ah)

LOAD_CURRENT_CX: actual load current
Dimension: percent of nominal current

→ 3Asciizeichen

CX:

FULLCUR_CX you see at **DATALOGGER** value (number 3).

If **FULLCUR_CX** > 50 then → CX10: x = 10
If **FULLCUR_CX** > 23 and **FULLCUR_CX** < 30 then → CX20: x = 20
If **FULLCUR_CX** < 22 then → CX40: x = 40

Load_current = (**LOAD_CURRENT_CX / FULLCUR_CX**) * x

CXN:

If CXN10 → x=10
If CXN20 → x=20
If CXN40 → x=40

The CXN Type you see by **the DATALOGGER** value (number 19)

Load_Ah = (**LOAD_AH_DAY_CXN** * x * 4) / 60

8. VOLTAGE_BATTERY_WIRES_CX_CXN

Shows you the voltage drop on the wires to the battery
→ 3Asciizeichen

9. PWM_CX_CXN

If the System has excess energy because the battery is fully charged. The PV current will reduced with pulse width modulation. The relation of pwm will shown you here.
→ 3Asciizeichen

The value is between 0 and 255:
Ratio_used_PV_current_in_% = (**PWM_CX_CXN** / 255) * 100%

Example:
PWM_CX_CXN = 200 → 78% of the PV current is used at the moment.

10. NIGHTHOURS_CX_CXN

Shows you the hours since the night begun when it is night.
Dimension: hours
→ 3Asciizeichen

Nighthours_final_value = **NIGHTHOURS_CX_CXN** / 10

Example: NIGHTHOURS_CX_CXN = 10 so it's for 1h night

11. NIGHTHOURS_LAST_NIGHT_CX_CXN

Shows you the hours of the last night.
Dimension: hours
→ 3Asciizeichen

Nighthours_last_night = **NIGHTHOURS_LAST_NIGHT_CX_CXN** / 10

Example: **NIGHTHOURS_LAST_NIGTH CX_CXN** = 120
→nighthours last night = 12hours

12. SPECBITS_CX_CXN

Convert the decimal number into a binary number:
→ 3Ascii characters

if Bit2 = 1 Load output off
if Bit2 = 0 Load output on
if Bit4 = 1 night at the moment
if Bit4 = 0 day at the moment

13. TEMPERATURE_INTERN_CX_CXN

Shows you the intern temperature of the controller
Dimension: °C
→4 Ascii characters

The first Ascii character is „+“ or „-“ and shows you if you should

Temperature = 25 -**TEMPERATURE_INTERN_CX_CXN**

Respectively:

Temperature = 25 +**TEMPERATURE_INTERN_CX_CXN**

Example:
TEMPERATURE_INTERN_CX_CXN = -009 → 25-9= 16°C

14. TEMPERATURE_ENVIRONMENTAL_CX_CXN

Shows you the temperature outside of the controller
Dimension: °C
→ 4 Ascii characters

The first Ascii character is „+“ or „-“ and shows you if you should add or subtract.

Temperature = 25 -**TEMPERATURE_ENVIRONMENTAL_CX_CXN**

Respetively:

Temperature = 25 +**TEMPERATURE_ENVIRONMENTAL_CX_CXN**

Example :
TEMPERATURE_ENVIRONMENTAL_CX_CXN = -009 → 25-7= 18°C

15. PV_AH_DAY_CXN / PV_CURRENT_CX

PV_CURRENT_CX: PV current at the moment
Dimension: in % of the nominal current
PV_AH_DAY_CXN: The used PV amperehours of the actual day
Dimension: amperehours (Ah)
→3Asciizeichen

CX:

FULLCUR_CX you see at **DATALOGGER** value (number 3).

If **FULLCUR_CX** $> 50_{10}$ then → CX10: x = 10
If **FULLCUR_CX** $> 23_{10}$ and **FULLCUR_CX** $< 30_{10}$ then → CX20: x = 20
If **FULLCUR_CX** $< 22_{10}$ then → CX40: x = 40

PV_CURRENT = (**PV_CURRENT_CX / FULLCUR_CX**) * x

CXN:

If CXN10: x=10
If CXN20: x=20
If CXN40: x=40

The CXN Type you see by the **DATALOGGE**R values (number 19)

PV_Ah = (**PV_AH_DAY_CXN** * x * 4) / 60

16. PV_CURRENT_CXN

PV current at the moment
Dimension: Ampere (A)
→ 4 Ascii characters

If CXN10: x=10
If CXN20: x=20
If CXN40: x=40

The CXN Type you see by the **DATALOGGE**R values (number 19)

PV_current = (**PV_CURRENT_CXN**/256) * x

Example:
CXN 10: **PV_CURRENT_CXN** = 26 → *PV_current* = 1A

17. LOAD_CURRENT_CXN

Load current at the moment
Dimension: Ampere (A)
→4 Ascii characters

If CXN10: x=10
If CXN20: x=20
If CXN40: x=40

The CXN Type you see by the **DATALOGGE**R values (number 19)

Load_current = (**LOAD_CURRENT_CXN**/256) * x

Example:
LOAD_CURRENT_CXN = 13 → *Load_current* = 0,5 A

18. REASON_LOAD_OFF_CXN

Convert this decimal number into a dual number and the bits has following explanation:
→ 4 Ascii characters

Bit0 = 1 → Load off because battery is empty
Bit1 = 1 → Load off manual by CXN button
Bit2 = 1 → Load off because load current was higher than nominal current
Bit3 = 1 → PV current reduce because of to high intern temperature
Bit4 = 1 → Load off because of nightlight function
Bit5 = 1 → Load off because of too high battery voltage
Bit6 = 1 → Load off because of too high intern temperature

19. EXCESS_AH_DAY_CXN

Shows you the excess energy (amperehours) of the actual day.
Dimension: amperehours (Ah)

→3Asciizeichen

If CXN10: x=10
If CXN20: x=20
If CXN40: x=40

The CXN Type you see by the **DATALOGGE**R values (number 19)

excess_Ah_day = (**EXCESS_AH_DAY_CXN** * x * 4) / 60

DATENLOGGER:

If you send out asciicharacter 33 (“!”) then you get the DATALOGGER data of CX respectively CXN controller. Please notice that the CX / CXN controller send back as echo the “!” to confirm that communication worked.

The controller send the DATALOGGER values always with two asciicharacters seperated with 32 (space). The asciicharacters represent hexnumbers (1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). The first character has always the highest rating.

The datalogger can separated in

- Setting values
- SOS (State of System) values
- Data about last week, last month and last year
- Serialnumber (only CXN)

Datalogger Setting values

Valuenumber	Value name	Number
1,2	values have no meaning	1 …5
		6
3	FULLCUR_CX	7 8
		9
4…10	values have no meaning	10…29
		30
11	MENUSTATE2_CX_CXN	31 32
		33
12	EVENINGHOURS_CX_CXN	34 35
		36
13	MORNINGHOURS_CX_CXN	37 38
		39
14	MENUSTATE_CX_CXN	40 41
		42
15	NIGHTLEVEL_CX_CXN	43 44
		45
16	RESERVED_CX_CXN	46 47
		48
17	RESERVED_CX_CXN	49 50
		51
18	RESERVED_CX_CXN	52 53
		54
19	TYPE_CXN RESERVED_CX	55 56
		57
20	VERSIONNUMBER_CX_CXN	58 59
		60

3. Fullcur_CX

If **FULLCUR_CX** $> 50_{10}$ $(= 32_{16})$ → it's a CX10 controller
If 23_{10} $(=17_{16})$ **< FULLCUR_CX** $< 30_{10}$ $(=1E_{16})$ → it's a CX20 controller
If **FULLCUR_CX** $< 22_{10}$ $(=16_{16})$ → it's a CX40 controller

11. MENUSTATE2_CX_CXN

Convert it to a binary number and look at the Bits
Bit 0 and 1 shows you the setting of the interface

Bit1	Bit0	
0	0	Serial interface EXCESS ENERGY & CURRENT DATA
0	1	Serial interface EXCESS ENERGY & DATALOGGER
1	0	Serial interface BIDIRECTIONAL NO EXCESS ENERGY

Bit 2 shows you if the menu button of the CX is locked or not:

Bit2 = 0 → button not locked
Bit2 = 1 →button locked

Example:
Menustate2 = 2_{16} =2_{10}= 0000 0010_2
Bit2 and Bit1 = 10 → Serial interface Bidirectional no excess energy
Bit3 = 0 → button not locked

12. EVENINGHOURS_CX_CXN

If the nightlight function is activated (look at **MENUSTATE_CX_CXN DATALOGGER** value 14) then you can see here the setting.

0 → Nightlight function Evening off
1 → Nightlight function Evening 1HR
2 → Nightlight function Evening 2HR
3 → Nightlight function Evening 3HR
4 → Nightlight function Evening 4HR
5 → Nightlight function Evening 5HR
6 → Nightlight function Evening to 4 Hours before mid of night
7 → Nightlight function Evening to 3 Hours before mid of night
8 → Nightlight function Evening to 2 Hours before mid of night
9 → Nightlight function Evening to 1 Hours before mid of night
10 → Nightlight function Evening to mid of night

Example:
EVENINGHOURS_CX_CXN = 2 → Nightlight function Evening 2HR

13. MORNINGHOURS_CX_CXN

If the nightlight function is activated (look at **MENUSTATE_CX_CXN DATALOGE**R VALUE 14) then you can see here the setting.

0 → Nightlight function Morning off
1 → Nightlight function Morning 1HR
2 → Nightlight function Morning 2HR
3 → Nightlight function Morning 3HR
4 → Nightlight function Morning 4HR
5 → Nightlight function Morning 5HR
6 → Nightlight function Morning to 4 Hours before mid of night
7 → Nightlight function Morning to 3 Hours before mid of night
8 → Nightlight function Morning to 2 Hours before mid of night
9 → Nightlight function Morning to 1 Hours before mid of night
10 → Nightlight function Morning to mid of night

Example:
MORNINGHOURS_CX_CXN = 8
→ Nightlight function Morning to 2 Hours before mid of night

14. MENUSTATE_CX_CXN

Convert the value in a binary number and the single Bits show how the chargecontroller is set:

Bit2	Bit1	Bit0	
0	0	0	Low voltage disconnect current compensated 11.4-11.9V
0	0	1	Low voltage disconnect current compensated 11.0-11.75V
0	1	0	Low voltage disconnect current compensated/ daptive 11.0V-12.2V
0	1	1	Low voltage disconnect 11.5V
1	0	0	Low voltage disconnect 11.0V

Bit3 = 0 → Battery type liquid electrolyte
Bit3 = 1 → Battery type GEL (VRLA)

Bit4 = 0 → Buzzer off
Bit4 = 1 → Buzzer on

Bit7	Bit6	Bit5	
0	0	0	Nightlight function off
1	1	0	Nightlight function DUSK TO DAWN
1	0	1	Nightlight function EVENING/MORNING

By „Nightlight function EVENING/MORNING" the values EVENINGHOURS_CX_CXN (DATALOGGER value 12) and MORINGHOURS_CX_CXN (DATALOGGER value13) shows you the setting.

Example:
MENUSTATE_CX_CXN= $0C_{16} = 12_{10}$ = 0000 1010

Bit2 and Bit1 and Bit0 = 010 → Low voltage disconnect current
Compensated / adaptive 11.0V-12.2V
Bit3: = 1 → Battery type GEL (VRLA)
Bit4 = 0 → Buzzer off
Bit7 and Bit6 and Bit5 = 000 → NIGHTLIGHT_FUNCTION_OFF

15. NIGHTLEVEL_CX_CXN

This value shows you the PV panel voltage level when the CXN respectively CX controller should know it's night. If PV panel voltage decrease under this voltage the controller know it's night and start for example with nightlight functions.

First value is in a 12V system and the second value for a 24V System

NIGHTLEVEL_CX_CXN = $1E_{16}$ = Day/Night PV voltage: 1.0/2.0V
NIGHTLEVEL_CX_CXN = $2E_{16}$ = Day/Night PV voltage: 1.6/3.1V
NIGHTLEVEL_CX_CXN = $3E_{16}$ = Day/Night PV voltage: 2.1/4.2V
NIGHTLEVEL_CX_CXN = $4E_{16}$ = Day/Night PV voltage: 2.7/5.4V
NIGHTLEVEL_CX_CXN = $5E_{16}$ = Day/Night PV voltage: 3.2/6.5V
NIGHTLEVEL_CX_CXN = $6E_{16}$ = Day/Night PV voltage: 3.8/7.6V
NIGHTLEVEL_CX_CXN = $7E_{16}$ = Day/Night PV voltage: 4.4/8.8V
NIGHTLEVEL_CX_CXN = $8E_{16}$ = Day/Night PV voltage: 4.9/9.8V
NIGHTLEVEL_CX_CXN = $9E_{16}$ = Day/Night PV voltage: 5.5/11.0V
NIGHTLEVEL_CX_CXN = AE_{16}= Day/Night PV voltage: 6.0/12.1V
NIGHTLEVEL_CX_CXN = BE_{16}= Day/Night PV voltage: 6.6/13.2V
NIGHTLEVEL_CX_CXN = CE_{16}= Day/Night PV voltage: 7.2/14.3V
NIGHTLEVEL_CX_CXN = DE_{16}= Day/Night PV voltage: 7.7/15.4V

Example:
NIGHTLEVEL_CX_CXN = $8E_{16}$ = 142_{10} → Day/Night threshold4.9/9.8V solar voltage

19. TYPE_CXN

Shows you the CXN Controller type:

Type_CXN = $0A_{16} = 10_{10}$ → CXN10
Type_CXN = $14_{16} = 20_{10}$ → CXN20
Type_CXN = $28_{16} = 40_{10}$ → CXN40

20. VERSIONNUMBER_CX_CXN

Convert it to a decimal number and you have the versionnumber of the microcontroller software.

Datalogger State of System values

SOS (State of System) Values

This values are saved since the beginning of the datalogger recording.
If the DATALOGGER will delete this values will set all to zero.

All data about amperehours and voltage will calculated as you see in the following lines:

Calculation of the amperehours:

CXN10 respectively CX10 → x=10
CXN20 respectively CX20 → x=20
CXN40 respectively CX40 → x=40

Insert x in the following formula:

amperehours_value_dec	=	The amperehours value in the **DATALOGGER** converted into a decimal number
result_amperehours	=	result (dimension: amperehours)

result_amperehours = (amperehours_value_dec * x * 4) / 60

Calculation of voltage values:

Voltage_value_dec	=	The voltage value in the datalogger converted into a decimal number
result_voltage	=	result (dimenstion: volt)

By 12V system: result_voltage = (Voltage_value_dec * 0,032V) + 9V
By 24V system: result_voltage = [(Voltage_value_dec * 0,032V) + 9V] * 2

Rating if a value is composed on more parts:

Part1 have always the highest rating:

Valuenumber	Value name	Bytenumber	Ascii character	Ascii value	Decimal
21	DEEP_DISCHARGE_EVENTS_CX_CXN(part1)	61	48	0	
		62	48	0	
		63		Space	17
22	DEEP_DISCHARGE_EVENTS_CX_CXN(part2)	64	49	1	
		65	49	1	
		66		Space	
23	WEEKS_WITHOUT_FULL_BATTERY_CX_CXN	67	48	0	
		68	57	9	9
		69		Space	
24	MONTHS_WITHOUT_FULL_BATTERY_CX_CXN	70	48	0	
		71	55	7	7
		72		Space	
25	SUMMARY_SOC_MORNING_CX_CXN(part1)	73	52	0	
		74	54	A	
		75		Space	2740
26	SUMMARY_SOC_MORNING_CX_CXN(part2)	76	65	B	
		77	52	4	
		78		Space	
27	PV_AH_CX_CXN(part1)	79	48	0	
		80	49	0	
		81		Space	
28	PV_AH_CX_CXN(part2)	82	50	1	
		83	48	0	4113
		84		Space	
29	PV_AH_CX_CXN (part3)	85	49	1	
		86	49	1	
		87		Space	
30	LOAD_AH_CX_CXN(part1)	88	48	0	
		89	49	0	
		90		Space	
31	LOAD_AH_CX_CXN (part2)	91	50	1	
		92	49	0	4113
		93		Space	
32	LOAD_AH_CX_CXN (part3)	94	48	1	
		95	48	1	
		96		Space	
33	DATALOGGER_DAYS_CX_CXN (part1)	97	48	0	
		98	49	1	
		99		Space	274
34	DATALOGGER_DAYS_CX_CXN(part2)	100	49	1	
		101	50	2	
		102		Space	

table 3: datalogger values

21. and 22. DEEP_DISCHARGE_EVENTS_CX_CXN

This values shows the number of load disconnects in the time periode of the whole **DATALOGGER** record.
Convert into a decimal number.

Example:
Deep_charge_event part1 and part2 = $0011_{16} = 17_{10}$

23. WEEKS_WITHOUT_FULL_BATTERY_CX_CXN

Sums up the number of weeks were the battery was not fully charged for the whole week.
Convert into a decimal number.

Example:
Number of weeks without fully charged battery = $09_{16} = 9_{10}$

24. MONTHS_WITHOUT_FULL_BATTERY_CX_CXN

Sums up the number of months were the battery was not fully charged for the whole month.
Convert into a decimal number.

Example:
Number of months without fully charged battery = $07_{16} = 7_{10}$

25. and 26. SUMMARY_SOC_morning

Convert into a decimal number and divide it through the datalogger record days (DATALOGGER number 33/34) then multiple this with 6,6 and then you have the average of the battery state of charge in the mornings in %.

$$summary\_soc\_morning\_final\_value = \frac{summary\_soc\_morning}{\text{datalogger days (32/33)}} * 6{,}6$$

Example:
Summary_soc_morning part1 and part2 = $0AB4_{16} = 2740_{10}$
Datalogger_days = $0112_{16} = 274_{10}$

$$\rightarrow summary\_soc\_morning\_final\_value = \frac{2740}{274} * 6{,}6 = 66\%$$

27, 28 and 29. PV_AH_CX_CXN

Calculate it as in you see on page 25.
This value shows you how many amperehours has been used from the PV panel.

Example:
PV current Ah part1 and part2 and part3 = $001011_{16} = 4113_{10}$

Type = CXN10: x = 10
PV_amperehours_final = (4113 * 10 * 4) / 60 = 2742

→ PV amperehours at the whole time periode
of the DATALOGGER = 2742Ah

30, 31 and 32. LOAD_AH_CX_CXN

Calculate it as in you see on page 25.
This value shows you how many amperehours has the load used.

Example:
Load_Ah part1, part2 and part3 = $001011_{16} = 4113_{10}$

Type = CXN10: x = 10
Load_amperehours_final = (4113 * 10 * 4) / 60 = 2742

→ Load amperehours at the whole time periode
of the datalogger= 2742Ah

33 and 34. DATALOGGER_DAYS_CX_CXN

Shows you the whole time periode of the datalogger
Convert it to a decimal number.

Example:
Datalogger days = $112_{16} = 274_{10}$

Datalogger data last week, last months, last year

Valuenumber	Value name	Bytenumber	character	Ascii value	Decimal
35	DAY1_BATTERY_VOLTAGE_MAX	103 104			
		105		Space	
36	DAY1_BATTERY_VOLTAGE_MIN	106 107			
		108		Space	
37	DAY1_PV_AH	109 110			
		111		Space	
38	DAY1_LOAD_AH	112 113			
		114		Space	
39	DAY1_MAX_LOAD_CURRENT	115 116			
		117		Space	
40	DAY1_EXCESS_AH	118 119			
		120		Space	
41	DAY1_MAX_PV_CURRENT	121 122			
		123		Space	
42	DAY1_SOC	124 125			
		126		Space	
43	DAY1_STATE	127 128			
		129		Space	
44…97	day2 ….day7	130 …290			
		291		Space	
98…133	week1…week4	292 ...398			
		399		Space	
134…241	month1…month12	400 …722			
		723		Space	
241…250	reserved	724 …758			
		759		Space	

table 4: Data day1

The datalogger save seven data sets from the last seven days, four data sets from the last four weeks and twelve data sets from the last twelve months. A data set includes following data:

- **BATTERY_VOLTAGE_MAX**
  maximal battery voltage from the battery of this day respectively the week or month. Calculate it as you see on page 25.

- **BATTERY_VOLTAGE_MIN**
  minimal battery voltage from the battery of this day respectively the week or month. Calculate it as you see on page 25.

- **PV_AH**
  By the day data: PV Amperehours of this day
  By the week and month data it's the average of all days in this week respectively month. Calculate it as you see on page 25

- **LOAD_AH**
  By the day data: Load Amperehours of this day
  By the week and month data it's the average of all days in this week respectively month. Calculate it as you see on page 25

- **EXCESS_AH**
  By the day data: Amperehours of the PV generator which can't used from the system (excess energy) of this day.

  By the week and month: Average excess amperehours of all days in this week respectively month of the PV generator which can't used from the system (excess energy)
  Calculate it as you see on page 25

- **MAX_LOAD_CURRENT**
  By day data: maximal Load current of this day
  By week and month data: average of all maximal load currents in this week respectively month

  If CX10 / CXN10: x = 10
  If CX20 / CXN20: x = 20
  If CX40 / CXN40: x = 40

  Max Load current = (**MAX_LOAD_CURRENT** / 64) * x

- **MAX_PV_Current**
  By day data: maximal PV current of this day
  By week and month data: maximal PV current in this week respectively month

  If CX10 / CXN10: x = 10
  If CX20 / CXN20: x = 20
  If CX40 / CXN40: x = 40

  Max PV current = (**MAX_PV_CURRENT** / 64) * x

- **SOC**
  Shows you the state of battery charge at morning and evening

  By day data: state of battery charge morning and evening

  By week and month data: Average of battery state of charge at the mornings and evenings of each day of this week respectively month.

  Convert the high and low hex number into a decimal number
  The first hex number is a value for the SOC in the EVENING and the other one for the SOC in the MORNINGS:

  Multiple this number with 6,6% and you have the state of battery charge at evening respectively in mornings .

  Example:
  DATALOGGER SOC Value: C7
  C_{16}: 12_{10} → SOC EVENING : 12 * 6,6% = 79%
  7_{16}: 7_{10} → SOC MORNING : 7 * 6,6% = 46%

- **STATE**
  Convert this Value into a binary number. Each bit shows following information.

  Bit0 = 1 → Load disconnected
  The battery has been disconnected from the load output because the state of charge was too low.

  Bit1 = 1 → Fully charged battery
  The battery has been fully charged.

  Bit2 = 1 → PV over current
  The PV current was over the nominal current

  Bit3 = 1 → Load over current
  The Load current was over the nominal current

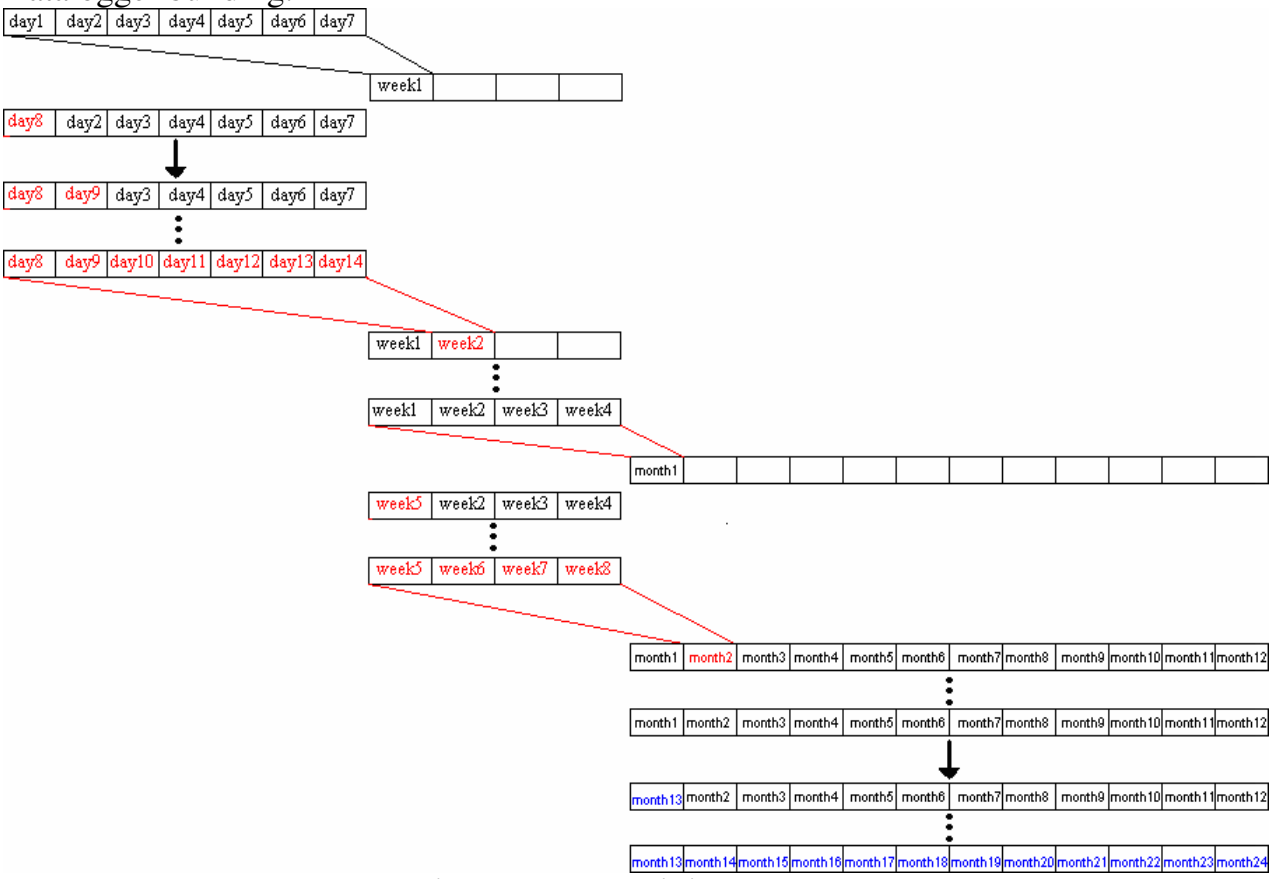
  Bit4 = 1 → Over battery voltage
  By 12V system the battery voltage was over 15,5V
  By 24V system the battery voltage was over 31V

  Bit5 = 1 → PV current reduce because of too high temperature

  Bit6 = 1 → Load disconnect because of too high temperature

The datasets will saved in the following order as you see in the picture.

Datalogger building:



Picture: datalogger building

Serialnumber

Only CXN:

Valuenumber	Value name	Bytenumber
251	SERIALNUMBER_CXN (part1)	750
		751
		752
252	SERIALNUMBER_CXN (part2)	753
		754
		756
253	SERIALNUMBER_CXN (part3)	757
		758
		759
254	SERIALNUMBER_CXN (part4)	760
		761
		762
255	SERIALNUMBER_CXN (part5)	763
		764
		765
256	SERIALNUMBER_CXN (part6)	766
		767

table 5: serial number

Commands to set the CX/CXN controller

Ascii characters which you need to send Commands to the CX/CXN:

Ascii value	Byte	Asciinumber
0	00110000	48
1	00110001	49
2	00110010	50
3	00110011	51
4	00110100	52
5	00110101	53
6	00110110	54
7	00110111	55
8	00111000	56
9	00111001	57
A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
E	01000101	69
F	01000110	70
+	00101011	43
-	00101101	45

%	00100101	37
C	01100011	99
X	01111000	120
P	01010000	80
H	01001000	72
O	01001111	79
G	01000111	71
I	01001001	73
J	01001010	74
K	01001011	75
L	01001100	76
M	01001101	77
N	01001110	78
Q	01010001	81
A	01000001	65

table 6: Ascii characters for commands

In table8 you have a list of all command which you can send to the controller

The CX/CXN sends back each character after reception. Don't send the next character before you get the Echo.

command	***description***
Space (Ascci character 32)	Space key :cx sends the normal uart values
! (Ascii character 31)	„!“ : cx sends the datalogger values
%cxPHOGxx	The hex value xx is stored in menustate(EEPROM) example: %cxPHOG10 : => menustate = 0x10 => Buzzer on
%cxPHOHxx	The hex value xx is stored in menustate2(EEPROM)
%cxPHOIxx	The hex value xx is stored in eveninghours(EEPROM)
%cxPHOJxx	The hex value xx is stored in morninghours(EEPROM)
%cxPHOLAA	clear datalogger in the CXN
%cxPHOMxx	set nightlevel

table 7: List of all commands to the CXN

The following pages shows you some examples how you can set the CXN:

Set "MENUSTATE" (DATALOGGER Value 14)

You can set the
- Battery protections setting
- Battery typ
- Buzzer
- Nightlight function

To set the controller look at value15 by the datalogger values there you can see what possibilities you have:

Example:

Low voltage disconnect 11,5V → Bit3 = 1, Bit2 = 0, Bit1 = 0.
Battery typ Gel → Bit4 = 1
Buzzer on → Bit5 = 1
Nightlight function Evening/morning: Bit8 = 1, Bit7=0, Bit6 = 1

Convert this binary number into a hex number:
$10111100_2=BC_{16}$

Send now the Ascii character for setting the menuestate and and the end send BC as you see following:

%cxPHOGBC

Read out the DATALOGGER again and now the value menustate have the value BC.

Set "MENUSTATE2" (DATALOGGER Value 11)

To set values in menustate2 send following Ascii characters.

%cxPHOH--

For - - send a hex number with tow digits with the Bits which you want to set.

Set "NIGHTLEVEL" (DATALOGGER Value 15)

Nightlevel = $1E_{16}$ = Day/Night thres hold1.0/2.0V Solar voltage
Nightlevel = $2E_{16}$ = Day/Night thres hold1.6/3.1V Solar voltage
Nightlevel = $3E_{16}$ = Day/Night thres hold2.1/4.2V Solar voltage
Nightlevel = $4E_{16}$ = Day/Night thres hold2.7/5.4V Solar voltage
Nightlevel = $5E_{16}$ = Day/Night thres hold3.2/6.5V Solar voltage
Nightlevel = $6E_{16}$ = Day/Night thres hold3.8/7.6V Solar voltage
Nightlevel = $7E_{16}$ = Day/Night thres hold4.4/8.8V Solar voltage
Nightlevel = $8E_{16}$ = Day/Night thres hold4.9/9.8V Solar voltage
Nightlevel = $9E_{16}$ = Day/Night thres hold5.5/11.0V Solar voltage
Nightlevel = AE_{16} = Day/Night thres hold6.0/12.1V Solar voltage
Nightlevel = BE_{16} = Day/Night thres hold6.6/13.2V Solar voltage
Nightlevel = CE_{16} = Day/Night thres hold7.2/14.3V Solar voltage
Nightlevel = DE_{16} = Day/Night thres hold7.7/15.4V Solar voltage

If you want to change the nightlevel setting send following characters before:

%cxPHOM

For Example if you want to set the for "Day/Night threshold4.9/9.8V Solar Voltage"
Then send:
%cxPHOM8E

Set "EVENINGHOURS" (DATALOGGER Value 12)

00 → Nightlight function Evening off
01 → Nightlight function Evening 1HR
02 → Nightlight function Evening 2HR
03 → Nightlight function Evening 3HR
04 → Nightlight function Evening 4HR
05 → Nightlight function Evening 5HR
06 → Nightlight function Evening to 4 Hours before mid of night
07 → Nightlight function Evening to 3 Hours before mid of night
08 → Nightlight function Evening to 2 Hours before mid of night
09 → Nightlight function Evening to 1 Hours before mid of night
0A → Nightlight function Evening to mid of night

Example:
If you want to set eveninghours "Nightlight function Evening to mid of night"

Send: %cxPHOI0A

Set "MORNINGHOURS" (DATALOGGER Values 13)

00 → Nightlight function Morning off
01 → Nightlight function Morning 1HR
02 → Nightlight function Morning 2HR
03 → Nightlight function Morning 3HR
04 → Nightlight function Morning 4HR
05 → Nightlight function Morning 5HR
06 → Nightlight function Morning to 4 Hours before mid of night
07 → Nightlight function Morning to 3 Hours before mid of night
08 → Nightlight function Morning to 2 Hours before mid of night
09 → Nightlight function Morning to 1 Hours before mid of night
0A → Nightlight function Morning to mid of night

Example:
If you want to set eveninghours "Nightlight function Morning to 4 Hours before mid of night"

Send: %cxPHOJ06