1. Topologie, Grundsätzliches

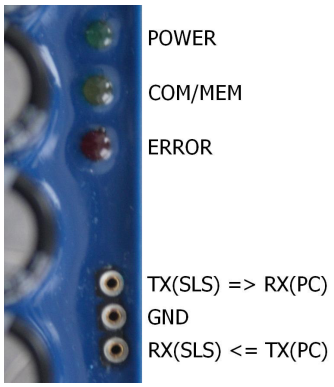
Sowohl SLSi als auch cSLSi unterstützt eine serielle (RS232) Schnittstelle u.a. zum Auslesen von Status-Information. Das Software-Protokoll ist für alle cSLSi/SLSi identisch.
Grundsätzlich ist nach dem Senden eines Befehls das Antwort-Telegramm abzuwarten. Erst nach Erhalt der Antwort darf ein neuer Befehl gesendet werden. Die Kommunikation kann nur vom Host initiiert werden. Die Einstellungen der seriellen Schnittstelle sind 115kBd, 1Startbit, 8 Datenbits, 1 Stoppbit, keine Parität.

1.1 SLS

Die serielle Verbindung ist beim SLS über eine 3,5mm Stereo-Klinkenbuchse realisiert. Die Pinbelegung des 3,5mm Klinkensteckers des Verbindungskabels ist nachfolgend gezeigt:

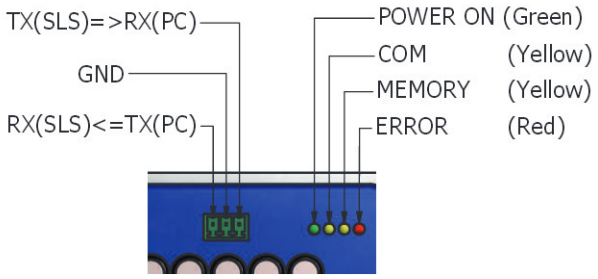
1.2 cSLS

Die serielle Verbindung ist beim cSLS über ein Buchsenleiste im 2,54mm Raster mit runden 0,5mm Pins realisiert. Der markierte PIN zeigt zu den LEDs.



1.3 SLSi

Die Pinbelegung des Steckers am SLSi finden Sie nachfolgend:



2. Protokoll

2.1 Struktur des Protokolls

Byte 0	Byte 1	Byte 2	Byte 3 .. (A-1)	Byte A
S	A	T	D	CRC

Byte 0: Das erste Byte (S) in einem Kommunikations-Rahmen ist als **Sync-Byte** implementiert. Dabei kommt das Zeichen '!' für eine initiale Kommunikation vom Host zum SLS zum Einsatz. Das rücklaufende Paket enthält dagegen das Sync-Zeichen '?'.

Byte 1: Dieses Byte (A) dient als **Zähler**. Es werden die Bytes von (Byte 0 .. Byte (A-1)) gezählt. (Der tatsächliche Wert des Zählers kann sich mit der Firmware-Version ändern!)

Byte 2: Diese Byte (T) stellt den eigentlichen **Befehl/Tag** dar, der vom SLS abgearbeitet werden soll. Einzelheiten siehe weiter unten. Rücklaufende Pakete schicken dieses Byte unverändert zurück.

Byte 3..(A-1): Optionale **Daten** bzw. Parameter für den auszuführenden Befehl bzw. Ergebnisdaten im Antwort-Telegramm.

Byte A: Nach den Datenbytes wird noch eine **Checksumme** (CRC) angehängt, die aus der Summe aller Bytes (excl. der Checksumme selbst) besteht.

Hin- und rücklaufende Pakete haben die gleiche, oben beschriebene Struktur.

2.2 Status-Abfrage:

Status anfordern:

Host sendet an SLS:

Byte 0	Byte 1	Byte 2	Byte 3
'!'	3_d	'S'	CRC

ACK: Status Rückmeldung:

SLS sendet an Host:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
'?'	63_d	'S'	T_P	UZK_L	UZK_H	Iac_L	Iac_H

Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
RPM_L	RPM_H	T_F	U_F	C_F	--	--	--

Byte 16	Byte 17	Byte 18	Byte 19	Byte 20	Byte 21	Byte 22	Byte 23
Tmax_DR	Umin_DR	Umax_DR	AMPS_H	AMPS_L	MaxRPM_H	MaxRPM_L	--

Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31
--	--	--	--	--	--	--	--

Byte 32	Byte 33	Byte 34	Byte 35	Byte 36	Byte 37	Byte 38	Byte 39
--	--	--	--	--	--	--	--

Byte 40	Byte 41	Byte 42	Byte 43	Byte 44	Byte 45	Byte 46	Byte 47
--	--	Id_L	Id_H	--	--	--	--

Byte 48	Byte 49	Byte 50	Byte 51	Byte 52	Byte 53	Byte 54	Byte 55
--	--	--	--	--	--	--	--

Byte 56	Byte 57	Byte 58	Byte 59	Byte 60	Byte 61	Byte 62	Byte 63
--	--	--	--	T_E	--	--	CRC

T_P (Powermodultemperatur): Umrechnung auf °C: T[°C] = (Temp / 2) - 10
(Messbereich: -10°C .. +117,5°C)

T_E (Elkotemperatur (nur bei SLSi)): Umrechnung auf °C: T[°C] = (Temp / 2) - 10
(Messbereich: -10°C .. +117,5°C)

UZK_L (Reglerspannung low-Byte)
UZK_H (Reglerspannung high-Byte): Umrechnung: U[V] = (UZK * MaxUZK) / 1023

mit MaxUZK:
24V Regler: 27,78V
42V Regler: 46,67V
60V Regler: 66,11V
(Skalierungsfaktor für die Spannung in Volt)

AMPS_L
AMPS_H : maximal freigebbarer Strom in 0,1 A Schritten

Iac_L (Motorstrom low-Byte)
Iac_H (Motorstrom high-Byte): Umrechnung: I[A_{eff}] = Iac * AMPS / 10 / 4095
Bit 15 ist das Vorzeichenbit

MaxRPM_L
MaxRPM_H : maximal freigebbare Motordrehzahl in U/min

RPM_L (Drehzahl low-Byte)
RPM_H (Drehzahl high-Byte): Umrechnung: N[U/min] = RPM * MaxRPM / 10922
Bit 15 ist das Vorzeichenbit (Drehrichtung)

T_F (Übertemperaturfehler):
Bit 7-0

R-0	R-0	R-0	U-0	U-0	U-0	U-0	U-0
SO_T	CMT	LMT	--	--	--	--	--

Der Wert in geschweiften Klammern ist dem Windows-Monitor unter Parameter->SLS zu entnehmen!

Bit 7 **SO_T:** SwitchOff_OverTemp (T_P > 100°C, T_E > 112°C) =>Failsafe (SLS stopped)
Bit 6 **CMT:** CutOff_MaxTemp (T_P > 90°C, T_E > 106°C)
Bit 5 **LMT:** Limit_MaxTemp(T_P > {Temp_Lim}, T_E > 90°C)
Bit 4-0 **unimplemented:** read as '0'

U_F (Über- /Unterspannungsfehler):
Bit 7-0

R-0	R-0	R-0	U-0	R-0	R-0	R-0	U-0
SO_OV	CMV	LMV	--	SO_UV	CUV	LUV	--

Die Werte in geschweiften Klammern sind dem Windows-Monitor unter Parameter->Batterie zu entnehmen!

Bit 7 **SO_OV:** SwitchOff_OverVoltage (24V Regler: U > 25,66V
42V Regler: U > 43,11V
60V Regler: U > 61,07V)
Bit 6 **CMV:** CutOff_MaxVoltage (U > {U_Batt_max})
Bit 5 **LMV:** Limit_MaxVoltage (U > {U_Batt_max} - 1V)
Bit 4 **unimplemented:** read as '0'
Bit 3 **SO_UV:** SwitchOff_UnderVoltage (24V Regler: U < 7,98V
42V Regler: U < 15,51V
60V Regler: U < 15,51V
bis SN599 bei den SLS: U < 22,49V)
Bit 2 **CUV:** CutOff_UnderVoltage (U <{U_Batt_Low})
Bit 1 **LUV:** Limit_UnderVoltage (U < {U_Batt_Lim})
Bit 0 **unimplemented:** read as '0'

C_F (Fehler der internen Kontrolle):
Bit 7-0

R-0	U-0	R-0	U-0	R-0	U-0	U-0	R-0
PL_F	--	ZS_F	I_F	OS_F	LL_F	2PH_F	FS

Bit 7 **PL_F:** PhaseLoss_Fault -> stop+retry
Bit 6 **unimplemented:** read as '0'
Bit 5 **ZS_F:** ZeroSpeed_Fault -> stop+retry
Bit 4 **I_F:** IOffset_Fault -> stop+retry
Bit 3 **OS_F:** OverSpeed_Fault -> stop+retry
Bit 2 **LL_F:** Loadloss_Fault -> stop+retry
Bit 1 **2PH_F:** 2-Phasen-PWM
Bit 0 **FS:** Failsafe (SLS stopped) -> check -> clear Error

Tmax_DR (Temp Max derate Register)**:**
Umin_DR (Überspannungs derate Register)**:**
Umax_DR (Unterspannungs derate Register)**:** Diese 3 Register geben genauer an, wie weit abgeregelt wird. Wertebereiche für die 3 Derate-Register:
0x40 (Es wird noch nicht abgeregelt) bis
0x00 (Regler schaltet ab)

Id_L (Id low-Byte)
Id_H (Id high-Byte): Umrechnung: $I[A_{eff}] = Iac * AMPS / 10 / 4095$
Bit 15 ist das Vorzeichenbit

NACK: Übertragungsfehler:

SLS sendet an Host:

Byte 0	Byte 1	Byte 2	Byte 3
'?'	3_d	'?'	CRC

2.2.1 Servosignal überschreiben:

Status anfordern:

Host sendet an SLS:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
'!'	7_d	'S'	1_d	Aktiv	Signal_L	Signal_H	CRC

Um einen Timeout zu verhindern muss das TAG "Servosignal überschreiben" zyklisch gesendet werden. Der Timeout beträgt 300ms.

Aktiv : 0xAA Signal wird überschrieben.
0x00 Signal wird nicht überschrieben.

Signal_L
Signal_H : Signalvorgabe in µs zulässiger Bereich:
800 .. 2200 µs

ACK: Status Rückmeldung:

SLS sendet an Host siehe 2.2

2.2.2 Servosignal offset:

Status anfordern:

Host sendet an SLS:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
'!'	5_d	'S'	2_d	Servooffset	CRC

Servooffset : Servosignaloffset in µsec
Wertebereich: + / - 127 µsec

ACK: Status Rückmeldung:

SLS sendet an Host siehe 2.2

2.2.3 Kontrolle über Bedienpult (kostenpflichtige Option):

Status anfordern:

Host sendet an SLS:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
'!'	14_d	'S'	3_d	Aktiv	Control	SPD_WMon_L	SPD_WMon_H

Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14
MtrCur_WMon_L	MtrCur_WMon_H	RegCur_WMon_L	RegCur_WMon_H	Accel_WMon	Decel_WMon	CRC

Um einen Timeout zu verhindern muss das TAG "Kontrolle über Bedienpult" zyklisch gesendet werden. Der Timeout beträgt 300ms.

Aktiv : 0xAA Signal wird überschrieben.
0x00 Signal wird nicht überschrieben.

Control:
Bit 7-0

U-0	U-0	U-0	U-0	U-0	W	W	W
--	--	--	--	--	Direction	Start/Stop	Feststellbremse_Aktiv

Bit 7-3 **unimplemented:** read as '0'
Bit 2 Direction
Bit 1 Stop = 0; Start = 1
Bit 0 Feststellbremse aktiv = 1

SPD_WM_L
SPD_WM_H : Drehzahlvorgabe
Wertebereich: 0 .. 1023
1023 entspricht der maximal freigebbaren Motordrehzahl (**MaxRPM)**

MtrCur_WMon_L
MtrCur_WMonl_H : Motorstromvorgabe
Wertebereich: 0 .. 1023
1023 entspricht dem maximal freigebbaren Strom (**AMPS)**

RegCur_WMon_L
RegCur_WMon_H : Rückspeisestromvorgabe
Wertebereich: 0 .. 1023
1023 entspricht dem maximal freigebbaren Strom (**AMPS)**

Accel_WMon: Beschleunigungsrate
Wertebereich: 1 .. 255

Decel_WMon: Bremsrate
Wertebereich: 1 .. 255

ACK: Status Rückmeldung:

SLS sendet an Host siehe 2.2

2.3 Fehlerrücksetzen/Reset

Fehler rücksetzen:

Host sendet an SLS:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
'!'	4_d	'R'	PAR	CRC

PAR:

Bit 7-0

W-0	U-0	U-0	W-0	U-0	U-0	U-0	U-0
Reboot	0	0	Clear	0	0	0	0

Bit 7 **Reboot:** komplette Software wird neu gestartet (via Reset-Vektor)
Bit 6-5 **unimplemented:** write as '0'
Bit 4 **Clear: a**lle Fehler werden zurück gesetzt
Bit 3-0 **unimplemented:** write as '0'

ACK -> Quittierung:

SLS sendet an Host:

Byte 0	Byte 1	Byte 2	Byte 3
'?'	3_d	'R'	CRC

NACK -> Übertragungsfehler:

SLS sendet an Host:

Byte 0	Byte 1	Byte 2	Byte 3
'?'	3_d	'?'	CRC