X-SCC-2

2 serielle Schnittstellen
(2 x RS-232 und 2 x RS-232, RS-422 oder RS-485)

10.28. X-SCC-2

10.28. X-SCC-2 10-331
10.28.1. Beschreibung 10-332
10.28.2. Blockschaltbild (X-SCC-2) 10-334
10.28.3. Software 10-335
10.28.4. Modul-Device-Treiber 10-335
10.28.4.1. Installation 10-335
10.28.4.2. Kanaleigenschaftsstruktur CPS_SCC 10-335
10.28.4.3. Funktionsweise 10-335
10.28.4.4. Konfiguration der seriellen Schnittstellen 10-336
10.28.4.5. DSR-Eingang 10-338
10.28.4.6. Direkter Zugriff auf den ESCC 10-339
10.28.4.7. Interrupt 10-340
10.28.5. Treiberprogramm CQmax 10-343
10.28.5.1. Grundstruktur 10-344
10.28.5.2. Handshake (asynchrone Kommunikation) 10-345

10.28.5.3. Installation 10-345
10.28.5.4. Parametrierung und Initialisierung 10-346
10.28.5.5. Empfangen 10-346
10.28.5.6. Senden 10-348
10.28.5.7. Die Parameter des Programmes X1PA005.LIB 10-349
10.28.5.8. Fehlercodes in Parameter 1 10-355
10.28.5.9. Funktionen und Prozeduren von CQmax 10-356
10.28.5.10. Bedeutung der Fehlerrückgabecodes 10-359
10.28.5.11. Die Aktions-Filter des Programms X1PA005 10-360
10.28.5.12. Aktions-Filter und zugehörige Filter-Argumente 10-361
10.28.5.13. Zusätzliche Parameter für die Aktionsfilter 10-363
10.28.6. Anschlusspins des Moduls (bezogen auf den Modul-Stecker A) 10-367
10.28.7. Besondere Eigenschaften 10-368

10.28.1. Beschreibung

Das Modul X-SCC-2 bietet 2 unabhängige, serielle Kommunikationskanäle. Es verwendet dazu den „Enhanced Serial Communication Controller“ ESCC Z85230 von ZILOG mit einem 14,7456 MHz Quarzoszillator. Der Controller verfügt über einen 8-Byte großen Empfangs-FIFO und einen 4 Byte grossen Sende-FIFO je Kanal.

Dieser Controller bietet eine Vielzahl von Betriebsarten, wie z.B. asynchrone Kommunikation, Byte-orientierte synchrone Kommunikation (Monosync- oder Bisync-Protokoll), Bit-orientierte synchrone Protokolle (HDLC oder SDLC).

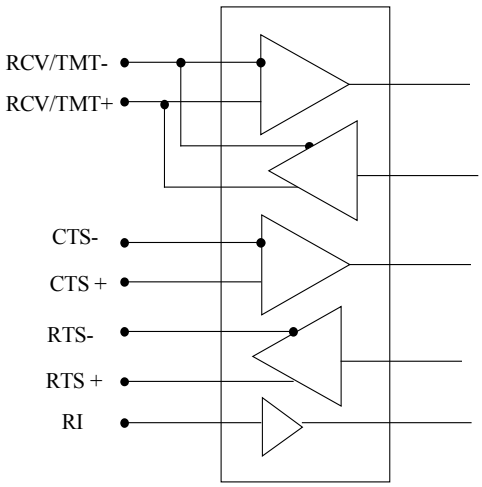
Bei der Verwendung als RS-232 Schnittstelle sind alle Modem-Steuersignale (RTS, CTS, DTR, DSR, Ri, DCD) vorhanden, zusätzlich ein Clock-Eingang an CLKio.

Ein Taktsignal kann von außen angelegt oder oder intern erzeugt werden. Pro Kanal steht ein Baudratengenerator und PLL zur Verfügung. CLKio, Ri und CTS sind z.B. als Clock-Eingänge und RTS Clock-Ausgang programmierbar.

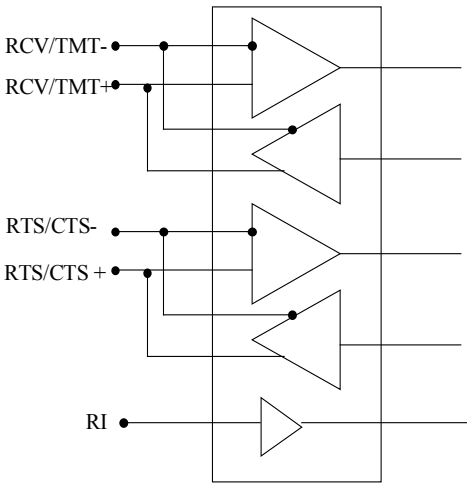
Das Modul ist in drei Versionen lieferbar:

X-SCC-2/U

Physikalische Schnittstelle per Software umschaltbar zwischen RS-232, RS-422, RS-485/1 (Voll-Duplex) und RS-485/2 (Halb-Duplex). Die Unterscheidung zwischen RS-485/1 (Voll-Duplex) und RS-485/2 (Halb-Duplex) bezieht sich nur auf die RTS und CTS Steuerleitungen. Bei RS-485 und RS-422 können die Abschlusswiderstände per Software zugeschaltet werden.



RS-485/1 (Voll-Duplex)



RS-485/2 (Halb-Duplex)

X-SCC-2/R

Nur RS-232 möglich (max. 120 KBaud).

X-SCC-2i/C

Wie X-SCC-2/R mit zusätzlicher 20mA Current Loop Sender/Empfänger (galv. getrennt) Schnittstelle und zwei 20mA Konstantstromquellen je Kanal (je eine für Senden und Empfangen). RS-232: max. 120 KBaud, 20mA Current Loop: max 20kBaud bei 400m Leitungslänge.

Wenn ein Teil der Schnittstelle (RCV oder TMT) mit Hilfe einer der beiden 20 mA Konstantstromquellen den Strom für die Verbindung liefert, wird er als aktiv bezeichnet. Wenn der Strom von der Gegenstation kommt, als passiv. Nur ein passiver Teil ist auf dem Modul galvanisch getrennt. Die Konfiguration (aktiv/passiv) wird über Verbindungen am externen Anschlußstecker eingestellt.

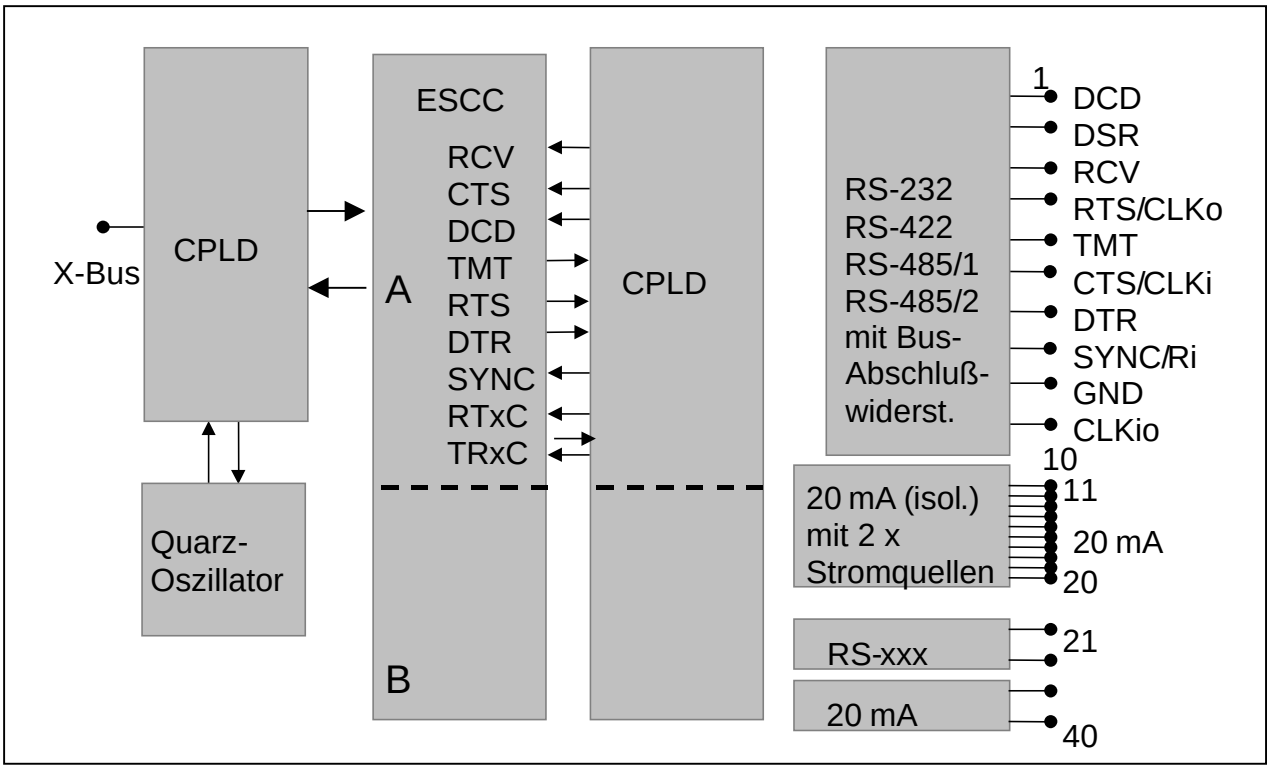
Beachten Sie bitte, dass bei 20 mA Verbindungen + mit - und - mit + der Gegenstation verbunden wird. CCS1 und CCS2 sind die Ausgänge der beiden 20 mA Konstantstromquellen.

Signal X-SCC-2	**Pin Kanal A**	**Pin Kanal B**	**RCV: TMT:**	**aktiv aktiv**	**aktiv passiv**	**passiv aktiv**	**passiv passiv**
-12 V	21	31		-	-	-	-
TMT+	22	32		TMT-	TMT+	TMT-	TMT+
TMT-	23	33		⌐¬*	TMT-	⌐¬*	TMT-
GND	24	34		⌐¬	-	⌐¬	-
CCS1+	25	35		TMT+	-	TMT+	-
RCV+	26	36		RCV+	RCV+	RCV+	RCV+
RCV-	27	37		⌐¬*	⌐¬*	RCV-	RCV-
GND	28	38		⌐¬	⌐¬	-	-
CCS2+	29	39		RCV-	RCV-	-	-
+12V	30	40		-	-	-	-

- := Pin ist nicht angeschlossen
* := am externen Anschlußstecker verbinden

Die Stromquellen auf dem Modul werden von +12 Volt gespeist. Um den Spannungshub zu vergrößern, kann die Rückführung statt an Ground auch an -12 Volt erfolgen.

10.28.2. Blockschaltbild (X-SCC-2)



10.28.3. Software

Für die Programmierung gibt es zwei Möglichkeiten:

- CQmax: Ein Treiber mit eigener Anwenderschnittstelle, der kompatibel zu der Vorgänger-Hardware MODULAR-4 ist.

- MDD: Die Treiberschnittstelle entspricht allen anderen Modulen. Derzeit ist im MDD noch nicht die volle Funktionalität implementiert. Zukünftige Versionen werden eine höhere Benutzerfreundlichkeit bieten (ähnlich der anderer Module wie zum Beispiel X-COM-4).

10.28.4. Modul-Device-Treiber

10.28.4.1. Installation

Der Modul-Device-Treiber für OsX hat die Programmnummer 8054h und den Dateinamen mxscc2.exe. Der Modul-Device-Treiber für Windows hat den Namen mxscc2.sys. Die Installation aus einem PC-Programm (z.B. für Steckplatz 1, Layer 0) erfolgt mit folgendem Bibliotheksaufruf:

Error = max_load_mdd(hModul, 1, 0, 0, 0x8054, NULL, &hMDD);

Der entsprechende Befehl in einer INS-Datei (z.B. für Steckplatz 1, Layer 0) lautet:

MAXLOADMDD slot=1 layer=0 progno=8054

10.28.4.2. Kanaleigenschaftsstruktur CPS_SCC

Die CPS für das Modul hat den Namen CPS_XSCC.

10.28.4.3. Funktionsweise

Das Modul kann nur von einem MDD zur Zeit verwendet werden.

Die derzeitige MDD-Version bietet nur die Möglichkeit den seriellen Schnittstellen-Controller direkt zu programmieren. Dazu können Kanäle auf die einzelnen Register des ESCC geöffnet werden. Darüber können die einzelnen Register gezielt geschrieben und gelesen werden. Alle Einstellungsmöglichkeiten des Controllers stehen so transparent zur Verfügung. Bei dieser Methode ist die genaue Kenntnis der Funkti-

onsweise des Controllers und seiner Register unerläßlich. Die Beschreibung ist im Internet unter www.zilog.com herunterladbar (Doc ID UM0109).

10.28.4.4. Konfiguration der seriellen Schnittstellen

Über die eigentliche Funktionalität des ESCC hinaus gehend bietet das Modul verschiedene Möglichkeiten zur Verschaltung von ESCC-Signalen an die Stecker-Pins. Außerdem muss die physikalische Schnittstelle festgelegt werden. Dazu ist ein Konfigurationskanal mit folgender CPS zu öffnen:

Strukturelement	**Werte**	**Bedeutung**
.usVersion	*0*	Version der CPS-Struktur-Definition
.usChannel	*0 oder 1*	Schnittstelle
.usDevice	*DEVICE_CTRL*	Steuerkanal
.usIndexFirst	*XSCC_CTRL_PHYS_IFC*	Konfiguration der pysikalischen Schnittstelle
.usIndexLast	*=.usIndexFirst*	
.usType		Auswahl der physikalischen Signal-Art
	COM_RS232	
	COM_RS422	(nur für X-SCC-2/U)
	COM_RS485_HALF_DUPLEX	(nur für X-SCC-2/U)
	COM_RS485_FULL_DUPLEX	(nur für X-SCC-2/U)
	COM_CURRENT_LOOP	(nur für X-SCC-2/C)
.usFlags	*_CP_EXCLUSIVE*	Dieses Flag muss gesetzt sein! Der Zugriff auf eine Schnittstelle erfolgt immer exklusiv.
	XSCC_DTR_INTERNAL	Nur für RS-232: Ist dieses Flag gesetzt, kann das am DTR-Pin ausgegebene Signal über den Ausgabedienst des Kanals geschaltet werden. DTR kommt dann nicht vom ESCC. Ist das Bit nicht gesetzt, wird das DTR-Signal des ESCC auf dem DTR-Pin ausgegeben.
	XSCC_RTS_FROM_TRXC	Am RTS-Pin wird das TRxC-Signal des ESCC ausgegeben. Im ESCC bestehen diverse Einstellungsmöglichkeiten, welches Signal am TRxC-Pin ausgegeben wird. In diesem Fall muss der TRxC-Pin des ESCC im ESCC-Register WR11 als Ausgang konfiguriert werden. Ist das Bit nicht gesetzt, wird das RTS-Signal des ESCC auf dem RTS-Pin ausgegeben.

Strukturelement	Werte	Bedeutung
.usReadMode	*IO_MODE_RAM*	
.usWriteMode	*IO_MODE_DIRECT*	
.usTRxC		Nur für RS-232: Festlegen der Verbindung TRxC-Pin des ESCC mit dem Stecker (der TRxC-Pin des ESCC muss dafür im ESCC-Write-Register WR11 als Eingang konfiguriert werden).
	XSCC_SEL_CLKIO	Pin CLKIO mit TRxC verbinden[1]
	XSCC_ SEL_RI	Pin RI mit TRxC verbinden
	XSCC_ SEL_CTS	Pin CTS mit TRxC verbinden[1]
	XSCC_ SEL_NC	Keine Verbindung TRxC nach außen
.usRTxC		Festlegen der Verbindung des RTxC-Pins des ESCC mit dem Stecker
	XSCC_ SEL_CLKIO	Pin CLKIO mit RTxC verbinden
	XSCC_ SEL_RI	Pin RI mit RTxC verbinden
	XSCC_ SEL_CTS	Pin CTS mit RTxC verbinden
	XSCC_ SEL_PCLK	An den RTxC-Pin wird der halbe ESCC-Takt (7,3728MHz angelegt)
.usMode	*0*	Reserviert

Anmerkungen

Der Kanal für die Konfiguration einer Schnittstelle muss geöffnet werden, bevor die anderen Kanäle zu der Schnittstelle geöffnet werden können.

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_UCHAR.

max_write_channel_uchar

Setzen/Zurücklesen des DTR-Pins auf 0 oder 1 (nur wenn *XSCC_DTR_INTERNAL* gesetzt ist).

Sonderdienste

max_channel_control, Steuerbefehl CMD_STOP schaltet RTS/CTS und RCV/TMT ab. Mit CMD_START können die Anschlüsse wieder aktiviert werden.

max_channel_control, Steuerbefehl CMD_XSCC_SLOW: Die Steilheit der Flanken wird zur EMV-Verbesserung begrenzt. Mit CMD_XSCC_FAST wird die Begren-

[1] Nicht in allen Modes einstellbar

zung abgeschaltet (Default-Einstellung). In dieser Betriebsart kann die maximale Übertragungsgeschwindigkeit erreicht werden.

max_channel_control, Steuerbefehl CMD_XSCC_RBUS_ON, CMD_XSCC_RBUS_OFF (nur für RS-4xx): Bus-Abschlußwiderstände ein- (Default-Einstellung) bzw. ausschalten.

max_channel_control, Steuerbefehl CMD_DIR_INPUT: Schaltet eine RS-485-Schnittstelle auf Empfang (Sender wird abgeschaltet). CMD_DIR_OUTPUT: Schaltet eine RS-485-Schnittstelle auf Senden (Sender wird eingeschaltet). Die Umschaltung bezieht sich nur auf das RCV/TMT Signal. Über diesen Sonderdienst kann in allen RS-4xx Modes der TMT-Sender aus-/bzw. eingeschaltet werden.

max_channel_control, Steuerbefehl CMD_DIR_INPUT_RTSCTS: Schaltet eine RS-485/2-Schnittstelle (Halb-Duplex) auf Empfang (Sender wird abgeschaltet). CMD_DIR_OUTPUT_RTSCTS: Schaltet eine RS-485/2-Schnittstelle (Halb-Duplex) auf Senden (Sender wird eingeschaltet). Die Umschaltung bezieht sich nur auf das RTS/CTS Signal. Über diesen Sonderdienst kann in allen RS-4xx Modes der RTS-Sender aus-/bzw. eingeschaltet werden.

10.28.4.5. DSR-Eingang

Im RS-232 Betrieb kann der Zustand des DSR-Eingangs über folgenden Kanal ausgelesen werden:

Strukturelement	Werte	Bedeutung
.usVersion	*0*	Version der CPS-Struktur-Definition
.usChannel	*0 oder 1*	Schnittstelle
.usDevice	*DEVICE_DIN*	Eingang
.usIndexFirst	*0*	DSR
..usIndexLast	*UsIndexFirst*	DSR
.usMode	*0*	Reserviert
.usFlags	*0*	Reserviert
.usReadMode	*IO_MODE_DIRECT*	Eingang wird direkt gelesen
.usWriteMode	*0*	Kein Schreibzugriff

Anmerkungen

Der Kanal kann nur verwendet werden, wenn die entsprechende Schnittstelle als RS-232 Schnittstelle konfiguriert ist.

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_UCHAR.

max_read_channel_uchar

Lesen des DSR-Pins (0 oder 1).

10.28.4.6. Direkter Zugriff auf den ESCC

Für Betriebsarten des ESCC, die nicht vom MDD unterstützt werden, bietet der direkte Zugriff auf die Register des ESCC die Möglichkeit, alle Features des Controllers zu verwenden. Dazu muss für jedes verwendete Register ein Kanal geöffnet werden. Zu beachten ist, dass durch *.usWriteMode* angegeben wird, ob es sich um ein Schreibregister des ESCC (*.usWriteMode=IO_MODE_DIRECT* für WR0 bis WR15) oder ein Leseregister des ESCC (*.usWriteMode=0* für RR0 bis RR15) handelt. Nicht alle Schreibregister können zurückgelesen werden. Der MDD hält daher immer die Kopie des zuletzt geschriebenen Wertes. Dieser wird beim Aufruf des Eingabediestes zurückgeliefert.

Strukturelement	Werte	Bedeutung
.usVersion	*0*	Version der CPS-Struktur-Definition
.usChannel	*0 oder 1*	Schnittstelle
.usDevice	*DEVICE_REGISTER*	Konfiguration
.usIndexFirst	*0..15, XSCC_WR7_PRIME*	Register-Nummer
..usIndexLast	*UsIndexFirst*	Register-Nummer
.ulFlags	*0*	Reserviert
.usReadMode	*IO_MODE_RAM*	für Register WR0 bis WR15 und WR7 Prime
	IO_MODE_DIRECT	für Register RR0 bis RR15
.usWriteMode	*IO_MODE_DIRECT*	für Register WR0 bis WR15 und WR7 Prime
	0	für Register RR0 bis RR15

Anmerkungen

Die Block/DMA-Transfer-Funktionalität des ESCC kann nicht verwendet werden (Bit 5 bis 7 in WR1 und Bit 2 in WR14 müssen =0 sein). Auf das Register WR7 Prime wird über den Kanal mit *usIndexFirst= XSCC_WR7_PRIME* zugegriffen. Der MDD sorgt für die richtige Adressierung, so dass Bit 0 in WR15 für den Zugriff nicht beachtet werden muss. Der Zugriff auf das Register WR7 erfolgt ebenfalls ohne Berücksichtigung von Bit 0 in WR15 über den Kanal mit *usIndexFirst=7.*

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_UCHAR.

max_write_channel_uchar (nur für WR0 bis WR15 sowie XSCC_WR7_PRIME)
max_read_channel_uchar

10.28.4.7. Interrupt

Jeder der beiden Kanäle des ESCC kann folgende Ereignisse per Interrupt signalisieren:

- Empfang eines Zeichens
- „Special Receive Condition“ beim Empfang (Overrun, Framing Error, End of Frame, Parity Error)
- Sendepuffer ist leer
- Änderung einer externen Steuerleitung (DCD, CTS, SYNC)
- Besondere Bedingungen wie Break-Signal, Abort, Beginn einer CRC-Übertragung, Transmit Underrun/EOM (näheres im ESCC Handbuch).

Welche Interrupt-Quellen tatsächlich Interrupts auslösen sollen, wird beim Öffnen des Kanals angegeben.

Sollen bei direkter Register-Programmierung des ESCC dessen Interrupts genutzt werden, muss zusätzlich ein Kanal mit der nachfolgenden CPS geöffnet werden, der die MDD-Interrupt-Service-Routine installiert. Diese nimmt den Interrupt entgegen, ruft die Kanal-spezifische Callback-Funktion auf, die bei diesem Kanal sinnvollerweise angegeben werden sollte und setzt anschließend den Interrupt zurück.

Der Interrupt-Kanal übernimmt beim Öffnen die Konfiguration der ESCC-Register WR1, WR2, WR9 und WR15 komplett. Diese dürfen anschliessend nicht mehr durch direkte Register-Zugriffe verändert werden. Bit 2 in WR15 (Status-FIFO enable) wird beim Öffnen des Interrupt-Kanals =0 gesetzt. Falls es anderweitig benutzt werden soll, muss WR15 zurückgelesen werden. Im erhaltenen Wert darf dann nur Bit 2 verändert werden.

Bei Verwendung der gepufferten Kommunikation darf der Interrupt-Kanal nicht geöffnet werden, da die Interrupts dann komplett vom MDD bearbeitet werden.

Strukturelement	**Werte**	**Bedeutung**
.usVersion	*0*	Version der CPS-Struktur-Definition
.usChannel	*0 oder 1*	Schnittstelle
.usDevice	*DEVICE_HW_INT*	Interrupt
.usIndexFirst	*0*	Reserviert
..usIndexLast	*UsIndexFirst*	Reserviert
.usFlags	*_CP_EXCLUSIVE*	Dieses Flag muss gesetzt sein! Der Zugriff auf eine Schnittstelle erfolgt immer exklusiv.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usMode		Bitmaske zum gezielten Einschalten der Interrupt-Quellen. Die folgenden Bits können kombiniert werden.
	XSCC_INT_RCA	Interrupt bei jedem empfangenen Zeichen sowie bei einer Special Condition (s.o.)
	XSCC_INT_SPEC_EOF	Interrupt beim Empfang des Ende-Flags eines SDLC-Frames (End of Frame)
	XSCC_INT_FRAME_ERR	Interrupt bei Auftreten eines Frame-Fehlers (für asynchrone Kommunikation) bzw. CRC-Fehlers (SDLC-Mode)
	XSCC_INT_PARITY_ERR	Interrupt bei Auftreten eines Parity-Fehlers
	XSCC_INT_SPEC_OVERRUN	Interrupt, wenn der Empfangspuffer überläuft
	XSCC_INT_TBE	Interrupt, wenn der Sendepuffer leer ist
	XSCC_INT_BRK	Interrupt bei Empfang von Break oder Abort
	XSCC_INT_TMTUR_EOM	Interrupt nach Versenden der CRC im SDLC-Mode (s. Transmit Underrun/EOM-Status in RR0)
	XSCC_INT_CTS	Interrupt bei Änderung des CTS-Eingangs am ESCC
	XSCC_INT_SYNC	Interrupt bei Änderung des SYNC-Eingangs am ESCC (in async. Modes) bzw.

Strukturelement	Werte	Bedeutung
		des HUNT-Bits (s. WR2 Bit 4)
	XSCC_INT_DCD	Interrupt bei Änderung des DCD-Eingangs am ESCC
	XSCC_INT_ZERO	Interrupt, wenn der Zähler im Baud-Raten-Generator auf 0 herunterzählt.
.usReadMode	*0*	Reserviert
.usWriteMode	*0*	Reserviert

Eingabe- und Ausgabedienst

Nicht implementiert

Callback

Beim Auftreten eines Interrupt-Ereignisses wird die Callback-Funktion des entsprechenden Kanals aufgerufen. Sie bekommt einen ULONG-Wert übergeben. Die unteren 16-Bit in diesem Übergabeparameter kennzeichnen das aufgetretene Interrupt-Ereignis durch eine der Konstanten (bzw. eine Kombination mehrerer Konstanten) in *usMode* signalisiert. Die Bedeutung der oberen 16-Bit ist je nach Typ des Interrupt-Ereignisses unterschiedlich:

- XSCC_INT_TBE: High-Wort =0 (keine Bedeutung)
- XSCC_INT_RCA: High-Wort = empfangenes Zeichen
- Änderung einer Steuerleitung: High-Wort = Inhalt von RR0
- Besondere Bedingungen (s.o.): High-Wort = Inhalt von RR0
- „Special Receive Condition“: High-Wort = Inhalt von RR1

In vielen Anwendungsfällen sind dadurch innerhalb der Callback-Funktion keine weiteren Register-Zugriffe mehr erforderlich. Zu beachten ist, dass die Interrupts je Kanal priorisiert sind:

Empfangspuffer/Special Condition > Sendepuffer > Externe Interrupts

Die Interrupts von Kanal A haben eine höhere Priorität als die von Kanal B.

10.28.5. Treiberprogramm CQmax

Das Treiberprogramm CQmax läuft auf X-MAX-1 und X-MAX-E Modulen. Deren Intelligenz wird zum einen für die Pufferung der Daten und zum anderen für die Abarbeitung beliebiger Protokolle benutzt. Beides läuft komplett unabhängig vom Host-PC. Der Host-PC gibt die Nutzdaten zu beliebiger Zeit mit hoher Geschwindigkeit zur Karte, die sich von da ab allein um das Senden der Daten und um die Erfüllung aller Protokollanforderungen kümmert.

Die Echtzeit-Programme für die serielle Kommunikation lassen sich in zwei Teile zerlegen. Der eine Teil übernimmt das Senden und das Empfangen von Zeichen aus dem bzw. in den Speicher des MAX-PC. Er setzt direkt auf der Hardware auf und stellt alle zeitkritischen Funktionen zur Verfügung. Er muss so programmiert sein, dass auch bei hohen Baudraten keine Zeichen verloren gehen. Dieser Teil wird als **Basiskommunikation** bezeichnet.

Der zweite Teil ist eher nicht zeitkritisch. Er übernimmt das gesamte **Protokollhandling**, also die Erzeugung und Überprüfung von Steuerzeichen und -signalen, Anforderung von Wiederholungen und alle anderen Aktionen, die zur Erfüllung der Protokollspezifikationen nötig sind. Natürlich kann auch in diesem Teil des Programms die Zeit eine wichtige Rolle spielen, wenn zum Beispiel Reaktionstelegramme innerhalb einer bestimmten Zeitspanne versandt werden müssen.

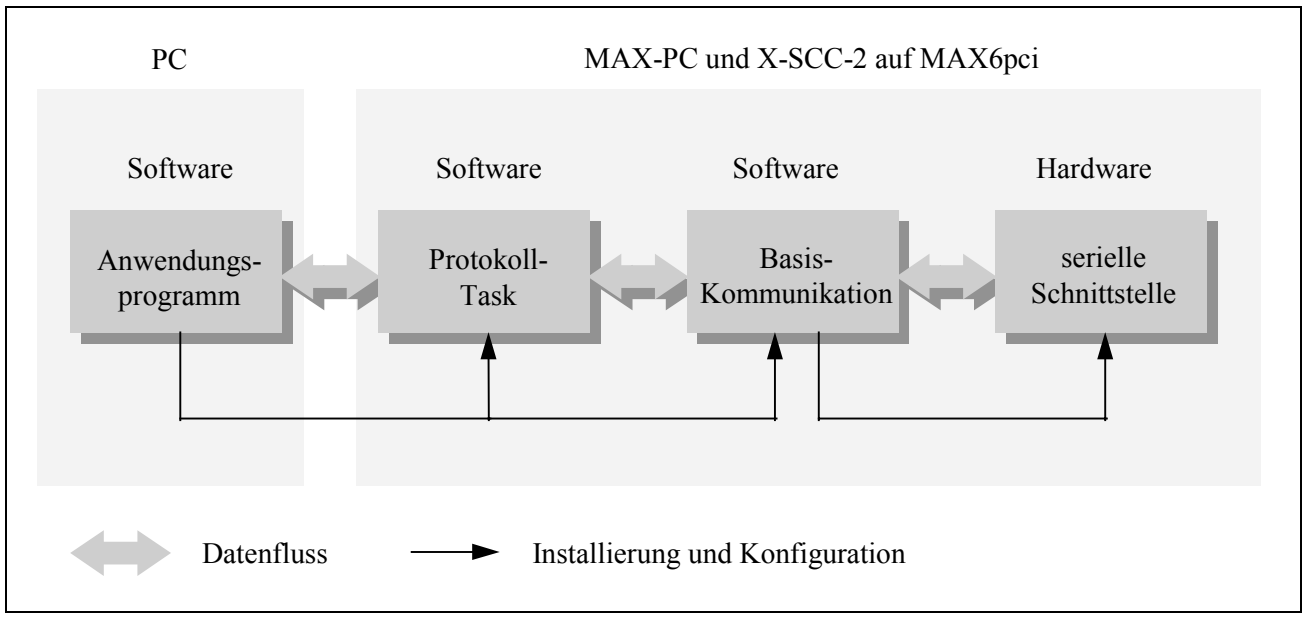


Abb. 1-1: Grundstruktur der seriellen Kommunikation

Der PC tauscht mit dem Teil bzw. mit der Task, die das Protokollhandling übernimmt, die Sende- und Empfangsdaten aus. Wenn ganz ohne Protokoll oder nur mit einem einfachen Handshake (RTS/CTS, XON/XOFF) kommuniziert wird, entfällt die Protokolltask, und der PC tauscht die Daten direkt mit den Tasks der Basiskommunikation aus.

10.28.5.1. Grundstruktur

Jeder Kommunikationsbaustein (SCC) hat 2 serielle Schnittstellen, von denen wiederum jede aus mehreren Gründen Interrupts auslösen kann. Für jede Schnittstelle und jeden ihrer Interrupts wird eine eigene Task installiert. Da aber jedes MAX-Modul X-SCC-2 nur eine einzige Interruptleitung zum MAX-PC hat, muss zunächst entschieden werden, von welcher Schnittstelle und aus welchem Grund der Interrupt ausgelöst wurde, um zu bestimmen, welche Task den Interrupt bearbeiten muss. Diese Aufgabe wird von einer Task übernommen, die unter dem jeweiligen Interrupt des X-SCC-2 installiert und als **Interruptmanager** bezeichnet wird.

Die Basiskommunikationsprogramme bestehen aus einem Interruptmanager (Programm X1PA004.LIB) und einem Kommunikationsprogramm (Programm X1PA005.LIB), die für die Schnittstellen des MAX-Moduls X-SCC-2 geeignet sind. Pro X-SCC-2 muss eine Interruptmanager-Task installiert sein und für jeden verwendeten Kanal eine Kommunikations-Task.

Anschließend werden nur mit den Kommunikationstasks Daten ausgetauscht, der Interruptmanager wird nie aus der Anwendungsebene heraus angesprochen.

Die Kommunikationstasks arbeiten mit zwei unterschiedlichen Datenpuffern:

Der **Sendepuffer** darf nur mit den entsprechenden Funktionen der Basiskommunikationstask beschrieben werden.

Der **Empfangspuffer** enthält die empfangenen Zeichen. Auch er darf nur über den Aufruf von Funktionen der Empfangstask ausgelesen werden.

Für beide Datenpuffer muss beim Initialisieren angegeben werden, wie groß sie sein sollen.

10.28.5.2. Handshake (asynchrone Kommunikation)

Für asynchrone Kommunikation bietet die Basiskommunikation zwei einfache Handshakemechanismen (RTS/CTS und XON/XOFF), mit denen verhindert werden kann, dass der Empfangspuffer überläuft. In beiden Fällen meldet die Empfangsstation, wenn der Empfangspuffer einen bestimmten (einstellbaren) 'Füllstand' erreicht hat. Der Sender sendet der Gegenstation dann vorübergehend keine Zeichen mehr, bis gemeldet wird, dass der 'Füllstand' wieder unter eine (einstellbare) Grenze gefallen ist.

Beim **RTS/CTS**-Handshake muss die RTS-Leitung des Empfängers mit der CTS-Leitung des Senders verbunden werden. Wenn ein bestimmter Füllstand (Meldegrenze für 'Puffer voll') erreicht ist, setzt der Empfänger RTS auf Null, der Sender unterbricht dann den laufenden Sendevorgang. Sobald der Puffer wieder leer ist (Meldegrenze für 'Puffer leer'), wird RTS wieder auf eins gesetzt, und der Sender fährt mit dem Senden von Zeichen fort.

Der **XON/XOFF**-Handshake benötigt im Gegensatz zu RTS/CTS keine zusätzlichen Steuerleitungen. Der Empfänger sendet, sobald die obere Meldegrenze erreicht ist, das Steuerzeichen XOFF zum Sender. XOFF ist ein beliebiges Zeichen, das auf Empfänger- und Senderseite gleich sein muss (üblicherweise 13h). Sobald auf der Sendeseite XOFF empfangen wird, wird das Senden weiterer Zeichen unterbunden, bis der Empfänger durch das Senden von XON wieder Empfangsbereitschaft meldet. XON ist ebenfalls ein frei definierbares Zeichen (üblicherweise 11h). Beim Arbeiten mit XON/XOFF ist zu beachten, dass sowohl XON als auch XOFF reservierte Steuerzeichen sind und in den Nutzdaten nicht vorkommen dürfen. Sie werden immer als Steuerzeichen interpretiert und gelangen nie in den Empfangspuffer.

10.28.5.3. Installation

Der Interruptmanager und die Kommunikationstask(s) werden zunächst installiert:

Für jedes verwendete X-SCC-2 muss jeweils ein Interrupt-Manager (Programm A004h) auf dem MAX-PC unter dem von dem MAX-Modul verwendeten Interrupt installiert (II-Task, kein Datenbereich benötigt) werden, z.B. mit folgendem Befehl in einer INS-Datei:

MAXINST file= "X1PA004.lib" no=A004 task=302 tasktype=MAX_II_TASK

irq=92 autoinit

Für jeden verwendeten Kanal ist jeweils eine Kommunikationstask (Programm A005, NI-Task, Datenbereich wird vom Programm selbst reserviert) zu installieren, , z.B. mit folgendem Befehl in einer INS-Datei:

MAXINST file= "X1PA005.lib" no=A005 task=320 tasktype=MAX_NI_TASK

autoinit

Die Tasknummern können im Prinzip frei gewählt werden; empfohlen wird jedoch folgende Zuordnung:

Interruptkanal	Tasknummer des Interrupt-Managers	Tasknummer der Kommunikationstask
PIRQ-6 (7ch)	301h	318h + Kanalnummer (0 oder 1)
PIRQ-7 (7dh)	302h	320h + Kanalnummer (0 oder 1)
PIRQ-2 (91h)	303h	328h + Kanalnummer (0 oder 1)
PIRQ-3 (92h)	304h	330h + Kanalnummer (0 oder 1)

10.28.5.4. Parametrierung und Initialisierung

Zunächst muss das Basiskommunikationsprogramm parametriert und gestartet werden, so dass es zum Senden und Empfangen bereit ist. Falls Sie aber Parameter (wie z.B. Baudrate) ändern oder die Kommunikation neu beginnen möchten, muss das Programm reinitialisiert werden. Nach dem Ändern von Parametern muss Prozedur 2 aufgerufen werden, damit die Änderungen wirksam werden.

Für die Installation der Basiskommunikation sind alle Parameter des Programms A005h, die mit '*' gekennzeichnet sind, entsprechend zu setzen. Teilweise sind die Parameter vorinitialisiert.

- Die Task-Nummer des zugehörigen Interruptmanagers muss in Parameter 6 jeder Kommunikationstask gesetzt werden. Es empfiehlt sich die Zuordnung wie auf Seite 10-346 vorzunehmen.
- Nach Setzen der gewünschten Parameter der Kommunikationstask und Aufruf der Prozedur 2 ist das Basiskommunikations-Programm konfiguriert und der Speicherplatz für die Puffer entsprechend der in den Parametern angegebenen Größe reserviert.
- Um Zeichen empfangen oder senden zu können, müssen noch die Funktionen 5 (Sendebereitschaft) und 15 (Empfangsbereitschaft) aufgerufen werden.

10.28.5.5. Empfangen

Empfangene Zeichen stehen im Empfangspuffer und können mit der Funktion 17 ausgelesen werden. Dabei wird die gewünschte Anzahl an Zeichen und ein Pointer auf einen Puffer, in den die Zeichen geschrieben werden sollen, an die Funktion

übergeben. Die Funktion überprüft dann den Status des Empfangspuffers, kopiert die empfangenen Zeichen in den angegebenen Puffer und liefert die Anzahl der tatsächlich gelesenen Zeichen zurück oder gegebenenfalls eine Fehlermeldung (siehe Tabelle der Fehlerrückgabecodes, Seite 10-359).

Um den Empfangsstatus zu ermitteln, liest man die zugehörigen Parameter (Nummern 262 und 264), in denen Fehlermeldungen sowie die Anzahl der Zeichen im Puffer enthalten sind. Alternativ kann dafür auch Funktion 6 aufgerufen werden (s.u.).

Das folgende Beispiel kopiert empfangene Zeichen in einen String. Wenn das Funktionsergebnis Null ist, sind die Daten gültig. Der beim Aufruf der Funktion übergebene String muss für mindestens 255 Zeichen Speicher reserviert haben.

Pascal:

```
FUNCTION rcv_string (scc_channel: BYTE; VAR data: STRING): MAX_ERROR;
VAR
  insize   : WORD = 255;
  outsize  : WORD = 0;
  dummyout, task : WORD;
  error    : MAX_ERROR;
BEGIN
  task := $318 + scc_channel;
  { Die Empfangsdaten werden nicht an data, sondern an data[1] }
  { geschrieben, da das erste Byte von Data die Stringlänge enthält.}
  error := max_call_func (hModul, task, 17, outsize, dummyout,
                          insize, indata[1]);
  data[0] := chr(insize);        {Stringlänge einstellen}
  rcv_string := error;
END;
```

C:

```
MAX_ERROR rcv_string (UCHAR scc_channel, char *data)
{
  void      *dummyout;
  USHORT    insize = 255;
  MAX_ERROR error;
  ushort    task, outsize = 0;

  task = 0x318 + scc_channel;
  error = max_call_func (hModul, task, 17, &outsize, &dummyout,
                         &insize, data);

  data[insize] = 0;  /* String mit 'Null' beenden */
  return(error);
```

10.28.5.6. Senden

Zeichen, die gesendet werden sollen, können mit der Funktion 7 an den Sendepuffer übergeben werden. Das Basiskommunikationsprogramm sendet dann die Zeichen. Der Funktion wird die Anzahl der zu sendenden Zeichen und der Zeiger auf einen Puffer, in dem die Zeichen stehen, übergeben. Eine Statusabfrage, ob die Zeichen gesendet oder in den Sendepuffer übernommen werden können, wird durch die Funktion vorgenommen. Im Fehlerfall wird ein Fehlercode zurückgeliefert (siehe Tabelle der Fehlerrückgabecodes, Seite 10-359).

Um den Sendestatus zu ermitteln (z.B. ob alle Zeichen gesendet wurden), liest man die zugehörigen Parameter (256 und 258), die Statusmeldungen (2 Byte) und die Anzahl der im Sendepuffer (4 Byte) enthaltenen Zeichen. Alternativ kann dafür auch Funktion 6 aufgerufen werden (s.u.).

Das folgende Beispiel zeigt eine Funktion zum Senden eines Strings. Der Rückgabewert ist Null, wenn der String in den Sendepuffer eingetragen werden konnte.

Pascal:

```
FUNCTION send_string (scc_channel: BYTE; data: STRING): MAX_ERROR;
VAR
  dummyin, task   : WORD;
  error           : MAX_ERROR;
  outsize, insize : WORD;
BEGIN
  { Die Sendedaten werden mit data[1] übergeben, da bei einer }
  { Übergabe mit data auch das erste Byte der Stringvariablen }
  { (= Stringlänge) übertragen würde.                         }
  task    := $318 + scc_channel;
  outsize := length(data);
  insize  := 0;
  error   := max_call_func (hModul, task, 7, outsize, data[1],
                            dummysize, dummyin);

  send_string := error;
END;
```

C:

```
MAX_ERROR send_string (UCHAR scc_channel, char *data)
{
  void        *dummyin;
  USHORT      outsize, insize, task;
  MAX_ERROR   error;

  task    = 0x318 + scc_channel;
  outsize = strlen(data);
  insize  = 0;
  error = max_call_func (hModul, task, 7,&outsize, &data, &insize, dummyin);

  return(error);
}
```

10.28.5.7. Die Parameter des Programmes X1PA005.LIB

Alle Parameter, die in der folgenden Tabelle nicht beschrieben sind, sind reserviert und dürfen nicht geändert werden.

Nr.	Typ	Init	Zugr[2]	Bedeutung des Parameters
0	UCHAR	0	R	Genereller Programmstatus: 0 = geladen, 1 = läuft, 2 = gestoppt, 3 = gestoppt nach Fehler
1	UCHAR	0	R	Fehlerinformation: gültig, wenn Parameter 0 = 3, siehe Fehlertabelle 10.28.5.8)
4	USHORT	0	R/W	Programmnummer des zugeh. Interruptmanagers
6*	USHORT	0	R/W	Tasknummer des zugehörigen Interruptmanagers
8*	UCHAR	0	R/W	Slot^Layer-Nr. des X-SCC-2 (Bit 4..7 = Slot, Bit 0..3 = Layer).
9*	UCHAR	0	R/W	Kanalnummer: Kanal A = 0, B = 1
10	UCHAR	1	R/W	Physikalische Verbindung (X-SCC-2) 1 = RS-232 3 = RS-422 9 = RS-485/2 10 = RS-485/1

[2] Zugriff auf Parameter: R= Nur Lesen, R/W = Lesen und Schreiben

Nr.	Typ	Init	Zugr[2]	Bedeutung des Parameters	
11*	UCHAR	1	R/W	Typ des Kommunikations-Controllers 1 = SCC 2 = ESCC 3-255 = reserviert	:
16	UCHAR	1	R/W	Kommunikationstyp: 1 = asynchron 2 = Mono-Sync 3 = Bi-Sync 4 = external Sync 5 = SDLC[2] 6 = SDLC Loop Mode[2] 7-255 = reserviert	
17	UCHAR	1	R/W	Daten-Codierung Senden und Empfangen: 1 = NRZ (Non-Return to Zero) 2 = NRZI[2] (Non-Return to Zero Inverted) 3 = FM1[2] (Biphase Mark) 4 = FM0[2] (Biphase Space) 5 = Manchester[2] (Biphase Level) 6 bis 255 = reserviert	
18*	UCHAR	1	R/W	Clock-Quelle Senden: 1 = Standard Baudratengenerator 2 = Von extern über CTS an RTxC[3] 3 = Intern, Ausgabe auf Clock-Leitung[3] 4 = Autogeneration (DPLL) [2, 3] 5 = Von extern über Ri an TRxC [3] 6-255 = reserviert	
19*	UCHAR	1	R/W	Clock-Quelle Empfangen (Bedeutung siehe Parameter 18)	
20	ULONG	1474 h	R/W	Quarzfrequenz	

[2] Noch nicht implementiert

[3] Taktein- bzw. Ausgang entspricht dem Baudratentakt

Nr.	Typ	Init	Zugr[2]	Bedeutung des Parameters
24*	ULONG	9600	R/W	Baudrate für Senden[2]
28*	UCHAR	8	R/W	Zeichenlänge für Senden in Anzahl Bit: 5, 6, 7, 8
29*	UCHAR	1	R/W	Anzahl Stopbit (Senden und Empfangen) 1 = ein Stopbit, 2 = zwei Stopbits, 3 = 1,5 Stopbits
30*	UCHAR	0	R/W	Paritätsbit-Generierung (Senden und Empfangen): 0 = keine, 1 = ungerade, 2 = gerade
32	UCHAR	0	R/W	Sync Character 1 (Low-Byte, WR6)
33	UCHAR	0	R/W	Sync Character 2 (High-Byte, WR7)
34*	ULONG	9600	R/W	Baudrate für Empfangen[2]
38*	UCHAR	8	R/W	Zeichenlänge für Empfangen[3] in Anz. Bit: 5, 6, 7, 8
39	UCHAR	0	R	Bitmaske für Zeichenlänge[4]
40*	ULONG	0	R/W	Sendepuffer-Größe (in Byte)
48*	ULONG	0	R/W	Empfangspuffer-Größe (in Byte)

[2] Es sind beliebige Baudraten einstellbar. Nach Aufruf der Prozedur 2 wird hier die nächste realisierbare Baudrate eingetragen.

[3] Standardmäßig liefert der Kommunikationsbaustein immer 8 Bit zurück, wobei bei Zeichenlängen kleiner 8 das höchstwertige Bit entsprechend der Parität gesetzt wird. Das Programm liefert normalerweise nur die gewünschte Anzahl von Bits/Zeichen. Alle anderen Bits werden = 0 gesetzt. Soll das Paritätsbit mitgeliefert werden, kann das über eine Bitmaske definiert werden (siehe Parameter 39)

[4] Bitmaske, die definiert, welche Bits zurückgeliefert werden sollen. Bit = 1 bedeutet: Bit wird nicht gelöscht.

Nr.	Typ	Init	Zugr[2]	Bedeutung des Parameters
54	UCHAR	0	R/W	Empfangenes Byte bei Parity-Error verwerfen oder speichern: 0 = speichern, 1 = verwerfen
55	UCHAR	0	R/W	Sync-Features: Bit 0 = 1: TxCRC Enable Bit 1 = 1: SDLC/CRC-16 Bit 2 = 1: CRC Preset I/O Bit 3 = 1: 6-Bit/8-Bit Sync Bit 4 = 1: RxCRC Enable Bit 5 = 1: Sync Character Load Inhibit Bit 6 = 1: Enter Hunt Mode Bit 7 = 1: Address Search Mode (SDLC)
256	USHORT	0	R	Sendestatus (Bitmap) Bit 0 = Sendepuffer (Software) ist leer Bit 1 = alle Zeichen gesendet (Hardware) Bit 2 bis 14 sind reserviert Bit 15 = Sendeteil angehalten (durch Befehl oder Protokoll)
258	ULONG	0	R	Anzahl Zeichen im Sendepuffer
262	USHORT	0	R	Empfangsstatus (Bitmap) Einmaliges Auftreten der Bedingung setzt das zugehörige Bit (Ausnahme Bit 15). Rücksetzung Bit 0 bis 4 durch Funktion 16 Bit 0 = Zeichenverlust wegen übergelaufenem Empfangspuffer Bit 1 = Zeichenverlust wegen Überlastung im ser. Controller Bit 2 = Parity-Fehler Bit 3 = Frame-Fehler Bit 4 = Break-Detection Bit 5 bis 14 sind reserviert Bit 15 = Empfangsteil angehalten (durch Befehl oder Protokoll)
264	ULONG	0	R	Anzahl Byte im Empfangspuffer

Nr.	Typ	Init	Zugr[2]	Bedeutung des Parameters
268	USHORT	0	R	Zustand der Eingangs-Steuerleitungen (Bitmap) Bit 0 = CTS (Clear To Send) Bit 1 = DCD (Data Carrier Detected) Bit 2 = RI (Ring Indicator)
290	USHORT	0	R	Zähler, Sendepuffer war voll[3]
292	USHORT	0	R	Zähler, Empfangspuffer übergelaufen[3]
294	USHORT	0	R	Zähler, Zeichenverlust (Overrun)[3]
296	USHORT	0	R	Zähler, Parity-Fehler[3]
298	USHORT	0	R	Zähler, Framing-Fehler[3]
300	USHORT	0	R	Zähler, Break-Fehler[3]
318	USHORT	0	R/W	Protokolltyp 0 = kein Protokoll 1 = XON/XOFF 2 = RTS/CTS
726	USHORT	1800 h	R/W	Bit 13 in diesem Wort definiert, ob die Abschlusswiderstände im RS-422 und RS-485 Mode eingeschaltet oder ausgeschaltet werden: Bit13 = 0: ausgeschaltet Bit13 = 1: eingeschaltet. Die übrigen Bits dürfen nicht verändert werden.

[3] Parameter wird durch Aufruf der Funktion 2 auf Null gesetzt

Hinweis zu Parameter 32 und 33:

Parameter 32:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Sync7	Sync6	Sync5	Sync4	Sync3	Sync2	Sync1	Sync0	Monosync, 8 Bits
Sync1	Sync0	Sync5	Sync4	Sync3	Sync2	Sync1	Sync0	Monosync, 6 Bits
Sync7	Sync6	Sync5	Sync4	Sync3	Sync2	Sync1	Sync0	Bisync, 16 Bits
Sync3	Sync2	Sync1	Sync0	1	1	1	1	Bisync, 12 Bits
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	SDLC
ADR7	ADR6	ADR5	ADR4	x	X	x	X	SDLC (Adress Range)

Parameter 33:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Sync7	Sync6	Sync5	Sync4	Sync3	Sync2	Sync1	Sync0	Monosync, 8 Bits
Sync5	Sync4	Sync3	Sync2	Sync1	Sync0	x	x	Monosync, 6 Bits
Sync15	Sync14	Sync13	Sync12	Sync11	Sync10	Sync9	Sync8	Bisync, 16 Bits
Sync11	Sync10	Sync9	Sync8	Sync7	Sync6	Sync5	Sync4	Bisync, 12 Bits
0	1	1	1	1	1	1	0	SDLC

Hinweis zur Nutzung des SDLC-Protokolls:

Folgende Parameter müssen gesetzt werden:

Parameter 11: =2, wenn ESCC verwendet werden soll

Parameter 16: = 5 für Kommunikationstyp SDLC

Parameter 29: = 0 für Sync-Mode Enable

Parameter 32: = Adresse für SDLC-Geräte (s.o.)

Parameter 33: = SDLC-Flag (= 7eh) (s.o.)

Parameter 55: = z.B. 55h (TxCRC Enable, CRC Preset I/O, Rx CRC Enable, Enter Hunt Mode)

10.28.5.8. Fehlercodes in Parameter 1

(Wenn Parameter 0 = 3 ist, dann steht in Parameter 1 die Fehlerursache.)

Fehlernummer in Parameter 1	**Erklärung**
0	Reserviert
1	Nicht genügend Platz für Sendepuffer-Reservierung
2	Falscher Parameter für Sendepuffer-Reservierung
3	Unbekannter Fehler während Sendepuffer-Reservierung
4	Falscher Parameter für Sendepuffer-Reservierung
7	Ungültige Sendepuffer-Nr.
8	Sendepuffer wird gerade benutzt (locked)
11	Nicht genügend Platz für Empfangspuffer-Reservierung
12	Falscher Parameter für Empfangspuffer-Reservierung
13	Unbekannter Fehler während Empfangspuffer-Reservierung
14	Falscher Parameter für Empfangspuffer-Reservierung
17	Ungültige Puffernummer für Empfangspuffer
18	Empfangspuffer wird gerade benutzt (locked)
30	Fehler beim Aufruf einer externen Funktion (aufgerufen durch Aktions-Filter)
40	Fehler bei Interrupt-Service-Aufruf
50	Keine Quarzfrequenz in EEPROM eingetragen
99	Unbekannter Fehler

10.28.5.9. Funktionen und Prozeduren von CQmax

Das Basiskommunikationsprogramm umfasst globale Prozeduren (ohne Übergabeparameter und Antwort) und globale Funktionen (mit Übergabe von Parametern und Antwort). In der folgenden Tabelle sind die Prozeduren in der Spalte 'Typ' mit P gekennzeichnet, die Funktionen mit F. Beim Aufruf von Funktionen müssen eine Reihe von Parametern übergeben werden. Sie sind in der folgenden Tabelle mit den Bezeichnern angegeben, mit der die Funktion in den Hochsprachen-Bibliotheken MAXW32.LIB und MAXRT.LIB aufgerufen wird.

Nr.	Typ	Bedeutung der Funktion
2	P	Start/Restart der Kommunikation (Konfiguration, Speicherreserv.)
4	P	Baudrate aus Parameter 24 und 34 übernehmen und einstellen
5	F	**Sendebereitschaft ein- und ausschalten,** optional Sendepuffer löschen Hin: insize = 2 outsize = 0 indata = *Kontroll-Wort* (s.u.) Rück: error = Fehlernummer *Kontroll-Wort:* 0 = Senden anhalten, Puffer nicht löschen 1 = Senden starten, Puffer nicht löschen 2 = Senden anhalten, Puffer löschen 3 = Senden starten, Puffer löschen
6	F	**Sendestatus melden** Hin: insize = 0 outsize = 6 outdata = *Rückgabe-Struktur* (s.u.) Rück: outsize = 6 error = Fehlernummer *Rückgabe-Struktur:* Sendestatus (USHORT), Anzahl Zeichen im Sendepuffer (ULONG), Bedeutung wie Parameter 256 und 258

Nr.	Typ	Bedeutung der Funktion
7	F	**Zeichenkette an Sendepuffer übergeben** Hin: insize = Anzahl zu übergebender Zeichen outsize = 0 indata = *Übergabe-Puffer* (s.u.) Rück: error = Fehlernummer *Übergabe-Puffer:* Datenstruktur, die *insize* zu sendende Bytes enthält.
15	F	**Empfangsbereitschaft ein- und ausschalten**, optional Empfangspuffer löschen Hin: insize = 2 outsize = 0 indata = *Kontroll-Wort* (s.u.) Rück: error = Fehlernummer *Kontroll-Wort:* 0 = Empfangen anhalten, Puffer nicht löschen 1 = Empfangen starten, Puffer nicht löschen 2 = Empfangen anhalten, Puffer löschen 3 = Empfangen starten, Puffer löschen
16	F	**Empfangsstatus melden** Hin: insize = 2 outsize = 6 outdata = *Rückgabe-Struktur* (s.u.) Rück: outsize = 6 error = Fehlernummer *Rückgabe-Struktur:* Empfangsstatus (USHORT), Anzahl Zeichen im Empfangspuffer (ULONG), Bedeutung wie Parameter 262 und 264
17	F	**Zeichenkette aus Empfangspuffer übernehmen** Hin: insize = 0 outsize = Anzahl angeforderter Zeichen outdata = *Rückgabe-Puffer* (s.u.) Rück: outsize = Anzahl gelesener Zeichen error = Fehlernummer *Rückgabe-Puffer:* Datenstruktur, die die gelesenen Zeichen aufnehmen kann, also mindestens *outsize* Byte umfasst. Nach dem Aufruf sind die ersten *outsize* Byte gültig.

Nr.	Typ	Bedeutung der Funktion
33	F	**Steuerleitungs-Ausgänge setzen** Hin: insize = 4 outsize = 0 indata = *Übergabe-Struktur* (s.u.) Rück: error = Fehlernummer *Übergabe-Struktur*: Wort 1: Maske für die betroffenen Steuerleitungen (USHORT), Bit = 1: Steuerleitung soll geändert werden Wort 2: Information, wie die Steuerleitung gesetzt werden soll Bitmap (Wort 1 und 2): Bit 0 = RTS Bit 1 = DTR Bit 2 = TMT-Break Bit 3 bis 15 reserviert
34	F	**Zustand der Steuerleitungs-Eingänge lesen** Hin: insize = 0 outsize = 2 outdata = *Rückgabe-Struktur* (s.u.) Rück: error = Fehlernummer *Rückgabe-Struktur:* Information, wie die Steuerleitung gesetzt wurde (Word) Bitmap: Bit 0 = CTS Bit 1 = DCD Bit 2 = RI Bit 3 = DSR Bit 4 bis 15 reserviert

10.28.5.10. Bedeutung der Fehlerrückgabecodes

Alle zurückgelieferten Fehler sind Meldungen vom Betriebssystem. Die folgende Tabelle zeigt, welche Fehler bei den einzelnen Funktionen auftauchen können und was sie bedeuten:

Funktion	Fehlernummer	Bedeutung
5	22h oder 23h 26h	Sendepuffer gesperrt Sendepuffer-Nummer ungültig
6	26h	Sendepuffer-Nummer ungültig
7	22h oder 23h 24h 26h	Sendepuffer gesperrt Sendepuffer voll Sendepuffer-Nummer ungültig
15	22h oder 23h 26h	Empfangspuffer gesperrt Empfangspuffer-Nummer ungültig
16	26h	Empfangspuffer-Nummer ungültig
17	22h oder 23h 26h	Empfangspuffer gesperrt Empfangspuffer-Nummer ungültig

Bei den Fehlermeldungen 'Puffer voll' bzw. 'Puffer gesperrt' können Sie den Funktionsaufruf zu einem späteren Zeitpunkt wiederholen. Damit sich der Pufferzustand ändern kann, müssen Sie die Kontrolle nach dem ersten Funktionsaufruf (der den Fehler gemeldet hat) zuerst wieder an das Betriebssystem zurückgeben.

10.28.5.11. Die Aktions-Filter des Programms X1PA005

Das Basis-Kommunikations-Programm X1PA005 kann unter bestimmten, einstellbaren Bedingungen Funktionen anderer Tasks aufrufen. Die Bedingungen stellen Filter für die Ausführung einer Aktion dar. Wir sprechen daher von Aktions-Filtern. Die Übergabe-Konvention für die aufgerufene Funktion entspricht der in Kapitel 6 beschriebenen Bibliotheksfunktionen. Daher können Funktionen einer anderen Task abhängig von Ereignissen, die in der Basiskommunikation aufgetreten sind, aufgerufen werden. Dieser Mechanismus ist in erster Linie zur Implementierung von Protokollen vorgesehen. Zur Zeit sind 10 Filter-Aktionen definiert. Sie können mit dem Parameter "Aktions-Filter-Maske" vom Programm einzeln aktiviert oder unterdrückt werden. Jedem Aktions-Filter ist ein Parameter-Satz von 12 Byte Länge zugeordnet (Parameter siehe Kapitel 10.28.5.13). Dieser enthält die Nummer (2 Byte) der Task, die die aufzurufende Funktion enthält, und die Nummer der Funktion (2 Byte), die bei zutreffender Bedingung aufgerufen werden soll. Es folgt ein 4 Byte langes Argument, das von Filter-Aktion zu Filter-Aktion unterschiedliche Bedeutung hat. Danach kommt die Funktionsadresse der aufzurufenden Funktion im Format Segment:Offset (4 Byte). Diese Adresse wird beim Aufruf der Startfunktion (Prozedur 2) vom Programm X1PA005 berechnet, wenn das zugehörige Bit in der Aktions-Filter-Maske gesetzt ist. Die Task, die die aufzurufende Funktion enthält, muss zu diesem Zeitpunkt bereits installiert sein.

Der aufgerufenen Funktion wird eine Struktur mit diversen Status-Informationen übergeben (s.u.).

Deklaration der Action-Filer-Funktionen

in C: **VOID MAXEXPORT name (USHORT *insize, VOID *inptr,**
USHORT *outsize, VOID *outptr);

in Pascal: **PROCEDURE name (VAR insize : Word; VAR inptr;**
VAR outsize: Word; VAR outptr);

Parameter	*insize*:	Anzahl der Daten, die der Funktion übergeben werden
	inptr:	Zeiger auf die Übergabedaten
	outsize:	Maximale Anzahl der Bytes, die zurück erwartet werden
	outptr:	Zeiger auf den Bereich, in dem die Task ihre Daten zurückgibt (Rückgabewert).

10.28.5.12. Konfiguration der Aktions-Filter

In Parameter 56 von Programm X1PA005 (siehe Kapitel 10.28.5.13) können z.Zt. 10 verschiedene Aktions-Filter ausgewählt werden.

Aktions-Filter 0

Bei jedem empfangenen Zeichen soll die zugeordnete Funktion aufgerufen werden.

Das Argument hat keine Bedeutung.

Im Rückgabewert (1 Byte) der Funktion wird festgelegt, ob das empfangene Zeichen nachträglich noch im Empfangs-Puffer gespeichert wird (= 1) oder nicht (= 0).

Aktions-Filter 1

Die zugeordnete Funktion wird aufgerufen wenn ein empfangenes Zeichen sich in einem Vergleichsstring befindet.

Das Argument enthält den Zeiger (Format Segment:Offset) auf den Vergleichsstring. Das erste Zeichen im Vergleichsstring enthält die Anzahl der nachfolgenden zu vergleichenden Zeichen.

Im Rückgabewert (Byte) der Funktion wird festgelegt, ob das empfangene Zeichen nachträglich noch im Empfangs-Puffer gespeichert wird (= 1) oder nicht (= 0).

Aktions-Filter 2

Die zugeordnete Funktion wird bei Empfangsfehlern aufgerufen.

Im Argument (untere 2 Byte) können Empfangsfehler maskiert werden. Die Bedeutung der Maske entspricht der Beschreibung des Parameters 262. Ein gesetztes Bit (=1) führt zum Aufruf der Funktion, ein nicht gesetztes unterbindet den Aufruf.

Es gibt kein Rückgabewort.

Aktions-Filter 3

Funktionsaufruf bei Empfangs-Füllstands-Überschreitung.

Das Argument enthält die obere Empfangs-Füllstands-Grenze.

Es gibt kein Rückgabewort.

Wenn dieses Filter benutzt wird, muss auch Filter 4 benutzt werden.

Aktions-Filter 4

Funktionsaufruf bei Empfangs-Füllstands-Unterschreitung.

Das Argument enthält die untere Empfangs-Füllstands-Grenze.

Es gibt kein Rückgabewort.

Wenn dieses Filter benutzt wird, muss auch Filter 3 benutzt werden.

Aktions-Filter 5

Funktionsaufruf bei Sende-Füllstands-Überschreitung

Das Argument enthält die obere Sende-Füllstands-Grenze.

Es gibt kein Rückgabewort.

Wenn dieses Filter benutzt wird, muss auch Filter 6 benutzt werden.

Aktions-Filter 6

Funktionsaufruf bei Sende-Füllstands-Unterschreitung.

Das Argument enthält die untere Sende-Füllstands-Grenze.

Es gibt kein Rückgabewort.

Wenn dieses Filter benutzt wird, muss auch Filter 5 benutzt werden.

Aktions-Filter 7

Funktionsaufruf, wenn alle Zeichen physikalisch gesendet sind.

Das Argument hat keine Bedeutung.

Es gibt kein Rückgabewort.

Aktions-Filter 8

Funktionsaufruf, wenn sich eine Eingangs-Steuerleitung (CTS, DCD, RI, DSR) ändert.

Das Argument enthält eine Maske für Steuerleitungs-Eingänge, die der Bedeutung des Parameters 268 entspricht. Ein gesetztes Bit (=1) führt zum Aufruf der Funktion, ein nicht gesetztes unterbindet den Aufruf.

Es gibt kein Rückgabewort.

Aktions-Filter 9

Funktionsaufruf bei nicht behebbarem Laufzeitfehler.

Die Fehlerursache muss durch Lesen des Parameter 1 des Programms X1PA005 ermittelt werden.

Das Argument hat keine Bedeutung.

Es gibt kein Rückgabewort.

10.28.5.13. Zusätzliche Parameter für die Aktionsfilter

Die folgenden Parameter des Programms X1PA005 dienen zur Konfiguration der Aktionsfilter.

Nr.	Typ	Init	Zugr	Bedeutung des Parameters
56	ULONG	0	R/W	Aktions-Filter-Maske (Bitmap), Aufruf bei Bit 0 = 1: jedem empfangenen Zeichen Bit 1 = 1: empfangenem Zeichen im Vergleichs-string enthalten Bit 2 = 1: Fehler beim Empfang (Overflow, Parity, etc.) Bit 3 = 1: Empfangs-Füllstands-Überschreitung Bit 4 = 1: Empfangs-Füllstands-Unterschreitung Bit 5 = 1: Sende-Füllstands-Überschreitung Bit 6 = 1: Sende-Füllstands-Unterschreitung Bit 7 = 1: leerem Empfangspuffer Bit 8 = 1: Änderung der Steuerleitungs-Eingänge (z.B. CTS, DCD, RI, DSR). Bit 9 = 1: nicht behebbarem Laufzeitfehler Bit 10 bis 31 sind reserviert

Nr.	Typ	Init	Zugr	Bedeutung des Parameters
60				Aktions-Filter 0: bei jedem empfangenen Zeichen
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	–	Argument: ohne Bedeutung
	ULONG	0	R	Funktionsadresse (Segment:Offset)
72				Aktions-Filter 1: empfangenes Zeichen ist im Vergleichsstring (siehe Parameter 320)
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: Zeiger auf Vergleichsstring (Segment:Offset)
	ULONG	0	R	Funktionsadresse (Segment:Offset)
84				Aktions-Filter 2: Empfangsfehler
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: Fehlermaske (Bit 0 bis 15, Bitmap), Bedeutung wie Parameter 262
	ULONG	0	R	Funktionsadresse (Segment:Offset)
96				Aktions-Filter 3: Empfangs-Füllstands-Überschreitung
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: obere Empfangs-Füllstands-Grenze
	ULONG	0	R	Funktionsadresse (Segment:Offset)
108				Aktions-Filter 4: Empfangs-Füllstands-Unterschreitung
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: untere Empfangs-Füllstands-Grenze
	ULONG	0	R	Funktionsadresse (Segment:Offset)

Nr.	Typ	Init	Zugr	Bedeutung des Parameters
120				Aktions-Filter 5: Sende-Füllstands-Überschreitung
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: obere Sende-Füllstands-Grenze
	ULONG	0	R	Funktionsadresse (Segment:Offset)
132				Aktions-Filter 6: Sende-Füllstands-Unterschreitung
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: untere Sende-Füllstands-Grenze
	ULONG	0	R	Funktionsadresse (Segment:Offset)
144				Aktions-Filter 7: Alle Zeichen gesendet
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: ohne Bedeutung
	ULONG	0	R	Funktionsadresse (Segment:Offset)
156				Aktions-Filter 8: Änderung einer Steuerleitung
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: Maske für Steuerleitungen, Bedeutung wie bei Status der Steuerleitungen
	ULONG	0	R	Funktionsadresse (Segment:Offset)
168				Aktions-Filter 9: nicht behebbarer Laufzeitfehler
	USHORT	0	R/W	Aktions-Task
	USHORT	0	R/W	Aktions-Funktion
	ULONG	0	R/W	Argument: ohne Bedeutung
	ULONG	0	R	Funktionsadresse (Segment:Offset)
180	6 * 12 = 72 Byte	0	R	Aktions-Filter 10 bis 15: reserviert Task, Funktion, Argument, Funktionsadresse (Aufbau wie Filter 0)

Nr.	Typ	Init	Zugr	Bedeutung des Parameters
318	USHORT	0	R/W	Protokolltyp 0 = kein Protokoll 1 = XON/XOFF : Aktions-Filter 1, 3 und 4 werden verwendet, die Funktionen 38, 36 und 37 werden aufgerufen 2 = RTS/CTS: Aktions-Filter 3, 4 und 8 werden verwendet, die Funktionen 40, 41 und 42 werden aufgerufen
320	16 Byte	0	R/W	Vergleichsstring für Filteraktion 1 Aufbau: Das erste Zeichen enthält die Anzahl der zu vergleichenden Zeichen im String, danach folgen die Zeichen (maximal 15), bei Protokolltyp = 1 werden automatisch die XON/XOFF Zeichen (11H, 13H) eingetragen.

10.28.6. Anschlusspins des Moduls (bezogen auf den Modul-Stecker A)

Pin		**RS-232**	**RS-422**	**RS-485/1**	**RS-485/2**	**20 mA**
A	**B**					**Current Loop**
1	11	**DCD**	**RCV-**	**n.c.**	**n.c.**	(DCD RS-232)
2	12	**DSR**	**CTS-**	**n.c.**	**CTS-**	(DSR RS-232)
3	13	**RCV**	**RCV+**	**n.c.**	**n.c.**	(RCV RS-232, ohne Funktion)
4	14	**RTS**	**RTS-**	**RTS-/CTS-**	**RTS-**	(RTS RS-232)
5	15	**TMT**	**TMT-**	**RCV-/TMT-**	**RCV-/TMT-**	(TMT RS-232)
6	16	**CTS**	**CTS+**	**n.c.**	**CTS+**	(CTS RS-232)
7	17	**DTR**	**TMT+**	**RCV+/TMT+**	**RCV+/TMT+**	(DTR RS-232)
8	18	**Ri**	**(Ri-Input)**	**(Ri-Input)**	**(Ri-Input)**	(Ri RS-232)
9	19	**GND**	**GND**	**GND**	**GND**	(GND)
10	20	**CLKio**	**RTS+**	**RTS+/CTS+**	**RTS+**	(CLKio RS-232)
21	31	(-12V)	(-12V)	(-12V)	(-12V)	**-12V Ausgang**
22	32	(TMT+)	(TMT+)	(TMT+)	(TMT+)	**TMT+**
23	33	(TMT-)	(TMT-)	(TMT-)	(TMT-)	**TMT-**
24	34	(GND)	(GND)	(GND)	(GND)	**GND**
25	35	(CCS1+)	(CCS1+)	(CCS1+)	(CCS1+)	**CCS1+**
26	36	(RCV+)	(RCV+)	(RCV+)	(RCV+)	**RCV+**
27	37	(RCV-)	(RCV-)	(RCV-)	(RCV-)	**RCV-**
28	38	(GND)	(GND)	(GND)	(GND)	**GND**
29	39	(CCS2+)	(CCS2+)	(CCS2+)	(CCS2+)	**CCS2+**
30	40	(+12V)	(+12V)	(+12V)	(+12V)	**+12V Ausgang**

<n.c.> Diese Eingänge haben in dieser Betriebsart keine Funktion, sind aber hochohmig galvanisch angeschlossen. Die Spannungen an diesen Pins darf die max. erlaubte Eingangsspannung nicht überschreiten (siehe Besondere Eigenschaften).

<1> CCSn (Constant Current Source) sind 4 Konstantstromquellen, je 2 für jeden Kanal.

10.28.7. Besondere Eigenschaften

Parameter	Wert	Einheit
Anzahl serieller Schnittstellen	2	
per Software wählbar	RS-232, RS-422, RS-485 oder 20 mA Current Loop (dann galv. getrennt)	
max. Baudrate je Kanal	920,75	kBaud
FIFO (je Kanal)	8	Byte
Controller	16 MHz ESCC	
RS-232		
Eingangsspannung, max.	±25	V
Eingangsschwelle Low, min.	0,6	V
Eingangsschwelle High, max.	2,0	V
Hysterese, typ.	0,5	V
Eingangswiderstand, min./typ./max.	3/5/7	kΩ
Ausgangsspannung (3 kOhm an GND), min./typ.	±5/±5,4	V
Ausgangsstrom (Ausgang an GND), max.	±60	mA
RS-422 und RS-485		
Eingangswiderstand, min.	48	kΩ
Eingangsstrom, max. (-7V .. +12V)	-0,15/0,25	mA
Differentielle Eingangsschwelle, min./max.	-200/-50	mV
Hysterese, typ.	30	mV
Differentielle Ausgangsspannung, min. (RS-422/RS-485) (RS-485 = 27 Ω, RS-422 = 50 Ω)	2/1,5	V
20mA Current Loop		
Empfänger		
min. Eingangsstrom für log. 1	12	mA
max. Eingangsstrom für log. 0	3	mA
Hysterese, typ.	0,8	mA
Sender		
min. Ausgangsstrom für log. 1	30	mA
max. Ausgangsstrom für log. 0	2	mA
max. Ausgangsspg. für log. 1 (I=20mA)	2,7	V
Temperatur-Bereich, Betrieb	-40 .. +85	°C

Abmessungen		29x58x8	mm
Gewicht, X-SCC-2/R / X-SCC-2/U		9,45/10,85	g
Stromaufnahme			
3,3V, typ.	(X-SCC-2/R / X-SCC-2/U)	255/305	mA
12V	(X-SCC-2/R)	55	mA
-12V	(X-SCC-2/R)	25	mA