Kommunikationsmodule zu MODULAR-4®/486

SORCUS

4. Auflage
15.09. 2001
SORCUS Computer GmbH
Im Breitspiel 11
69126 Heidelberg

M-COM-2

M-COM-2/P und /G

M-DAS-A

M-COM-8

M-IEC-1

M-DPM-12

M-DPS-12

M-CAN-1

M-ETH-1

Inhaltsverzeichnis

3. M-COM-2/P 3-1

4. M-DAS-A 4-1

5. M-COM-8 5-1

6. M-IEC-1 6-1

7. M-DPM-12 7-1

8. M-DPS-12 8-1

9. M-CAN-1 9-1

10. M-ETH-1 10-1

Index iii-1

Vorwort

Das vorliegende Handbuch beschreibt eine Auswahl von SORCUS Prozeß-Bus-Modulen (= SP-Bus-Modulen), die für verschiedene Aufgabenstellungen im Bereich Kommunikation entwickelt wurden:

M-COM-2	M-DPM-12
M-COM-2/P und /G	M-DPS-12
M-DAS-A	M-CAN-1
M-COM-8	M-ETH-1
M-IEC-1	

Alle aufgeführten Module sind auf allen MODULAR-4/486 und /586 Karten einsetzbar.

Das Buch ist inhaltlich in zwei Teile gegliedert. Das erste Kapitel, die Einführung, beschäftigt sich mit allgemeinen Sachverhalten, die für mehrere oder gar sämtliche Module gelten. Danach wird jedes einzelne Modul in einem gesonderten Kapitel behandelt.

Wir möchten darauf hinweisen, daß die in diesem Buch enthaltenen Beschreibungen neu bearbeitet wurden und daher von früheren Versionen abweichen können.

Als ergänzende Lektüre empfehlen wir Ihnen das MODULAR-4/486 Handbuch, das ausführliche Informationen zu den Basiskarten enthält.

Wir haben versucht, die Bezeichnungen in diesem Buch einheitlich zu halten. Bei den Steckern und Jumpern war uns dies aus technischen und historischen Gründen leider nicht möglich. Wir hoffen, daß wir dadurch keine Verwirrung stiften.

Darüber hinaus sind wir stets offen und dankbar für Verbesserungsvorschläge und stehen Ihnen gern für zusätzliche Informationen zur Verfügung.

SORCUS Computer GmbH

Notizen

1. Einführung

Mitgelieferte Software

Die im Lieferumfang eines Moduls enthaltenen Datenträger sind nicht kopiergeschützt. Die Programme dürfen für Zwecke des Anwenders beliebig oft kopiert werden. Wiederverkäufer können die Programme auch verändern und weiter verkaufen. Eine besondere Genehmigung der SORCUS Computer GmbH ist dazu nicht erforderlich.

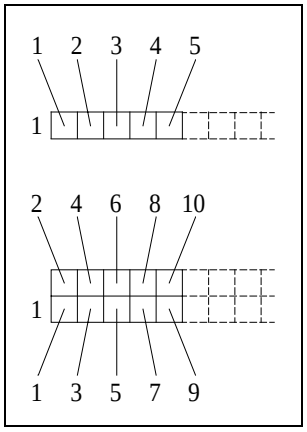
Eine Kopie der Datenträger wird, wie im Handbuch zum Betriebssystem Ihres PCs beschrieben, hergestellt. Verwahren Sie die Originale sicher vor schädigenden Einflüssen.

Da sich die Zahl der Programme auf den mitgelieferten Original-Datenträgern entsprechend dem aktuellen Entwicklungsstand ändert, ist an dieser Stelle kein Inhaltsverzeichnis angegeben. Aktualisierte Versionen können jederzeit ohne zusätzliche Gebühr von der SORCUS Homepage **www.sorcus.com** heruntergeladen werden.

Alle Programme auf den Original-Datenträgern sind als Hilfsmittel gedacht, um Ihnen den Einsatz der MODULAR-4 Karte zu erleichtern und zu verdeutlichen. Eine Garantie für die fehlerfreie Funktion dieser Programme übernimmt die Firma SORCUS Computer GmbH jedoch nicht. Insbesondere werden jede Haftung, Gewährleistung und eventuelle Schadenersatzansprüche, die sich aus dem Einsatz der MODULAR-4 Karte sowie der mitgelieferten Software ergeben könnten, ausgeschlossen.

Jumper

Vor dem Einbau eines Moduls, das Jumper (= Kurzschlußbrücken) enthält, sollten Sie die Einstellung aller Jumper kontrollieren und dokumentieren. Die Zählweise der einzelnen Pins in einreihigen und zweireihigen Jumperfeldern entnehmen Sie der nebenstehenden Abbildung, Pin 1 ist in den Lageplänen der Module jeweils gekennzeichnet.



Tragen Sie die Einstellung aller Jumper in das EEPROM des Moduls ein, da einige Programme (z.B. SNW32) aus diesen Daten die Konfiguration des Moduls ermitteln.

EEPROM

Alle Module verfügen über einen nichtflüchtigen Speicher, ein sogenanntes EEPROM, in dem Initialisierungs-, Konfigurations- und Korrekturdaten abgelegt sind. Nach dem Laden und Starten des Betriebssystems OsX auf der Basiskarte werden die EEPROM-Inhalte immer in den Parameterbereich des Betriebssystems kopiert.

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt.

Die Daten im EEPROM können auf zwei Arten gelesen und geschrieben werden:

1. Mit dem PC-Hilfsprogramm SNW32, das im Lieferumfang jeder Basiskarte enthalten ist.
2. Aus einem PC-Programm heraus durch Aufruf der entsprechenden Bibliotheksroutinen. Die Beschreibung dieser Routinen finden Sie im Benutzerhandbuch Ihrer Basiskarte.

! *Wenn Daten des EEPROMs direkt gelesen oder geändert werden sollen, darf kein Anwendungsprogramm auf der Karte laufen (Empfehlung: vorher und hinterher sicherheitshalber einen Reset der Karte durchführen).*

Beachten Sie, daß das Eintragen von Konfigurationsdaten in das EEPROM ein Modul nicht automatisch konfiguriert. Das EEPROM speichert lediglich Werte, die per Software ausgelesen und auf dem Modul eingestellt werden müssen.

Die Bedeutung von WORT-0 ist bei allen Modulen gleich. Hier sind Modultyp und Revision eingetragen.

Das Bit-0 in WORT-1 legt bei allen Modulen fest, ob das Modul nach dem Einschalten des PCs und nach einem Hardware-Reset der MODULAR-4 Karte entsprechend den Eintragungen im EEPROM konfiguriert und initialisiert wird.

Hochsprachenbibliotheken

Verwendung der Bibliotheken

Die Hochsprachenbibliotheken für die Module sind geeignet für die Basiskarten MODULAR-4/486, MODULAR-4/Z80 und MODULAR-4/Z280. Die MODULAR-4/Z80 und /Z280 Karten werden in dieser Beschreibung nicht behandelt. Bei der MODULAR-4/486 Karte können die Bibliotheken sowohl für den PC (unter **DOS** und **Windows 3.x**) als auch für Echtzeitprogramme, die lokal auf der Karte laufen, benutzt werden.

Die Bibliotheken sind für den Einsatz der Hochsprachen **Borland Pascal (Version 7.0), Turbo Pascal (Version 6.0 und 7.0)** und **Borland C++ (Version 3.1)** geeignet. Getestet wurden sie mit Borland Pascal Version 7.0 und Borland C++ Version 3.1. Die Windows-DLLs können natürlich auch mit anderen Programmiersprachen (z.B. Visual Basic) verwendet werden. Sie sind vorwiegend dazu gedacht, das Ansprechen der Module zu demonstrieren, und sind nicht bezüglich ihrer Geschwindigkeit optimiert.

Es wird keine Überprüfung der Übergabeparameter vorgenommen.

Allgemeine Hinweise:

1. Der Parameter **micro_slot** wird in allen Routinen verwendet und gibt den Modulsteckplatz an. Er kann bei der "großen" MODULAR-4/486 Karte die Werte 1 bis 4 annehmen, bei der "kleinen" MODULAR-4/486 Karte die Werte 1 bis 2 und bei der MODULAR-4/486 Karte mit Modulextender die Werte 2 bis 10.

2. Vor der Verwendung von Bibliotheksroutinen vom **PC** aus müssen folgende Routinen einmal aufgerufen werden:

   ml8_bib_startup bzw. **ml7_bib_startup** und
   ml8_reset bzw. **ml7_reset** oder **ml8_start** bzw. **ml7_start**

Für **Echtzeitprogramme**, die auf der MODULAR-4/486 Karte laufen sollen, ist kein solcher Aufruf erforderlich.

3. Bei vielen Modulbibliotheken muß außerdem eine eigene Initialisierungsroutine aufgerufen werden. Ob solch ein Aufruf nötig ist und welche Funktion das ist, entnehmen Sie jeweils dem Kapitel 'Hochsprachenbibliotheken' der einzelnen Modulbeschreibungen.

4. Die Angabe **EXPORT** bei der Beschreibung der Routinen für C bedeutet:

bei DOS-Programmen:	far pascal
bei Windows-Programmen (DLL):	_export far pascal

! *Wenn ein Modul aus verschiedenen Tasks oder vom PC und von einer Task angesprochen wird, kann es zu Problemen kommen.*
Insbesondere bei Digitalausgängen ist das Setzen einzelner Bits in der Regel nicht möglich. Sehr kritisch ist auch die Verwendung der Bibliotheken in Tasks, die sich gegenseitig unterbrechen können. Dazu gehört auch die Programmierung vom PC aus, da der PC auf der Karte laufende Tasks unterbrechen kann.
Das Problem läßt sich lösen, in dem man alle Modulzugriffe in eine zentrale Steuertask verlegt. Alle anderen Tasks und der PC benutzen dann keine Modulbibliotheks-Befehle mehr, sondern rufen Prozeduren und Funktionen der Steuertask auf, um das Modul anzusprechen.

Einbinden der Bibliotheken

Im folgenden wird beschrieben, wie Sie die Bibliotheken in Ihre Programme einbinden können. Dabei werden zwei Bezeichner benutzt, die als Platzhalter für modulspezifische Namen stehen. Die richtigen Namen für die einzelnen Module entnehmen Sie jeweils dem Kapitel 'Hochsprachenbibliotheken'.

MODULE: Angabe des Unterverzeichnisses, in dem sich alle Programme und Bibliotheken zum jeweiligen Modul befinden.

libname: Name der Bibliothek. Dieser Name wird für alle Dateien, die zur Bibliothek gehören, verwendet. Er setzt sich aus der Modulkennummer (dreistellige Zahl) und der Bezeichnung '_LIB' zusammen (z.B. libname = 'M032_LIB' für M-COM-2)

MODULAR-4/486: Pascal-Programme unter DOS

Hierfür werden folgende Dateien mitgeliefert:

Unterverzeichnis	**Dateiname**	**Kommentar**
MODULE\PASCAL	*libname*.PAS	Source-Datei PASCAL
MODULE\ML8\DOS\PASCAL	*libname*.OBJ	.OBJ-Datei
MODULE\ML8\DOS\PASCAL	*libname*.TP	Konfigurationsdatei

Verwenden Sie die Datei *libname*.PAS und geben Sie in der USES-Anweisung ML8BIB bzw. ML7BIB und *libname* an.

MODULAR-4/486: C-Programme unter DOS

Hierfür werden folgende Dateien mitgeliefert:

Unterverzeichnis	**Dateiname**	**Kommentar**
MODULE\C	*libname*.C	Source-Datei in C
MODULE\C	*libname*.H	Header-Datei für C
MODULE\ML8\DOS\C	*libname*.PRJ	Projektdatei für C

Binden Sie die Include-Dateien ML8BIB.H bzw. ML7BIB.H und *libname*.H in Ihr C-Programm ein (#include "Dateiname"). Linken Sie ML8BIB.LIB bzw. ML7BIB.LIB und *libname*.OBJ zu Ihrem C-Programm dazu.

MODULAR-4/486: Pascal-Programme unter Windows (DLL)

Hierfür werden folgende Dateien mitgeliefert:

Unterverzeichnis	Dateiname	Kommentar
MODULE\PASCAL	*libname*.PAS	Source-Datei PASCAL
MODULE\ML8\WINDOWS\PASCAL	*libname*.DLL	.DLL-Datei
MODULE\ML8\WINDOWS\PASCAL	*libname*.LIB	.LIB-Datei
MODULE\ML8\WINDOWS\PASCAL	*libname*.DEF	.DEF-Datei
MODULE\ML8\WINDOWS\PASCAL	*libname*.TPW	.TPW-Datei
MODULE\ML8\WINDOWS\PASCAL	*libname*.TP	Konfigurationsdatei

Geben Sie in der USES-Anweisung WML8BIB bzw. WML7BIB und *libname* an. Die Dateien WML8BIB.DLL und WML8BIB.TPW bzw. WML7BIB.DLL und WML7BIB.TPW, sowie *libname*.DLL und *libname*.TPW müssen im Default-Verzeichnis sein. Die Import-Unit *libname*.TPW wurde aus der Datei *libname*.PAS erzeugt.

MODULAR-4/486: C-Programme unter Windows (DLL)

Hierfür werden folgende Dateien mitgeliefert:

Unterverzeichnis	Dateiname	Kommentar
MODULE\C	*libname*.C	Source-Datei in C
MODULE\C	*libname*.H	Header-Datei für C
MODULE\ML8\WINDOWS\C	*libname*.PRJ	Projektdatei
MODULE\ML8\WINDOWS\C	*libname*.DLL	.DLL-Datei
MODULE\ML8\WINDOWS\C	*libname*.LIB	.LIB-Datei
MODULE\ML8\WINDOWS\C	*libname*.DEF	.DEF-Datei

Binden Sie die Include-Dateien ML8BIB.H bzw. ML7BIB.H und *libname*.H in Ihr C-Programm ein (#include "Dateiname"). Linken Sie WML8BIB.LIB bzw. WML7BIB.LIB und *libname*.LIB zu Ihrem C-Programm dazu. Die Dateien WML8BIB.DLL bzw. WML7BIB.DLL und *libname*.DLL müssen zur Laufzeit im Default-Verzeichnis sein.

MODULAR-4/486: Pascal-Programme auf der Karte

Hierfür werden folgende Dateien mitgeliefert:

Unterverzeichnis	Dateiname	Kommentar
MODULE\PASCAL	*libname*.PAS	Source-Datei in PASCAL
MODULE\ML8\RT\PASCAL	*libname*.OBJ	.OBJ-Datei für PASCAL
MODULE\ML8\RT\PASCAL	*libname*.TP	Konfigurationsdatei

Verwenden Sie die Datei *libname*.PAS und geben Sie in der USES-Anweisung ML8RTBIB bzw. ML7RTBIB und *libname* an.

MODULAR-4/486: C-Programme auf der Karte

Hierfür werden folgende Dateien mitgeliefert:

Unterverzeichnis	Dateiname	Kommentar
MODULE\C	*libname*.C	Source-Datei in C
MODULE\C	*libname*.H	Header-Datei für C
MODULE\ML8\RT\C	*libname*.PRJ	Projektdatei für C

Binden Sie die Include-Dateien ML8RTBIB.H bzw. ML7RTBIB.H und *libname*.H in Ihr C-Programm ein (#include "Dateiname"). Linken Sie ML8RTBIB.LIB bzw. ML7RTBIB.LIB und *libname*.OBJ zu Ihrem C-Programm dazu.

Weitere Hinweise zu den Hochsprachenbibliotheken finden Sie im Benutzerhandbuch zur MODULAR-4 Karte in den Kapiteln "PC-Hochsprachenbibliotheken" und "Echtzeitbibliotheken".

Programmierung mit I/O-Zugriffen

Wenn für Ihr PC-Betriebssystem oder für Ihre Programmiersprache keine Modulbibliotheken vorliegen oder wenn Sie geschwindigkeitsoptimierte Echtzeitprogramme schreiben wollen, müssen Sie die Funktionseinheiten der Module direkt programmieren. Das geschieht mit I/O-Zugriffen des Prozessors auf das jeweilige Modul.

Pro Modul stehen maximal 32 I/O-Adressen zur Verfügung. Die anzusprechende I/O-Adresse ergibt sich jeweils als Summe aus der steckplatzabhängigen Modul-Basis-Adresse (MBA, siehe folgende Tabelle) und einem Adreß-Offset.

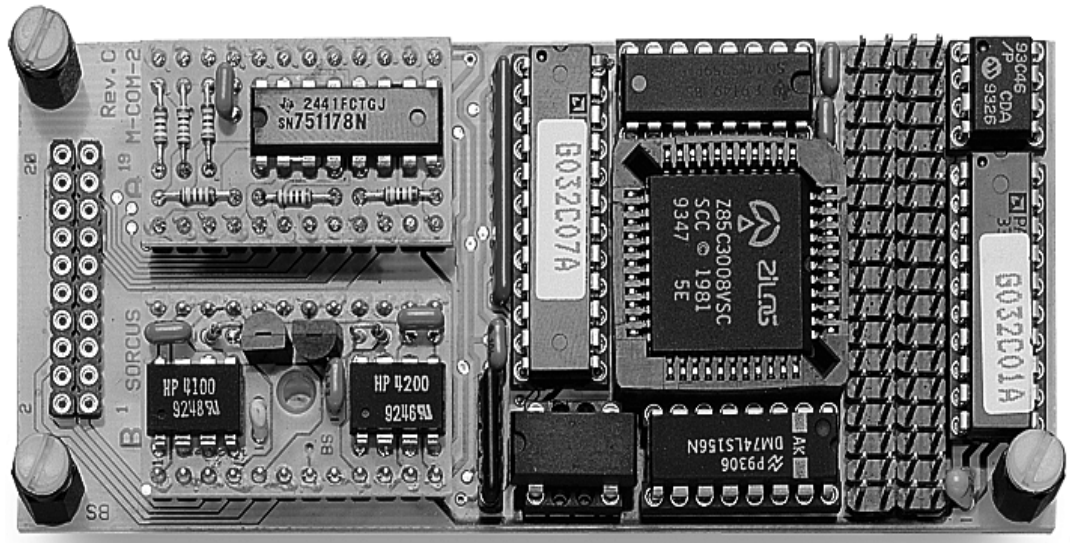
Steck-platz	1	2	3	4	5	6	7	8	9	10
MBA	400h	500h	600h	700h	420h	440h	460h	480h	4a0h	4c0h

Die I/O-Adressen und ihre Funktion sind bei den Modulen jeweils im Kapitel 'I/O-Adressen' zu finden. Dort ist auch angegeben, ob auf die jeweilige Adresse mit 8- oder mit 16-Bit-Befehlen zugegriffen werden darf. In den Tabellen gelten folgende Abkürzungen:

R: nur Lesezugriffe erlaubt
W: nur Schreibzugriffe erlaubt
R/W: Schreib- und Lesezugriffe erlaubt
8: nur 8-Bit-Zugriffe erlaubt
16: nur 16-Bit-Zugriffe erlaubt
x: die gelesenen bzw. geschriebenen Daten sind ohne Bedeutung

2. M-COM-2

Zwei universelle serielle Schnittstellen



Hinweis: Diese Beschreibung bezieht sich auf Rev. C, D, E und F des Moduls.

Funktionsbeschreibung

Das Modul M-COM-2 enthält zwei serielle synchrone/asynchrone Schnittstellen. Es ist für die Basiskarten MODULAR-4/486, MODULAR-4/Z80 und MODULAR-4/Z280 in je drei Bestückungsvarianten lieferbar (siehe folgende Tabelle 1-1). Als Standardbestückung ist das Modul mit einem Quarzoszillator von 7,3728 MHz ausgerüstet, alle Bestückungsvarianten können auf Wunsch auch mit einer anderen Quarzfrequenz, z. B. 4,9152 MHz geliefert werden.

Tabelle 1-1*:* Übersicht M-COM-2 Versionen

Modulversion	**für Basiskarte**	**Bestückungsvariante** [1]
M-COM-2/8	MODULAR-4/486	C-Link-Adapter je Kanal
M-COM-2/4	MODULAR-4/Z80, /Z280	C-Link-Adapter je Kanal
M-COM-2/P/8	MODULAR-4/486	Plastik-LWL für beide Kanäle
M-COM-2/G/8	MODULAR-4/486	Glas-LWL für beide Kanäle
M-COM-2/P/4	MODULAR-4/Z80, /Z280	Plastik-LWL für beide Kanäle
M-COM-2/G/4	MODULAR-4/Z80, /Z280	Glas-LWL für beide Kanäle

Bei der Version M-COM-2/8, auf die sich diese Beschreibung bezieht, erfolgt die Konfiguration der physikalischen Schnittstellen über C-Link-Adapter und ist damit an alle üblichen Pegel anpaßbar. Jede Schnittstelle ist über einen C-Link-Adapter getrennt konfigurierbar. Es werden C-Link-Adapter der Firma COMPLEX International (Ci) verwendet. Folgende C-Link-Adapter sind zur Zeit lieferbar (siehe Tabelle 1.2):

[1] LWL = Lichtwellenleiter

Tabelle 1-2: Übersicht C-Link-Adapter

C-Link-Adapter	Physikalische Schnittstelle	Kurzbeschreibung
CL232S	RS-232 bis 120 kBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS, DTR, DSR, RI, DCD Zusätzliche Funktionen: Mode 0: RI als Clock-Eingang Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang
CL232A/i	RS-232 bis 120 kBaud	Modem-Steuerleitungen (in Mode 0): TMT, RCV, RTS, CTS, DTR, DSR, RI, DCD Zusätzliche Funktionen: Mode 0: Zusätzliche RS-232-Leitung EXT als Clock-Eingang 1 [1] Mode 0: RI als Clock-Eingang 2
CL232A/o	RS-232 bis 120 kBaud	Modem-Steuerleitungen (in Mode 5): TMT, RCV, RTS, CTS, DTR, DSR, RI, DCD Zusätzliche Funktionen: Mode 5: Zusätzliche RS-232-Leitung EXT als Clock-Ausgang [1]
CL232i	RS-232 isol. bis 120 kBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang
CL422S	RS-422 bis 10 MBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang

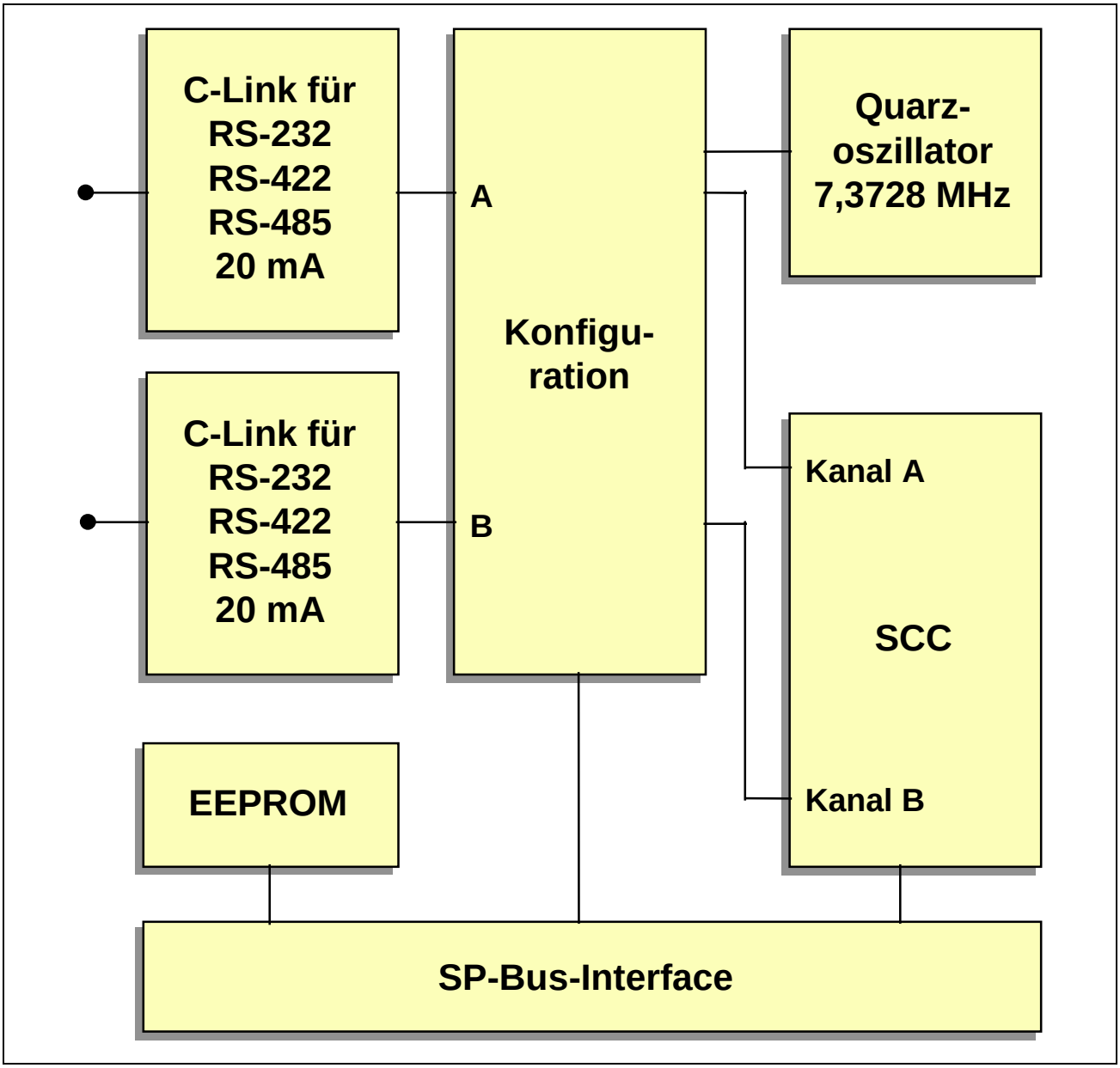
[1] EXT ist kein RS-232-Standardsignal. Es ist auch bei den üblichen 9-poligen oder 25-poligen D-Submin.-Steckern bzw. Buchsen nicht vorhanden. Die 9-polige Schnittstelle wurde hier um einen Pin erweitert, um bei RS-232 alle Modem-Steuersignale und zusätzlich einen Taktein- oder -ausgang zu ermöglichen. Bei jenen C-Links, die diesen Pin verwenden, kann er, wenn das entsprechende Signal nicht benötigt wird, frei bleiben (= nicht angeschlossen).

C-Link-Adapter	Physikalische Schnittstelle	Kurzbeschreibung
CL422S	RS-485 bis 10 MBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 0 und 2: RTS-Treiber disabled Mode 2: CTS als Clock-Eingang Mode 3: CTS als Clock-Eingang, RTS enabled Mode 5: RTS als Clock-Ausgang
CL422i	RS-422 isol. bis 10 MBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang
CL485i/U	RS-485 isol. bis 20 MBaud	Bis 20 MBaud geeignet, Umschaltung von Senden auf Empfangen per Software oder automatisch (z. B. für SDLC/HDLC)
CL200A	20 mA isol. bis 120 kBaud	20 mA Current Loop, zwei Konstantstromquellen auf dem C-Link. Passiv oder aktiv konfigurierbar (wenn passiv, dann galvanisch getrennt)

Die beiden seriellen Schnittstellen des Moduls sind unabhängig voneinander und unterschiedlich mit C-Link-Adaptern konfigurierbar. Es muß nur die Schnittstelle mit einem C-Link-Adapter bestückt werden, die auch verwendet wird. Das Modul ist mit einem SCC-Baustein Z8530 (bzw. 85C30 = CMOS-Version) oder optional Z85230 (= verbesserte und erweiterte Version mit größeren FIFOs) ausgerüstet. Außerdem enthält das Modul einen eigenen Quarzoszillator und zwei Baudratengeneratoren, ist also unabhängig vom CPU-Takt oder von Timern der Basiskarte. Für besondere Einsatzbedingungen ist das Modul auch so konfigurierbar, daß die Baudraten für Senden und Empfangen bei einem Kanal unterschiedlich eingestellt werden können. Außerdem kann je nach C-Link der Sende- und/oder Empfangstakt von außen bzw. ein Takt nach außen geliefert werden. Die Richtung ist per Software umschaltbar, vorausgesetzt der aufgesteckte C-Link-Adapter ist hierfür vorgesehen. Gleiches gilt für die automatische Umschaltung von Senden auf Empfangen, z. B. bei RS-485-Schnittstellen. Die Umschaltung kann per Software oder automatisch durch ein Steuersignal des SCC erfolgen.

Das Modul ist interruptfähig, der Interrupt vom Modul zur Basiskarte kann per Software angewählt und an IRQ-A bis IRQ-F der Basiskarte gelegt werden.

Blockschaltbild



Technische Daten

Parameter		Wert	Einheit
Anzahl serieller Schnittstellen		2	-
Serieller Kommunikationsbaustein (Option:		Z8530-8 Z85230-20)	-
Interruptfähig zur Basiskarte [1]		ja	-
Modem-Steuerleitungen je Schnittstelle (je nach C-Link)		RTS, CTS, DCD, DTR, Ri, DSR, EXT[2], CLK_{in}, CLK_{out}	-
Baudratengeneratoren Quarzoszillator: 7,3728 MHz (Option: 4,9152 MHz)		2	-
Versorgungsspannungen, ohne C-Links (von der Basiskarte; einige C-Links benötigen +12 Volt und -12 Volt, s.u.)		+5	V
Stromaufnahme ohne C-Links:	+5/+12/-12 Volt (typ.)	110/0/0	mA
mit 2 C-Links CL200A:	+5/+12/-12 Volt (typ.)	160/40/0[3]	mA
mit 2 C-Links CL232S:	+5/+12/-12 Volt (typ.)	124/0,5/0,8	mA
mit 2 C-Links CL232i:	+5/+12/-12 Volt (typ.)	250/-/-	mA
mit 2 C-Links CL232A/i:	+5/+12/-12 Volt (typ.)	220/70/70	mA
mit 2 C-Links CL232A/o:	+5/+12/-12 Volt (typ.)	122/1/1,2	mA
mit 2 C-Links CL422S:	+5/+12/-12 Volt (typ.)	314/-/-	mA
mit 2 C-Links CL485i/P:	+5/+12/-12 Volt (typ.)	390/-/-	mA
mit 2 C-Links CL485i/U:	+5/+12/-12 Volt (typ.)	390/-/-	mA
Betriebstemperatur		0 bis 70	°C
Abmessungen (L x B x H)		106 x 45 x 15	mm

[1] Per Software ist einer von 6 Interrupt-Eingängen der Basiskarte wählbar. Der Interrupt-Ausgang des Moduls ist aktiv Low. Da die Interrupt-Eingänge der Basiskarte flankengesteuert sind, muß der entsprechende Interrupt-Eingang also auf negative Flanke programmiert werden.

[2] Erklärung zum Signal EXT siehe oben bei der Beschreibung der C-Links

[3] abhängig von Anschlußkonfiguration

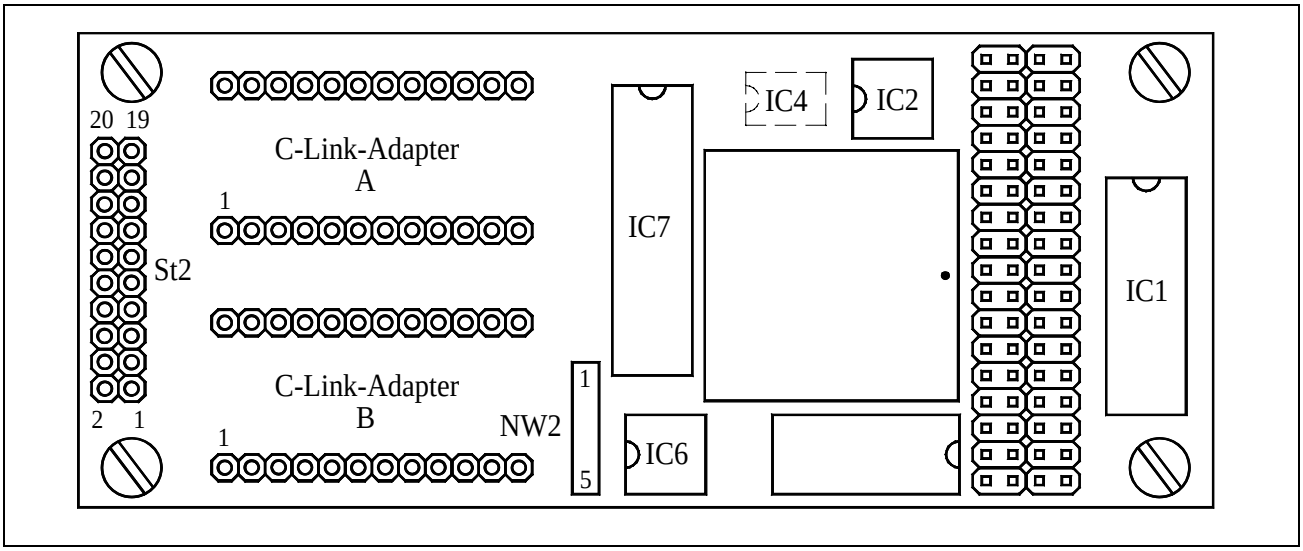
Lieferumfang

- Modul M-COM-2
- 20-poliger Pfostenstecker für Flachbandkabel
- CD bzw. Diskette und Programmbibliotheken (Pascal und C)

Konfiguration und Einbau

Vor dem Aufstecken des Moduls auf die Basiskarte sollten die aufgesteckten C-Link-Adapter überprüft werden (Kanal A und B und Pin 1 sind auf dem Modul und Pin 1 auf den C-Links gekennzeichnet). Das Modul und die C-Links enthalten keine Jumper, alle Einstellungen werden nach dem Einbau per Software vorgenommen.

Lageplan, Rev. F



Das Netzwerk NW2 ist nicht bestückt. Das gestrichelt eingezeichnete IC4 ist auf der Lötseite angebracht.

EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0110	0010 0000	2620h	Modultyp M-COM-2
1	0000 0000	0000 0001	0001h	Initialisierung
2	0010 0010	0001 0000	2210h	Bestückung
3	0000 0111	0011 0111	0737h	Quarzfrequenz
4	0000 0011	0000 0011	0303h	Bestückung und Umbau (Kanal A)
5	0000 0011	0000 0011	0303h	Bestückung und Umbau (Kanal B)
6	0000 0000	0000 0000	0000h	Physikalisches Interface (Kanal A)
7	0000 0000	0000 0000	0000h	Physikalisches Interface (Kanal B)
8	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	WORT-0: Kennung
								0	0	1	0	0	0	0	0	Modultyp: 32 = M-COM-2
				0	1	1	0									Revision: 1 = A, 2 = B, 3 = C, ...,6 = F
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort kann eingestellt werden, ob das Modul nach dem Einschalten und bei einem Hardware-Reset entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															1	Init nach Hardware-Reset: 0 = nein, 1 = ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-2: Bestückung

Bei der Angabe der GAL-Versionen (GALs sind vorprogrammierte ICs) steht das "x" für die Revision des Moduls. Bei einem Modul M-COM-2, Rev. F, trägt das GAL für IC1 also z. B. ein Etikett mit der Aufschrift "G032F01B".

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	WORT-2: Bestückung (werks. Einst.)
																geändert am: von:
												0	0	0	0	Typ SCC: 0 = SCC (8530) 1 = ESCC (85130) 2 = ESCC (85230)
								0	0	0	1					Max. Takt (SCC): 0 = 6 MHz 1 = 8 MHz 2 = 10 MHz 3 = 16 MHz 4 = 20 MHz
				0	0	1	0									GAL-Version IC1: 0 = unbekannt 1 = Vers. A ("G032x01A") 2 = Vers. B ("G032x01B") 3 = Vers. C ("G032x01C")
0	0	1	0													GAL-Version IC7: 0 = unbekannt 1 = Vers. A ("G032x07A") 2 = Vers. B ("G032x07B") 3 = Vers. C ("G032x07C") 4 = Vers. D ("G032x07D")

WORT-3: Quarzfrequenz

Die Angabe erfolgt mit 4 Ziffern in MHz mit 2 Vorkomma- und 2 Nachkommastellen. 4,9152 MHz würde also aufgerundet auf 4,92 und mit 4 Dezimalziffern 0492 (= 0492h) eingegeben. 0000h bedeutet, daß der Quarzoszillator nicht bestückt ist, die folgende Angabe 0737h bedeutet 7,37 MHz.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	1	WORT-3: Quarzfrequenz (werks. Einst.)
																geändert am: von:
												0	1	1	1	2. Ziffer nach Komma
								0	0	1	1					1. Ziffer nach Komma
				0	1	1	1									2. Ziffer
0	0	0	0													1. Ziffer

WORT-4 und WORT-5: Bestückung und Umbau

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1	WORT-4: Hardware Kanal A (werks. Einst.)
																geändert am: von:
															1	1 = C-Link möglich 0 = nicht vorgesehen
														1		1 = 24-poliger Sockel 0 = nicht vorbereitet
													0			1 = C-Link eingelötet 0 = nicht eingelötet
								0	0	0	0	0				Reserviert
							1									1 = Interface möglich 0 = nicht vorgesehen
						1										1 = Sockel 0 = nicht vorbereitet
					0											1 = fest eingelötet 0 = nicht eingelötet
				1												1 = Modul umgebaut auf Revision F 0 = nicht umgebaut
0	0	0	0													Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	WORT-5: Hardware Kanal B (werks. Einst.)
																geändert am: von:
															1	1 = C-Link möglich 0 = nicht vorgesehen
														1		1 = 24-poliger Sockel 0 = nicht vorbereitet
													0			1 = C-Link eingelötet 0 = nicht eingelötet
								0	0	0	0	0				Reserviert
							1									1 = Interface möglich 0 = nicht vorgesehen
						1										1 = Sockel 0 = nicht vorbereitet
					0											1 = fest eingelötet 0 = nicht eingelötet
0	0	0	0	0												Reserviert

WORT-6 und WORT-7: Physikalisches Interface (Kanal A und B)

In WORT-6 steht ein Code für das physikalische Interface, z. B. das eingesetzte C-Link, für Kanal A, in WORT-7 für Kanal B. Statt eines C-Link kann die physikalische Schnittstelle auch direkt auf dem Modul eingelötet sein, z. B. bei bestimmten Lichtwellenleitern (siehe auch WORT-4 und WORT-5).

Code		Physikalisches Interface	isol.	z. B. C-Link
0	(00h)	keines	-	-
1	(01h)	RS-232, mit allen Modem-Leitungen	nein	CL232S
2	(02h)	RS-232, nur 5 V Versorgung	nein	CL232V
3	(03h)	RS-232	nein	CL232A
4	(04h)	RS-232 (RCV, TMT, RTS/CLK_{out}, CTS/CLK_{in})	ja	CL232i
5	(05h)	20 mA	ja	CL200A
6	(06h)	RS-422 / RS-485	nein	CL422S
7	(07h)	RS-422 / RS-485	ja	CL422i
8	(08h)	Lichtleiter HP-System, seitlich	ja	CL800Q
9	(09h)	Lichtleiter HP-System, oben	ja	CL800L
10	(0ah)	Lichtleiter Toshiba Glas PCS (TODX296)	ja	CL800Q
11	(0bh)	Lichtleiter Toshiba Plastik APF (TODX297)	ja	CL800L
12	(0ch)	RS-232, Pin EXT als CLK_{in}	nein	CL232A/i
13	(0dh)	RS-232, Pin EXT als CLK_{out}	nein	CL232A/o

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-6: Interface Kanal A (werks. Einst.)
																geändert am: von:
								0	0	0	0	0	0	0	0	Code für Interface A
0	0	0	0	0	0	0	0									Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-7: Interface Kanal B (werks. Einst.)
																geändert am: von:
								0	0	0	0	0	0	0	0	Code für Interface B
0	0	0	0	0	0	0	0									Reserviert

Konfiguration und Steckerbelegung

Übersicht: Konfigurationsmöglichkeiten bei M-COM-2 (Rev. F) mit GAL "G032F07C", "G032F07D" und "G032F07E"

Drei Ports, d.h. 3 Bit stehen zur Konfiguration für jede Schnittstelle in Verbindung mit einem C-Link-Adapter zur Verfügung: Port A1, A2 und A3 für Schnittstelle A, Port B1, B2 und B3 für Schnittstelle B. Damit lassen sich je Schnittstelle per Software 8 Modes einstellen (Mode 0, 1, 2, 3, 4, 5 und 7 sind belegt, Mode 6 ist reserviert). Sie passen das Modul M-COM-2 an den jeweils aufgesteckten C-Link-Adapter an, bei einigen C-Link-Adaptern kann damit auch die Funktionalität verändert werden.

Abb. 2-1 zeigt die prinzipielle Verschaltung von SCC, C-Link und GAL für Kanal A des Moduls M-COM-2 (Rev. F02). In allen folgenden Abbildungen bedeutet: I := Eingang, O := Ausgang.

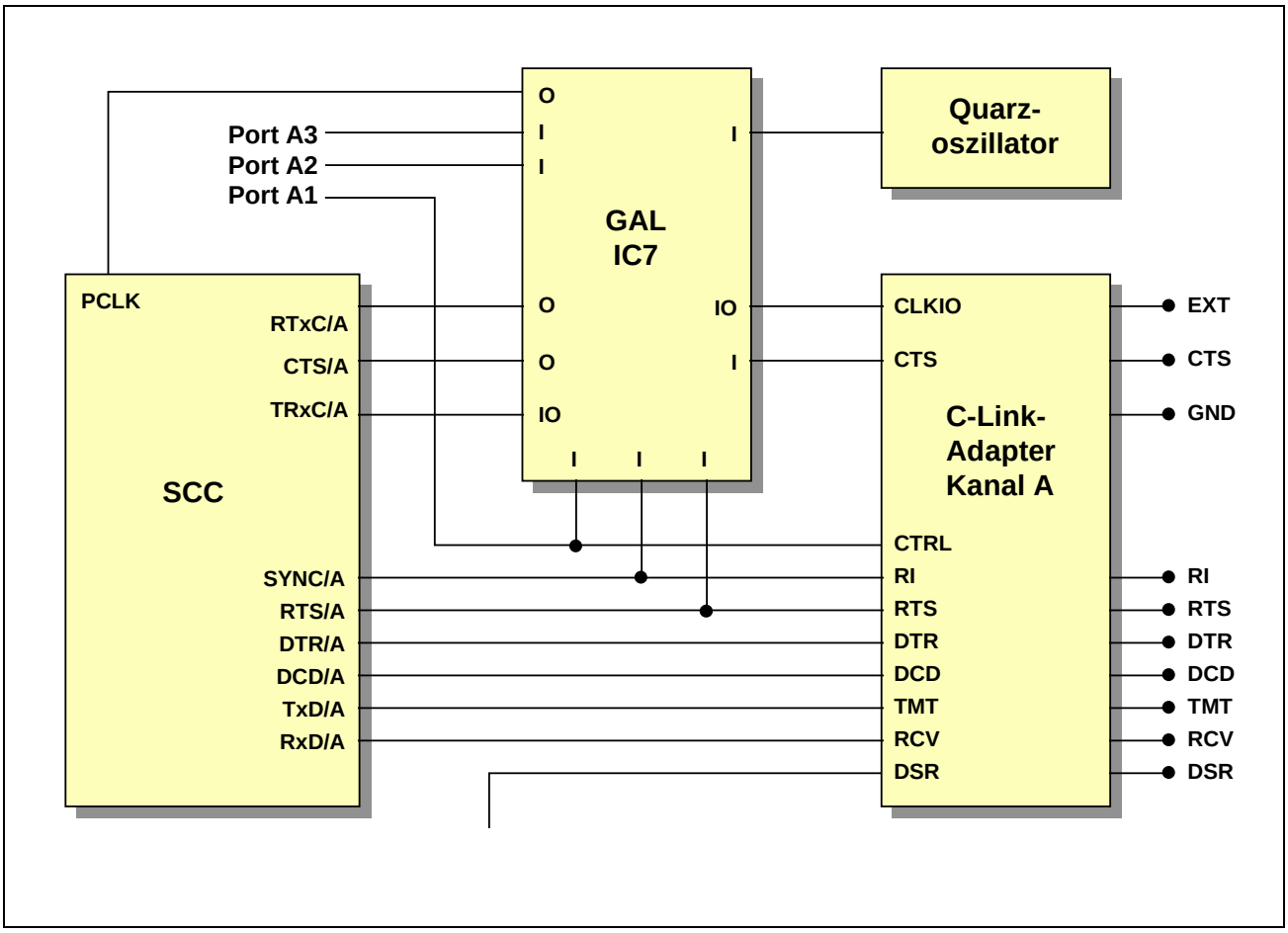


***Abb. 2-1**: Prinzipschaltbild der Konfigurationsmöglichkeiten für Schnittstelle A des Moduls M-COM-2 (Rev. F). Schnittstelle B ist identisch aufgebaut. Quarzoszillator und PCLK des SCC werden von beiden Kanälen genutzt. Vereinfachte Darstellung ohne die Konfigurationsmöglichkeiten im SCC. Die Verbindung von RI des C-Link-Adapters zum GAL IC7 ist erst ab Rev. F des Moduls vorhanden. Sie kann bei Rev. C und E aber auch durch werkseitigen Umbau erreicht werden (siehe EEPROM Wort 4, Bit 11). Die DSR-Leitung des C-Link-Adapters kann keinen Interrupt auslösen, der Status dieser Leitung kann aber gelesen werden (Logik hier nicht gezeigt).*

Anschlußstecker St2

Das Modul M-COM-2 wird über einen 20-poligen Pfostensteckverbinder St2 und ein entsprechendes Flachbandkabel mit der Außenwelt verbunden. Die Belegung von St2 ist so gewählt, daß Pin 1 bis 10 zu Kanal B und Pin 11 bis 20 zu Kanal A gehören. Die Pinbelegung beim jeweiligen Kanal ist abhängig vom aufgesteckten C-Link-Adapter. Sie ist dort angegeben.

In den folgenden Tabellen ist meist auch die Belegung eines fertig konfektionierten Kabels angegeben. Dieses Kabel muß separat bestellt werden (Best.-Nr. K2-2720). Dabei ist an einem Ende des 2 m langen, 20-poligen Flachbandkabels ein 20-poliger Stecker angebracht, der in St2 des M-COM-2 Moduls gesteckt wird. Am anderen Ende ist für jede der beiden Schnittstellen ein 9-poliger D-Submin.-Stecker angebracht (Belegung z. B. bei RS-232 C-Links entsprechend IBM-AT).

Die nachfolgende Beschreibung bezieht sich jeweils auf einen der beiden identisch aufgebauten Kanäle A und B. Bei allen Namen von Signalen und Pins ist der Zusatz /A bzw. /B der Übersichtlichkeit halber weggelassen.

C-Link-Adapter CL232S: RS-232 mit allen Modem-Steuerleitungen

Für einen mit diesem C-Link-Adapter ausgerüsteten Kanal kann Mode 1, 3 oder 5 eingestellt werden. Die damit möglichen Funktionen der RS-232-Anschlüsse CTS, RI und RTS und der SCC-Pins RTxC und TRxC sind in Tabelle 2-1 angegeben.

Tabelle 2-1: Zusammenfassung der möglichen Modes mit CL232S (siehe auch **Abb. 2-2** bis Abb. 2-4). Die mit * gekennzeichneten Signale RTS*, CTS* und Ri* beziehen sich auf die RS-232-Anschlußpins.

Mode	A3 B3	A2 bzw. B2	A1 B1	Funktion von CTS	Funktion von RTS	Funktion von RTxC	TRxC Ein-/ Ausgang	Funktion von TRxC
1	0	0	1	CTS	RTS	Quarz	Eingang	keine
3	0	1	1	an RTxC	RTS	von CTS*	Eingang	keine
5	1	0	1	CTS	von TRxC	Quarz	Ausgang	an RTS*

Tabelle 2-2: Pinbelegung von St2 und Funktion der RS-232-Anschlüsse mit C-Link CL232S in Mode 1, 3 und 5

RS-232 Signal	RS-232 Ein-/Ausgang	Funktion in Mode 1	3	5	Pin (St2): Kanal A	B
DCD	Eingang	DCD	DCD	DCD	11	1
DSR	Eingang	DSR	DSR	DSR	12	2
RCV	Eingang	RxD	RxD	RxD	13	3
RTS	Ausgang	RTS	RTS	CLK_{out}	14	4
TMT	Ausgang	TxD	TxD	TxD	15	5
CTS	Eingang	CTS	CLK_{in}	CTS	16	6
DTR	Ausgang	DTR	DTR	DTR	17	7
Ri	Eingang	Ri	Ri	Ri	18	8
GROUND	Ausgang	GND	GND	GND	19	9
EXT	-	-	-	-	20	10

- = Pin ist nicht angeschlossen

CL232S in Mode 1 (siehe auch Abb. 2-2):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

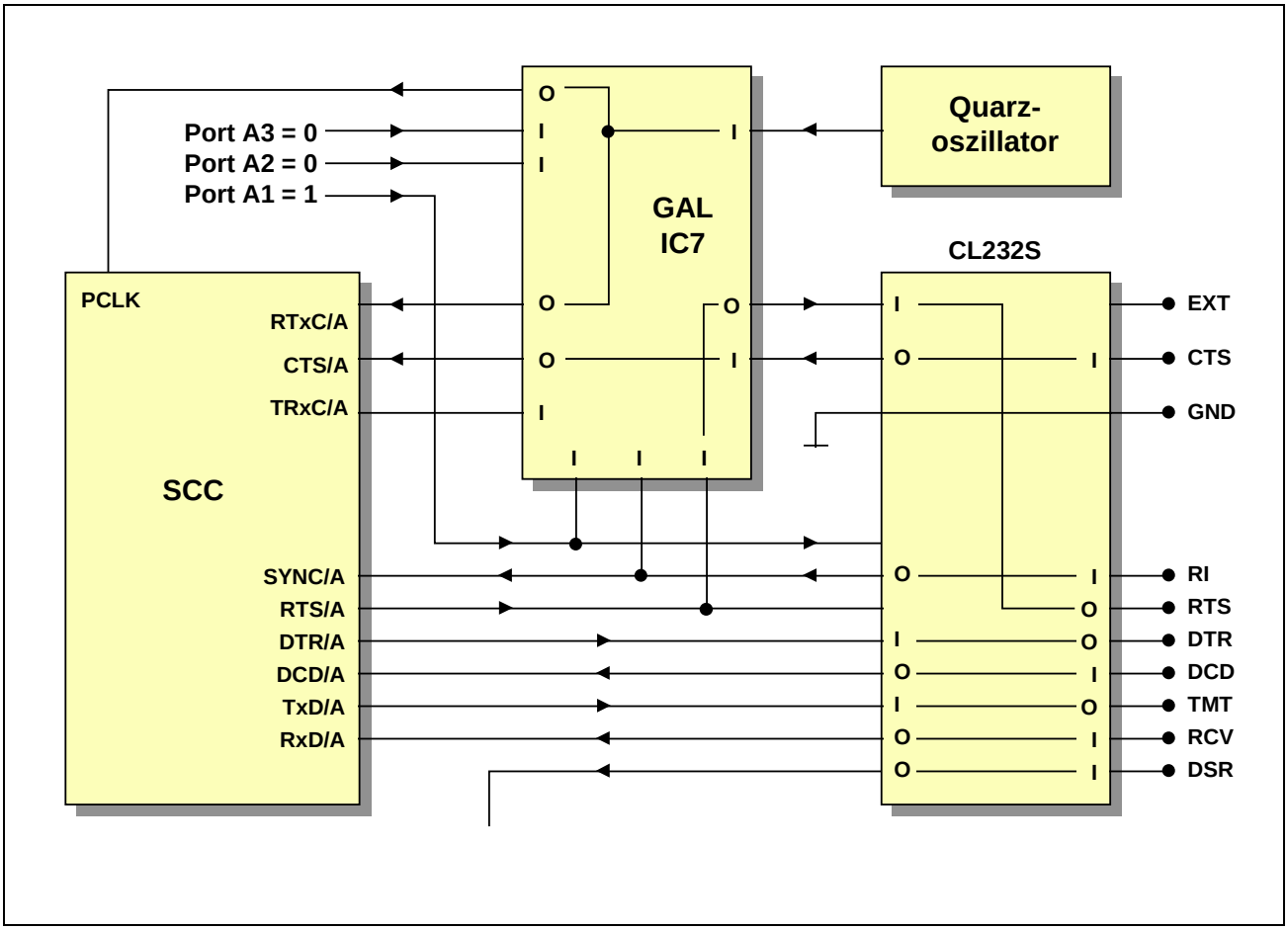


***Abb. 2-2**: C-Link-Adapter CL232S in Mode 1 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL232S in Mode 3 (siehe auch Abb. 2-3):

Der RS-232-Anschluß CTS dient als Clock-Eingang und liegt am RTxC Pin des SCC des zugehörigen Kanals.

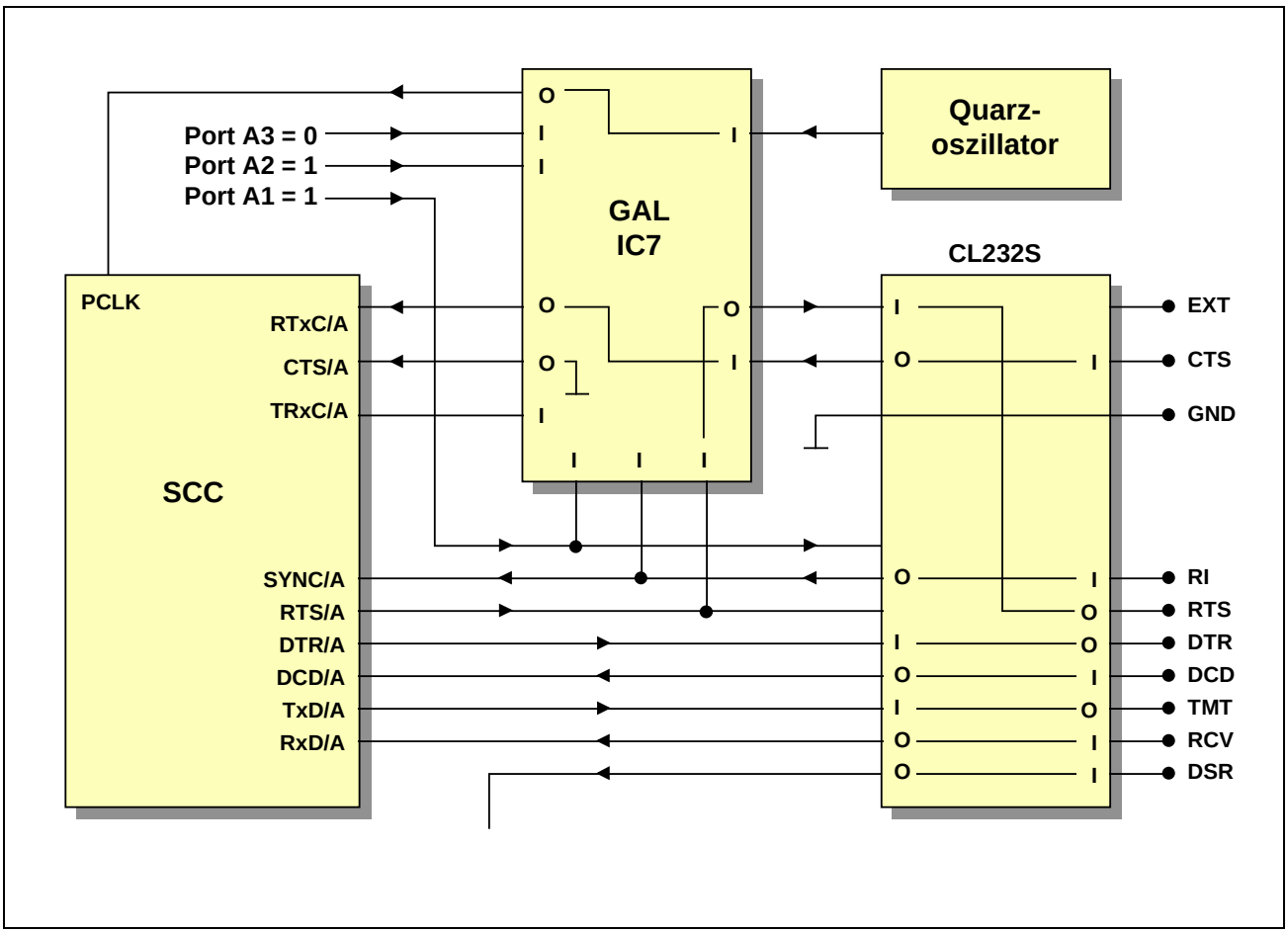


***Abb. 2-3**: C-Link-Adapter CL232S in Mode 3 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL232S in Mode 5 (siehe auch Abb. 2-4):

1. Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

2. Der RS-232-Anschluß RTS dient als Clock-Ausgang vom TRxC Pin des SCC des zugehörigen Kanals. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

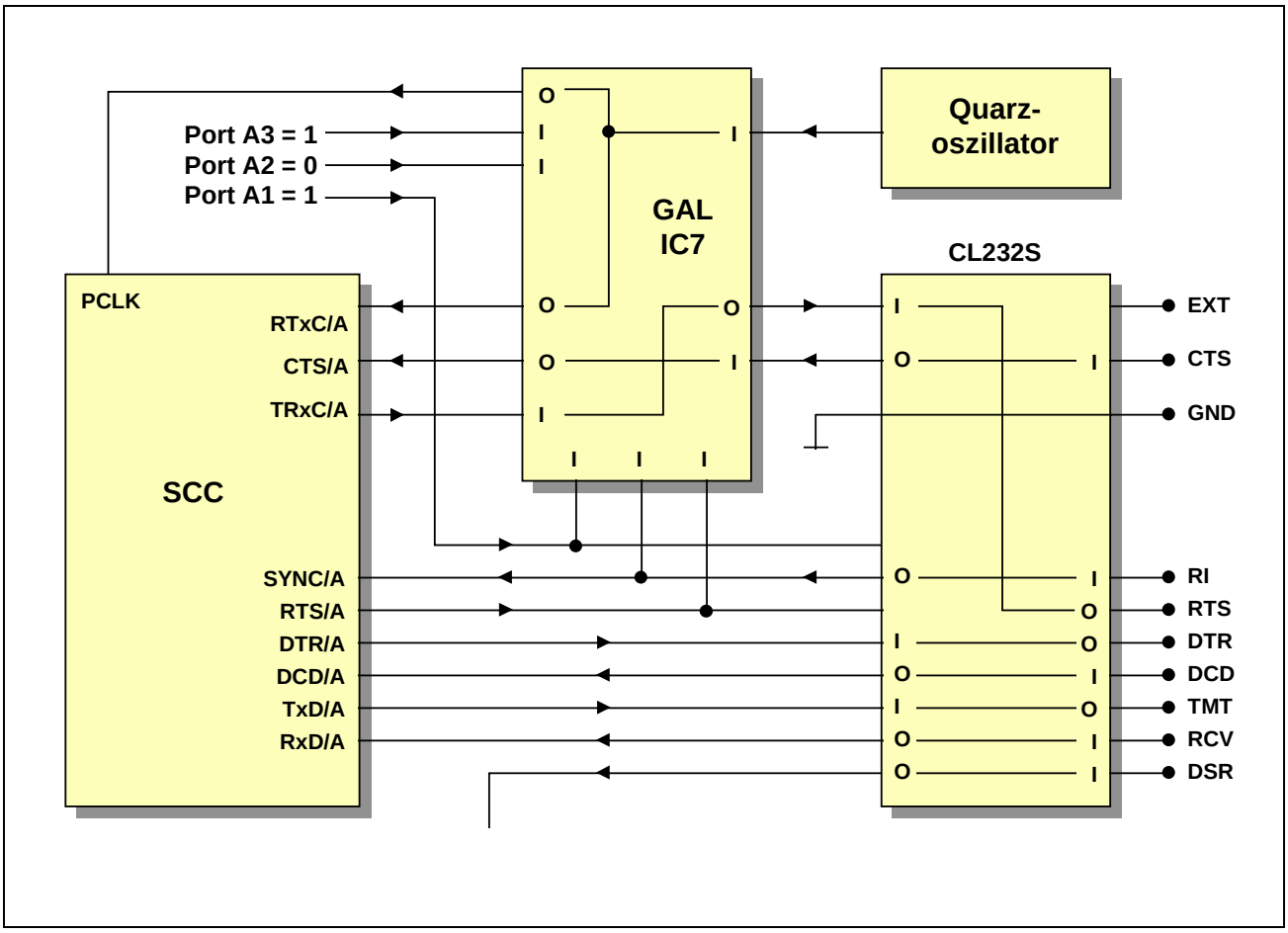


***Abb. 2-4**: C-Link-Adapter CL232S in Mode 5 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

C-Link-Adapter CL232A/i: RS-232 mit zusätzlichem CLK-Input

Für einen mit diesem C-Link-Adapter ausgerüsteten Kanal auf dem Modul M-COM-2 muß Mode 0 (d. h. für Kanal A: Port A3=0, A2=0, A1=0 bzw. für Kanal B: Port B3=0, B2=0, B1=0) oder Mode 4 eingestellt und im SCC der zugehörige TRxC Pin als Eingang konfiguriert werden.

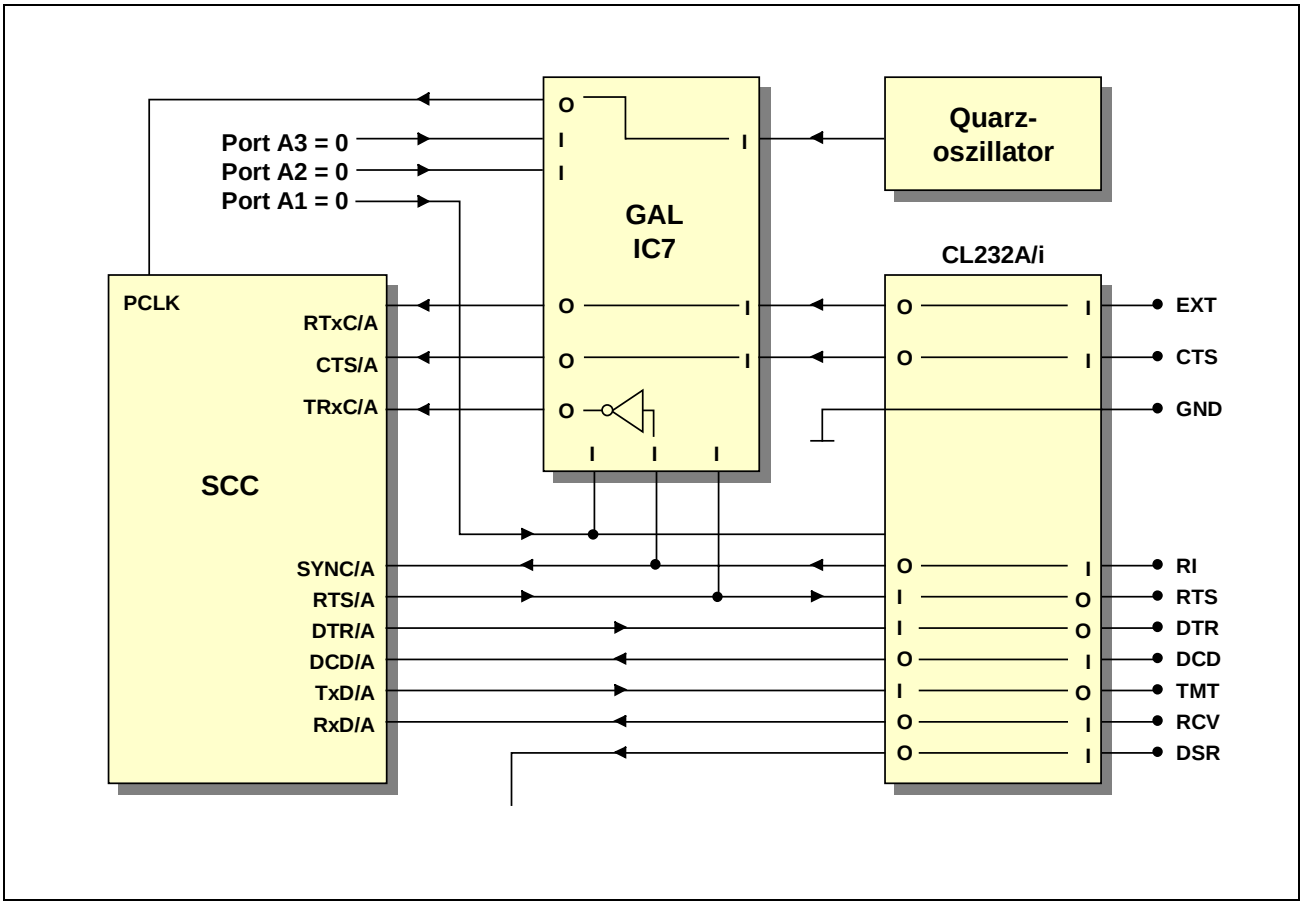


***Abb. 2-5**: Verschaltung bei Mode 0 von Kanal A des Moduls M-COM-2 (Rev. F), ausgerüstet mit einem C-Link-Adapter CL232A/i (Rev. A). Die Angaben gelten entsprechend auch für Kanal B.*

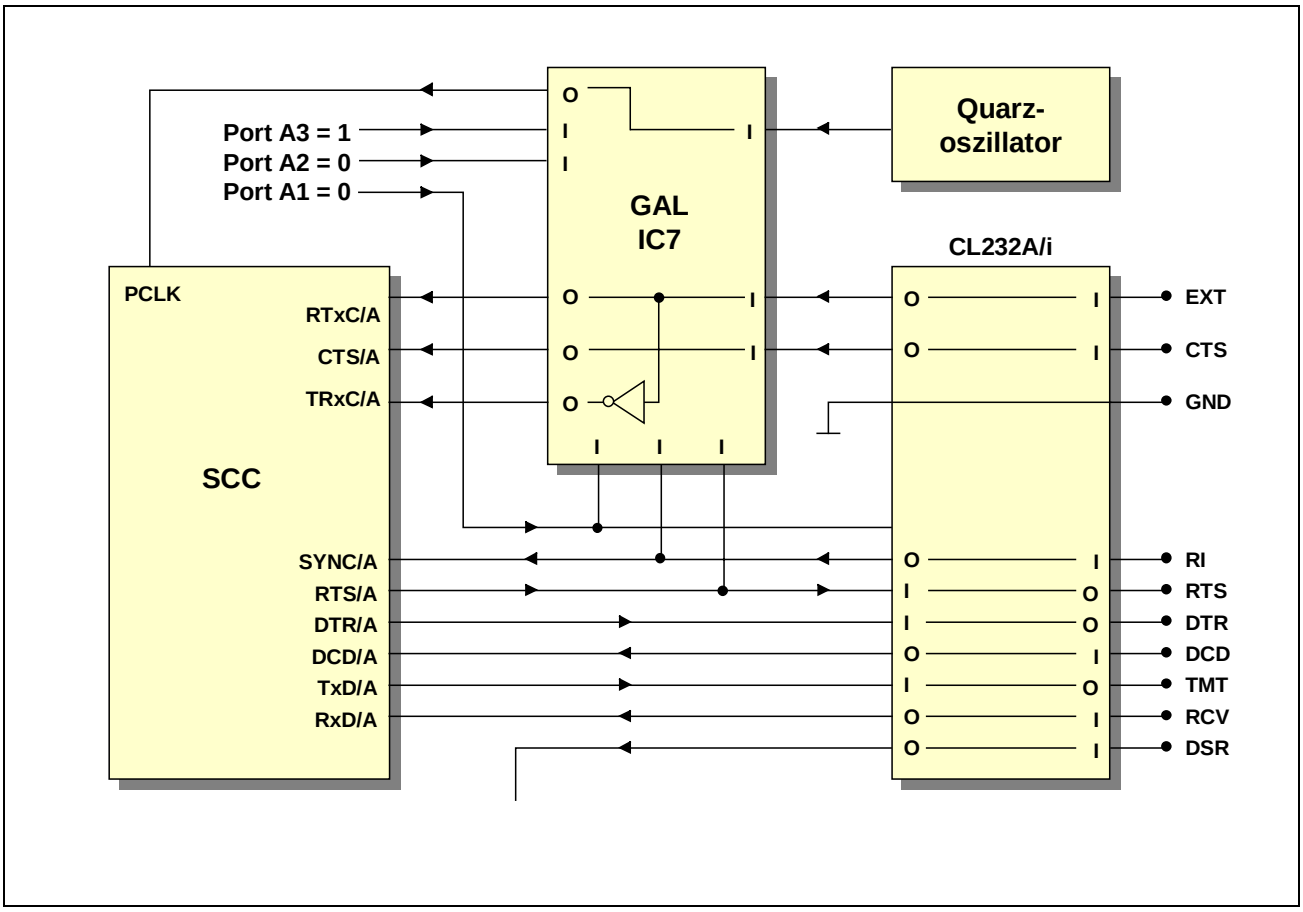


***Abb. 2-6**: Verschaltung bei Mode 4 von Kanal A des Moduls M-COM-2 (Rev. F), ausgerüstet mit einem C-Link-Adapter CL232A/i (Rev. A). Die Angaben gelten entsprechend auch für Kanal B.*

Die RS-232-Schnittstelle weist folgende Besonderheiten auf:

1. Über den RS-232-Anschluß EXT (Eingang) kann ein Takt an Pin RTxC des SCC gelegt werden. Intern im SCC kann dieser dann als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden.
2. Der RS-232-Anschluß RI kann entweder als Modem-Steuerleitung RI dienen (an den zugehörigen SYNC Pin des SCC und damit interruptfähig), oder es kann darüber ein weiterer Takt von außen an Pin TRxC des SCC gelegt werden. Intern im SCC kann dieser dann als Empfangs- und/oder Sendetakt konfiguriert werden.

***Tabelle 2-3**:* St2 Pinbelegung mit C-Link CL232A/i

RS-232 Signal	**RS-232 Ein-/Ausgang**	**Funktion**	**Pin (St2) Kanal A**	**Kanal B**
DCD	Eingang	DCD	11	1
DSR	Eingang	DSR	12	2
RxD	Eingang	RxD	13	3
RTS	Ausgang	RTS	14	4
TxD	Ausgang	TxD	15	5
CTS	Eingang	CTS	16	6
DTR	Ausgang	DTR	17	7
Ri	Eingang	Ri oder CLK_{in} (an SYNC und in Mode 0 zusätzlich invertiert an TRxC *)	18	8
GROUND	Ausgang	Ground	19	9
EXT	Eingang	CLK_{in} (an RTxC und in Mode 4 zusätzlich invertiert an TRxC)	20	10

* := erst ab Modul Rev. F oder bei Rev. C und E mit Umbau

C-Link-Adapter CL232A/o: RS-232 mit zusätzlichem CLK-Output

Für einen mit diesem C-Link-Adapter ausgerüsteten Kanal auf dem Modul M-COM-2 sollte Mode 5 (für Kanal A: Port A3=1, A2=0, A1=1 bzw. für Kanal B: Port B3=1, B2=0, B1=1) eingestellt und im SCC der zugehörige TRxC Pin als Ausgang konfiguriert werden.

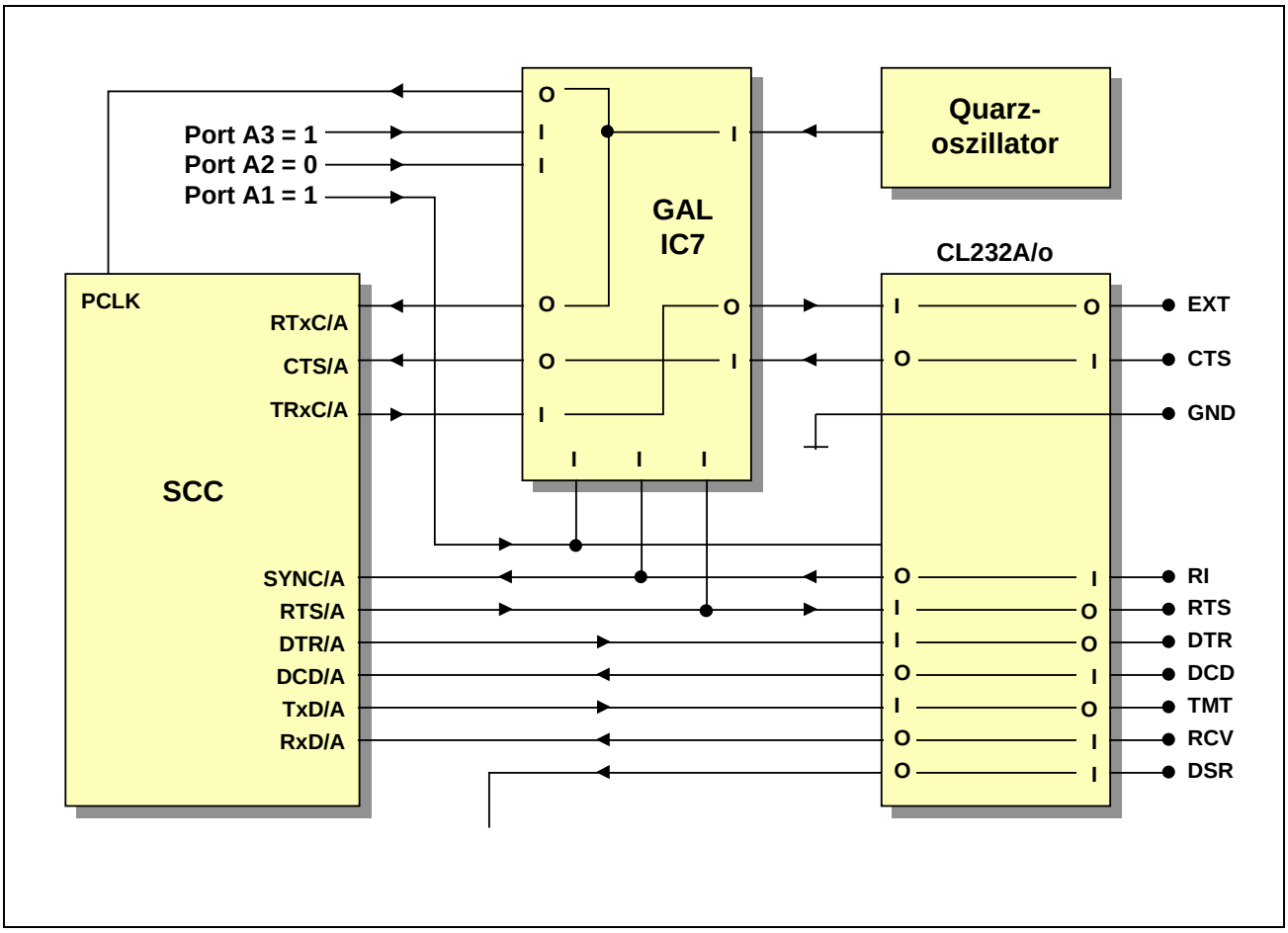


***Abb. 2-7**: Verschaltung bei Mode 5 von Kanal A des Moduls M-COM-2 (Rev. F), ausgerüstet mit einem C-Link-Adapter CL232A/o (Rev. A). Die Angaben gelten entsprechend auch für Kanal B.*

Die RS-232-Schnittstelle weist folgende Besonderheiten auf:

1. Über den RS-232-Anschluß EXT (Ausgang) kann dann das Signal am TRxC Pin dieses Kanals des SCC nach außen geliefert werden. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

2. Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann so bei asynchronem Betrieb die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

***Tabelle 2-4**:* St2 Pinbelegung mit C-Link CL232A/o

RS-232 Signal	**RS-232 Ein-/Ausgang**	**Funktion**	**Pin (St2) Kanal A**	**Kanal B**
DCD	Eingang	DCD	11	1
DSR	Eingang	DSR	12	2
RxD	Eingang	RxD	13	3
RTS	Ausgang	RTS	14	4
TxD	Ausgang	TxD	15	5
CTS	Eingang	CTS	16	6
DTR	Ausgang	DTR	17	7
Ri	Eingang	Ri (an SYNC des SCC)	18	8
GROUND	Ausgang	Ground	19	9
EXT	Ausgang	CLK_{out} (von TRxC des SCC)	20	10

C-Link-Adapter CL232i: galvanisch getrennte RS-232

Für einen mit diesem C-Link-Adapter ausgerüsteten Kanal kann Mode 1, 3 oder 5 eingestellt werden. Die damit möglichen Funktionen der RS-232-Anschlüsse CTS und RTS und der SCC-Pins RTxC und TRxC sind in Tabelle 2-5 angegeben.

***Tabelle 2-5**:* Zusammenfassung der möglichen Modes mit CL232i (siehe auch Abb. 2-7, Abb. 2-8 und Abb. 2-9). Die mit * gekennzeichneten Signale RTS* und CTS* beziehen sich auf die RS-232-Anschlußpins.

Mode	A3 B3	A2 bzw. B2	A1 B1	Funktion von CTS*	Funktion von RTS*	Funktion von RTxC	TRxC Ein-/ Aus-gang	Funktion von TRxC
1	0	0	1	CTS	RTS	Quarz	Eingang	keine
3	0	1	1	an RTxC	RTS	von CTS*	Eingang	keine
5	1	0	1	CTS	von TRxC	Quarz	Ausgang	an RTS*

***Tabelle 2-6**:* Pinbelegung von St2 und Funktion der RS-232-Anschlüsse mit C-Link CL232i in Mode 1, 3 und 5

RS-232 Signal	RS-232 Ein-/Ausgang	Funktion in Mode 1	3	5	Pin (St2) Kanal A	Kanal B
-	-	-	-	-	11	1
-	-	-	-	-	12	2
RxD	Eingang	RxD	RxD	RxD	13	3
RTS	Ausgang	RTS	RTS	CLK_{out}	14	4
TxD	Ausgang	TxD	TxD	TxD	15	5
CTS	Eingang	CTS	CLK_{in}	CTS	16	6
-	-	-	-	-	17	7
-	-	-	-	-	18	8
GROUND	Ausgang	GND	GND	GND	19	9
-	-	-	-	-	20	10

- = Pin ist nicht angeschlossen

CL232i in Mode 1 (siehe auch Abb. 2-8):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

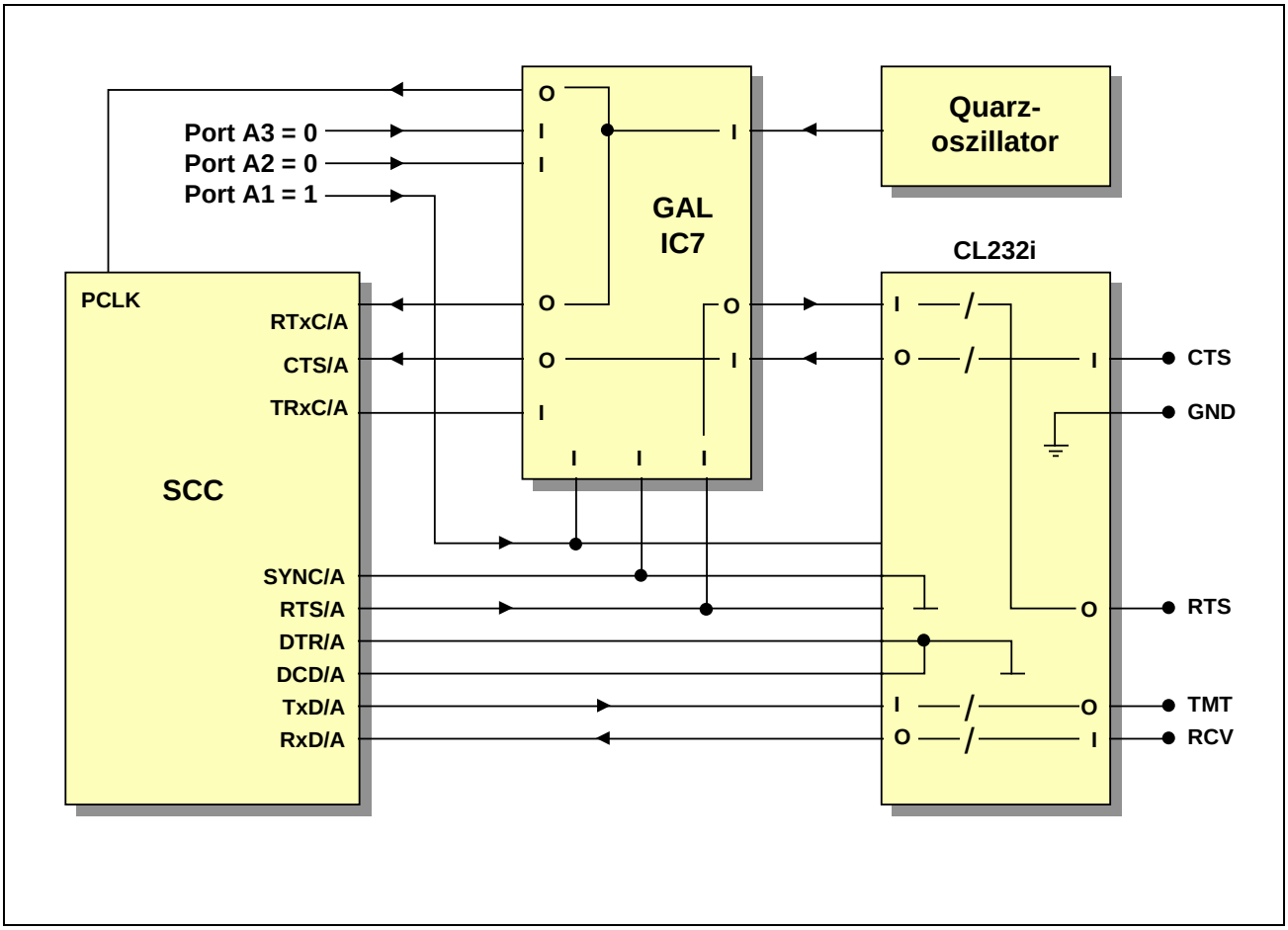


***Abb. 2-8**: C-Link-Adapter CL232i in Mode 1 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL232i in Mode 3 (siehe auch Abb. 2-9):

Der RS-232-Anschluß CTS dient als Clock-Eingang und liegt am RTxC Pin des SCC des zugehörigen Kanals.

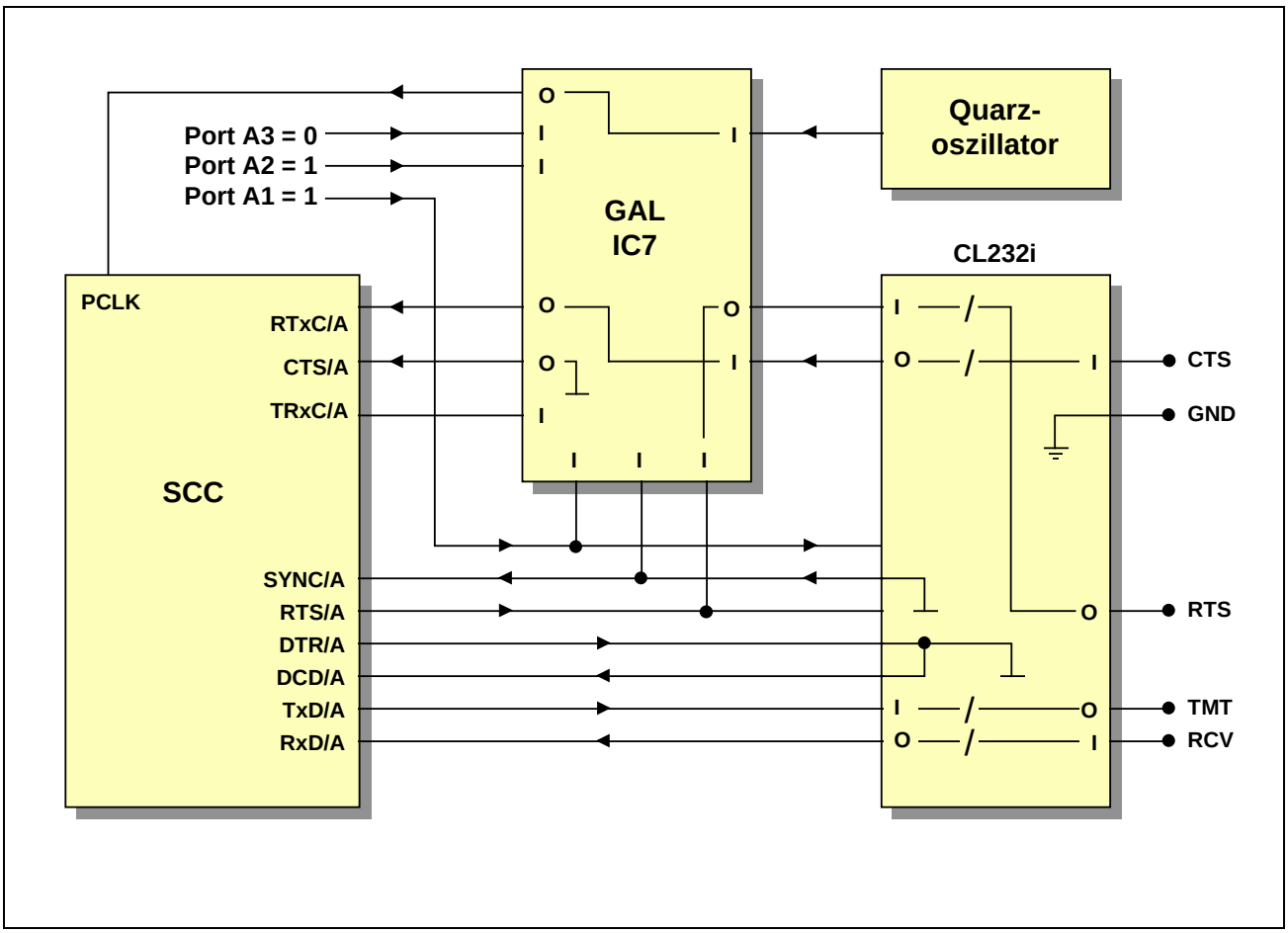


Abb. 2-9: C-Link-Adapter CL232i in Mode 3 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.

CL232i in Mode 5 (siehe auch Abb. 2-10):

1. Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

2. Der RS-232-Anschluß RTS dient als Clock-Ausgang vom TRxC Pin des SCC des zugehörigen Kanals. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

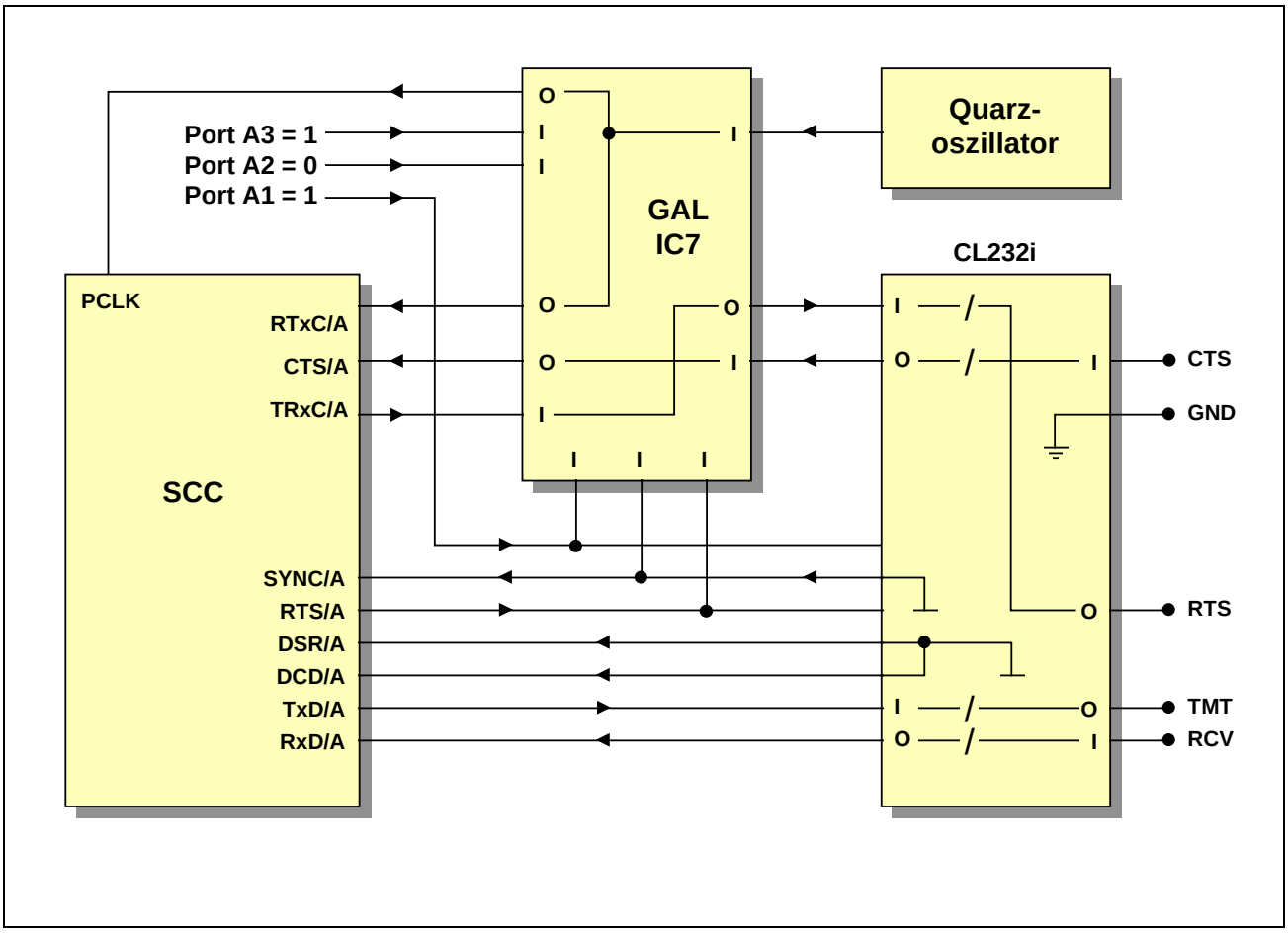


Abb. 2-10: C-Link-Adapter CL232i in Mode 5 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.

C-Link-Adapter CL422S: RS-422-Schnittstelle

Verfügbar sind die Modem-Steuersignale RTS und CTS (Mode 1). Per Software kann die CTS-Leitung auch als Takteingang (Mode 3) oder die RTS-Leitung als Taktausgang (Mode 5) geschaltet werden. Die damit möglichen Funktionen der RS-422-Anschlüsse CTS und RTS und der SCC-Pins RTxC und TRxC sind in Tabelle 2-7 angegeben. Der DTR-Pin des zugehörigen Kanals muß im SCC auf log. 1 gesetzt werden. Damit wird der RS-422-Sendetreiber enabled. Beachten Sie bitte, daß bei RS-422-Verbindungen + mit + und - mit - der Gegenstation verbunden wird.

Tabelle 2-7: Zusammenfassung der möglichen Modes mit CL422S und der Funktionen der RS-422-Leitungen (die mit * gekennzeichneten Signale CTS* und RTS* beziehen sich auf die RS-422-Anschlußpins).

Mode	**A3 B3**	**A2 bzw. B2**	**A1 B1**	**Funktion von CTS***	**Funktion von RTS***	**Funktion von RTxC**	**TRxC Ein-/ Ausgang**	**Funktion von TRxC**
1	0	0	1	CTS	RTS	Quarz	Eingang	keine
3	0	1	1	an RTxC	RTS	von CTS*	Eingang	keine
5	1	0	1	CTS	von TRxC	Quarz	Ausgang	an RTS*

Abschlußwiderstände

Das C-Link enthält zwei 120 Ω Abschlußwiderstände, einen zwischen RCV+ und RCV- und einen zwischen CTS+ und CTS-. Beide können optional per Lötbrücke (auf der Unterseite des C-Link) eingeschaltet werden. Wenn Lötbrücke B1 geschlossen ist, ist der Abschlußwiderstand zwischen RCV+ und RCV- eingeschaltet. Wenn Lötbrücke B2 geschlossen ist, ist der Abschlußwiderstand zwischen CTS+ und CTS- eingeschaltet. Eine offene Leitung (RCV oder CTS) wird durch interne Pull-Up- bzw. Pull-Down-Widerstände auf log. 1 gezogen.

Tabelle 2-8: Pinbelegung von St2 und Funktion der RS-422-Anschlüsse mit C-Link CL422S in Mode 1, 3 und 5 (Angaben für D-Submin. Stecker gelten für SORCUS Kabel K2-2720).

RS-422 Signal	**RS-422 Ein-/ Ausgang**	**Funktion in Mode 1**	**3**	**5**	**Pin (St2) Kanal A**	**B**	**Pin D-Submin. 9-pol.**
RTS-	Ausgang	RTS-	RTS-	CLK_{out}-	11	1	1
RTS+	Ausgang	RTS+	RTS+	CLK_{out}+	12	2	6
CTS-	Eingang	CTS-	CLK_{in}-	CTS-	13	3	2
CTS+	Eingang	CTS+	CLK_{in}+	CTS+	14	4	7
TMT-	Ausgang	TMT-	TMT-	TMT-	15	5	3
TMT+	Ausgang	TMT+	TMT+	TMT+	16	6	8
RCV-	Eingang	RCV-	RCV-	RCV-	17	7	4
RCV+	Eingang	RCV+	RCV+	RCV+	18	8	9
GROUND	Ausgang	GND	GND	GND	19	9	5
-	-	-	-	-	20	10	-

- = Pin ist nicht angeschlossen

CL422S für RS-422 in Mode 1 (siehe auch Abb. 2-11):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

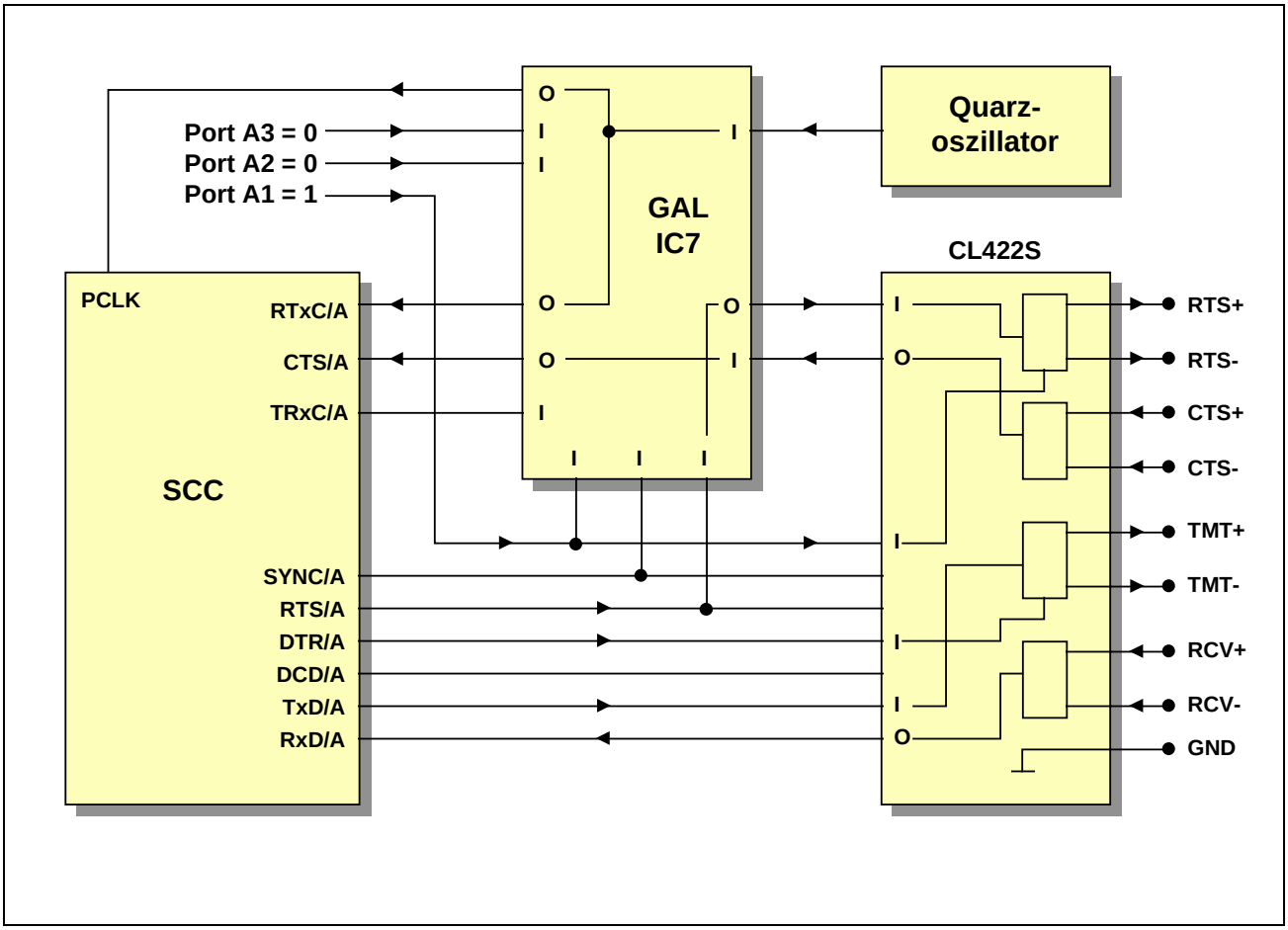


***Abb. 2-11**: C-Link-Adapter CL422S für RS-422 in Mode 1 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL422S für RS-422 in Mode 3 (siehe auch Abb. 2-12):

Der RS-422-Anschluß CTS dient als Clock-Eingang und liegt am RTxC Pin des SCC des zugehörigen Kanals. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden.

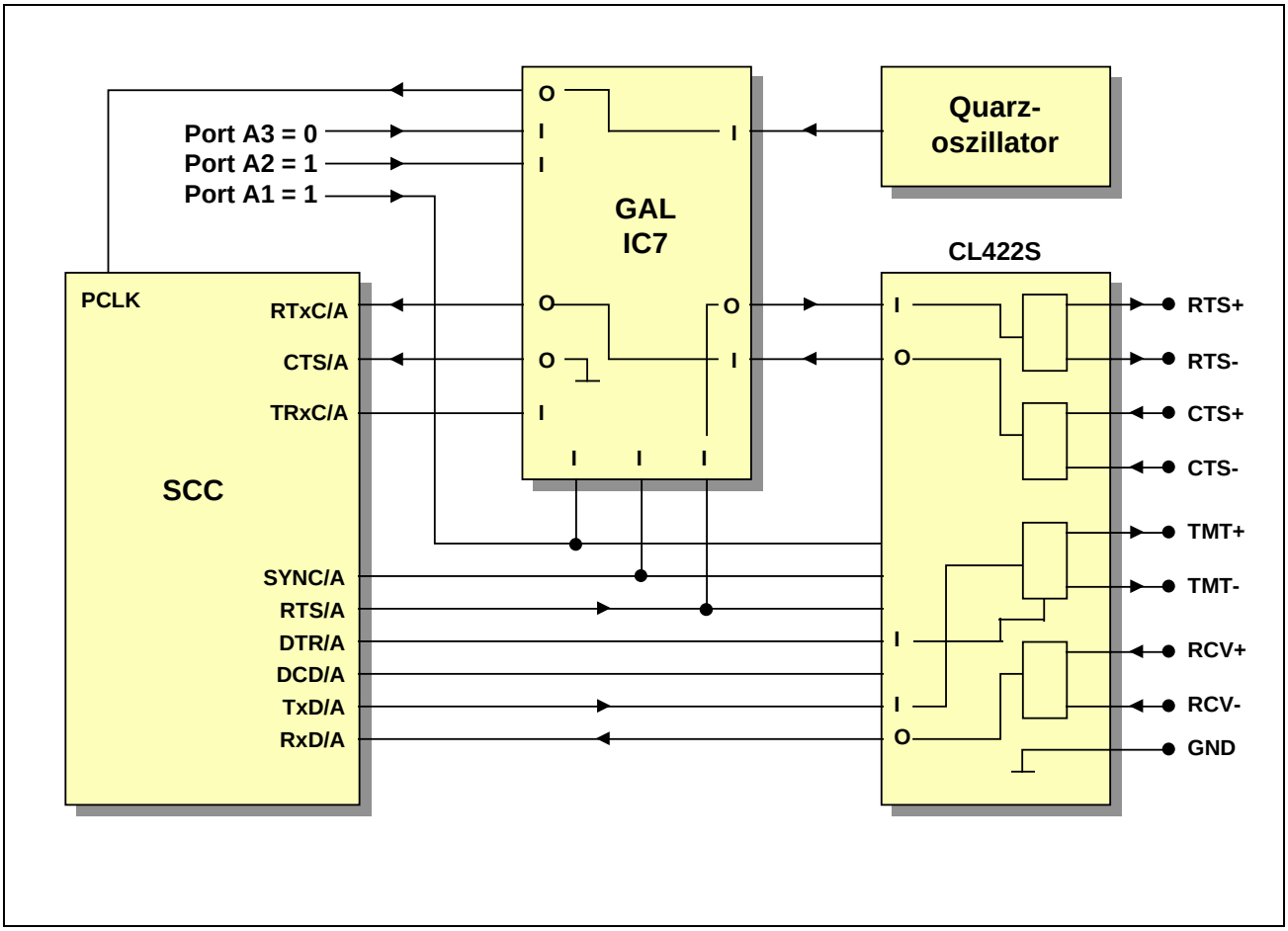


Abb. 2-12: C-Link-Adapter CL422S für RS-422 in Mode 3 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.

CL422S für RS-422 in Mode 5 (siehe auch Abb. 2-13):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

1. Der RS-422-Anschluß RTS dient als Clock-Ausgang vom TRxC Pin des SCC des zugehörigen Kanals. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

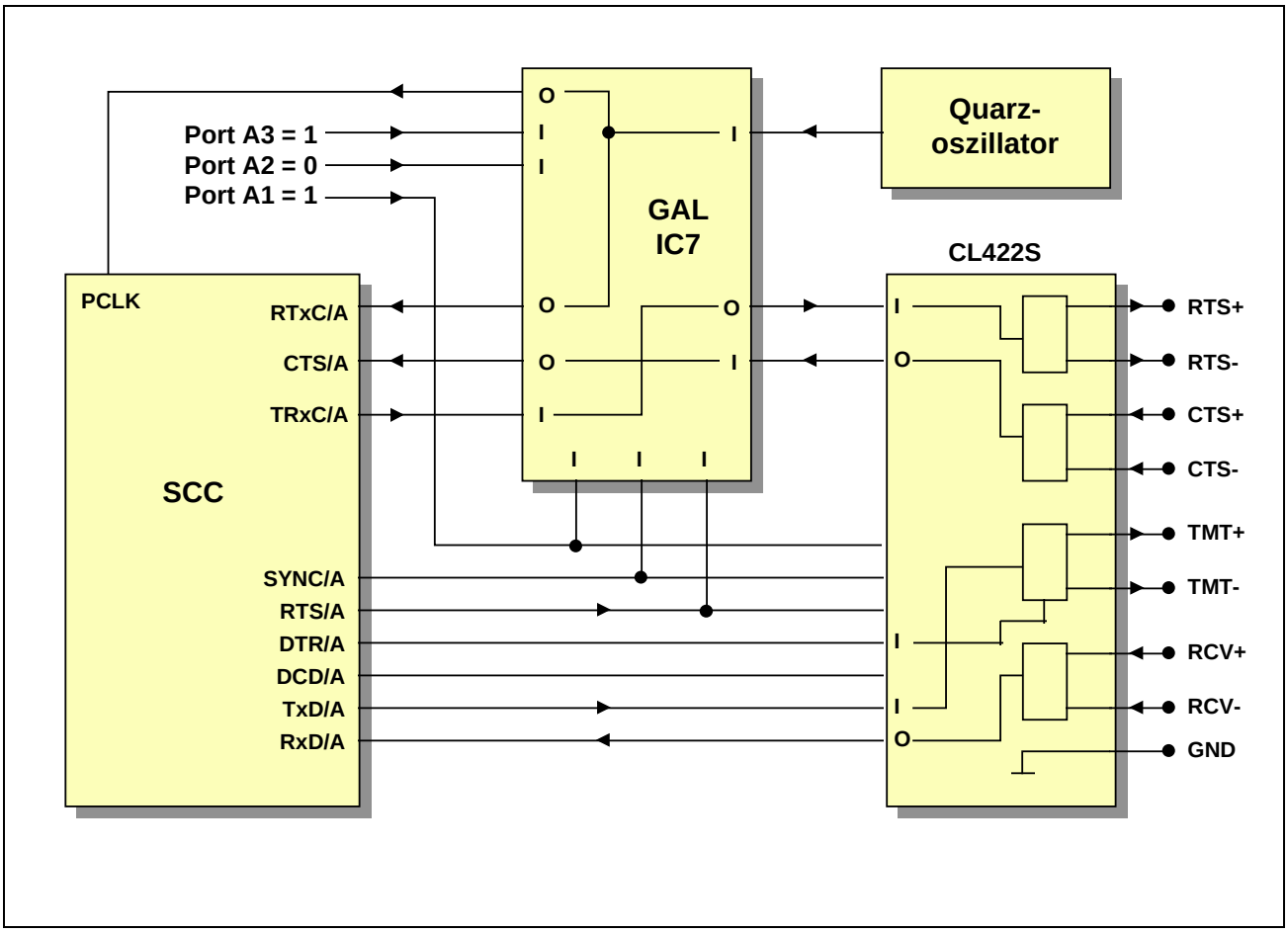


***Abb. 2-13**: C-Link-Adapter CL422S für RS-422 in Mode 5 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

C-Link-Adapter CL422S: RS-485-Schnittstelle

Für RS-485 muß RCV+ an TMT+ (= DATA+) und RCV- an TMT- (= DATA-) und beide Leitungen DATA+ und DATA- dann mit DATA+ und DATA- der Gegenstation verbunden werden.

Die Umschaltung zwischen Senden und Empfangen geschieht per Software über den Pin DTR des SCC. Da er im SCC noch für andere Zwecke programmiert werden kann, muß er als DTR (Bit 2 in WR14 des SCC = 0) programmiert werden. Nach Reset ist der DTR-Pin = 0, damit ist der RS-485-Sendetreiber abgeschaltet. Der Zustand des DTR-Pin wird über Bit 7 in WR5 des SCC gesteuert (Bit 7 = 0: Empfangen, Bit 7 = 1: Senden). Der RS-485-Empfänger ist immer aktiviert, so daß auch ein von der Schnittstelle selbst gesendetes Signal wieder am Empfangseingang des SCC erscheint. Wenn das nicht gewünscht ist, muß der Empfänger im SCC disabled werden (Bit 0 in WR3 des SCC).

Zusätzlich verfügbar sind die Modem-Steuersignale RTS und CTS (Mode 1). Per Software kann die CTS-Leitung auch als Takteingang (Mode 3) oder die RTS-Leitung als Taktausgang (Mode 5) geschaltet werden. Die damit möglichen Funktionen der RS-485-Anschlüsse CTS und RTS und der SCC-Pins RTxC und TRxC sind in Tabelle 2-9 angegeben.

***Tabelle 2-9**:* Zusammenfassung der möglichen Modes mit CL422S und der Funktionen der RS-485-Leitungen. RTS* und CTS* sind die RS-485-Leitungen, RTS und CTS die Signale (Pins) des SCC. Mode 0 ist mit angegeben, weil dieser Mode nach einem Hardware-Reset der Basiskarte, z. B. auch nach Power-On, eingeschaltet ist.

	A3	**A2 bzw.**	**A1**	**Funktion der RS-485-Leitungen**		**Funktion von**	**TRxC Ein-/**	**Funktion von**
Mode	**B3**	**B2**	**B1**	**CTS***	**RTS***	**RTxC**	**Ausgang**	**TRxC**
0	0	0	0	CTS	disabled	keine	Eingang	keine
1	0	0	1	CTS	RTS	Quarz	Eingang	keine
2	0	1	0	an RTxC	disabled	von CTS*	Eingang	keine
3	0	1	1	an RTxC	RTS	von CTS*	Eingang	keine
5	1	0	1	CTS	von TRxC	Quarz	Ausgang	an RTS*

Abschlußwiderstände

Das C-Link enthält zwei 120 Ω Abschlußwiderstände, einen zwischen RCV+ und RCV- und einen zwischen CTS+ und CTS-. Beide können optional per Lötbrücke (auf der Unterseite des C-Link) eingeschaltet werden. Wenn Lötbrücke B1 geschlossen ist, ist der Abschlußwiderstand zwischen RCV+ und RCV- eingeschaltet. Wenn Lötbrücke B2 geschlossen ist, ist der Abschlußwiderstand zwischen CTS+ und CTS- eingeschaltet. Eine offene Leitung (RCV oder CTS) wird durch interne Pull-Up- bzw. Pull-Down-Widerstände auf log. 1 gezogen.

Tabelle 2-10: Pinbelegung von St2 und Funktion der RS-485-Anschlüsse mit C-Link CL422S in Mode 1, 3 und 5 (Angaben für D-Submin. Stecker gelten für SORCUS Kabel K2-2720).

RS-485	**RS-485 Ein-/**	**Funktion in Mode**					**Pin (St2) Kanal**		**Pin D-Sub-min.**	
Signal	**Aus-gang**	**0**	**1**	**2**	**3**	**5**	**A**	**B**	**9-pol.**	
RTS-	ts	-	RTS-	-	RTS-	CLK-	11	1	1	
RTS+	ts	-	RTS+	-	RTS+	CLK+	12	2	6	
CTS-	Eingang	CTS-	CTS-	CLK-	CLK-	CTS-	13	3	2	
CTS+	Eingang	CTS+	CTS+	CLK+	CLK+	CTS+	14	4	7	
DATA- (TMT-)	Ausgang	TMT-	TMT-	TMT-	TMT-	TMT-	15	5	3	*
DATA+ (TMT+)	Ausgang	TMT+	TMT+	TMT+	TMT+	TMT+	16	6	8	*
(RCV-)	Eingang	RCV-	RCV-	RCV-	RCV-	RCV-	17	7	4	*
(RCV+)	Eingang	RCV+	RCV+	RCV+	RCV+	RCV+	18	8	9	*
GROUND	Ausgang	GND	GND	GND	GND	GND	19	9	5	
-	-	-	-	-	-	-	20	10	-	

ts = Tri-State
- = Pin ist nicht angeschlossen bzw. Funktion = keine
* = Lötverbindungen am D-Submin.-Stecker

Hinweise zur Programmierung (RS-485 mit C-Link CL422S):

In Mode 1 ist mit GAL "G032x07A" (x steht für die Revision des Moduls) der Pin RTxC des SCC nicht verwendet, mit GAL "G032x07B" oder GAL "G032x07C" liegt an diesem Pin der Takt des Quarzoszillators und kann intern z. B. als Eingangstakt für die DPLL oder als Sende- und/oder Empfangstakt verwendet werden.

CL422S für RS-485 in Mode 1 (siehe auch Abb. 2-14):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

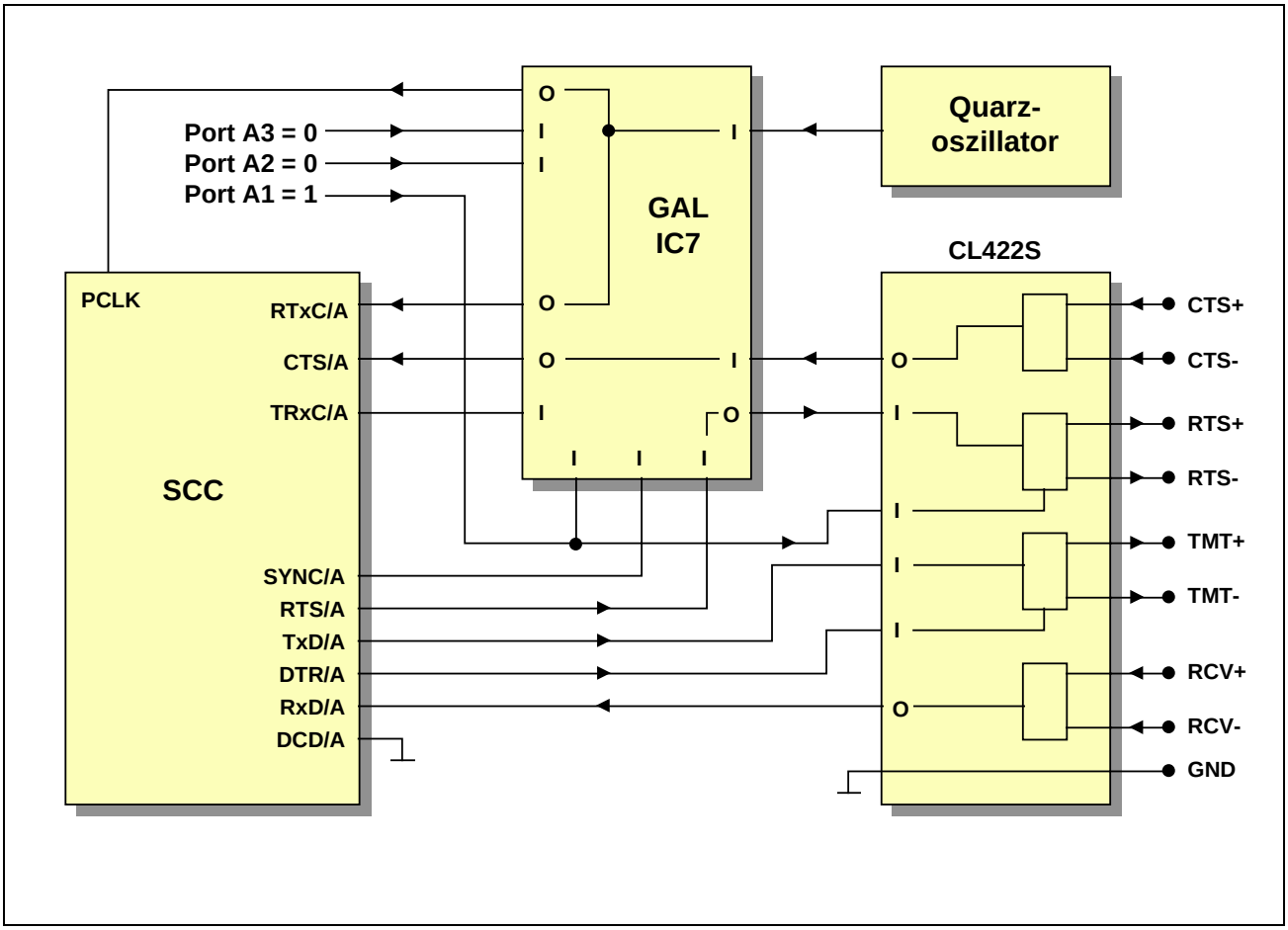


***Abb. 2-14**: C-Link-Adapter CL422S für RS-485 in Mode 1 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL422S für RS-485 in Mode 2 und 3 (siehe auch Abb. 2-15):

Beide Modes unterscheiden sich nur bezüglich der Funktion der RS-485-Anschlüsse RTS+ und RTS-. In Mode 2 sind sie auf Tri-State geschaltet (= hochohmig), in Mode 3 sind sie als RTS-Ausgänge aktiv. Die RS-485-Anschlüsse CTS+ und CTS- können als Clock-Eingang dienen, das Signal liegt am RTxC Pin des SCC des zugehörigen Kanals.

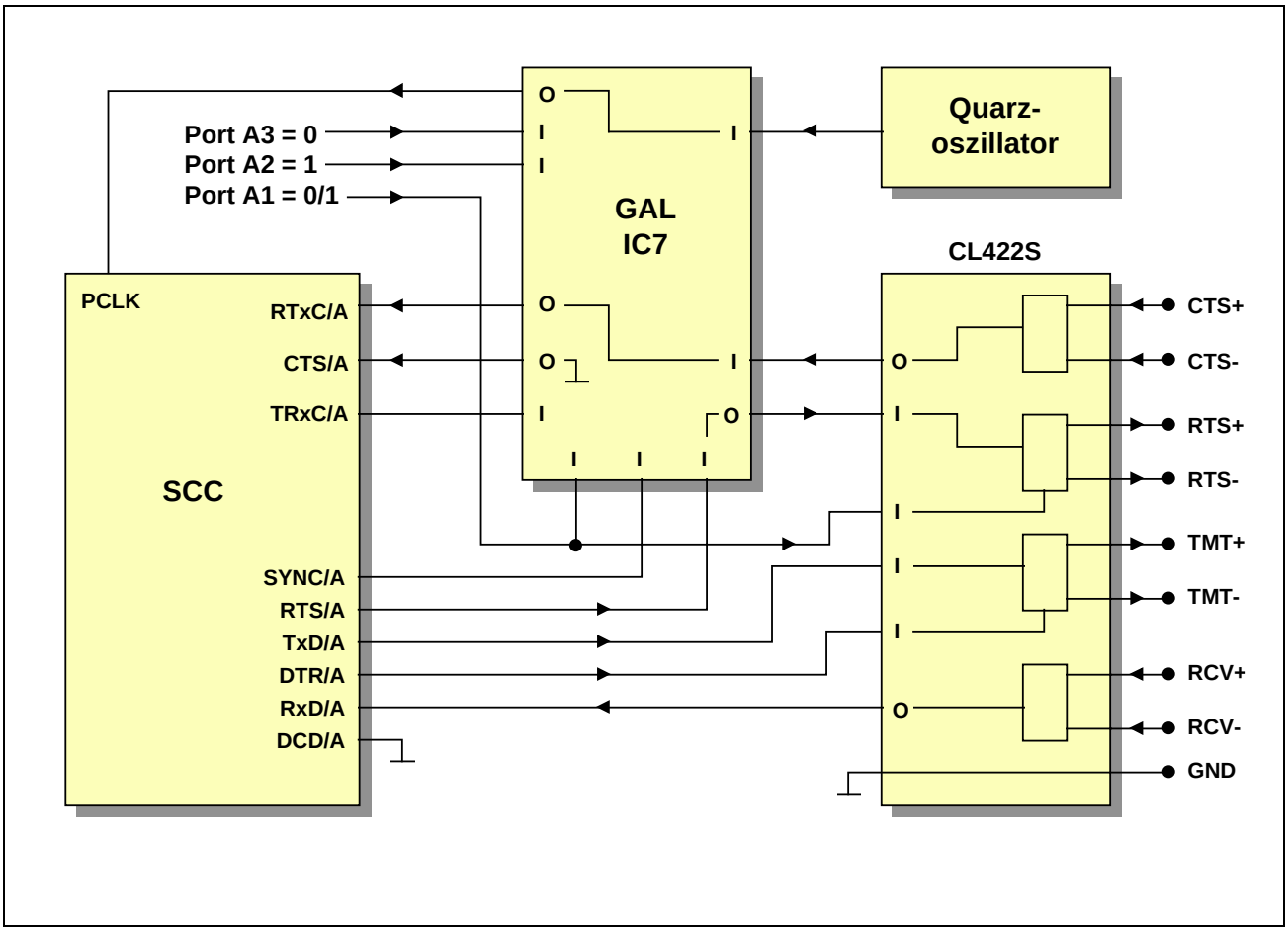


***Abb. 2-15**: C-Link-Adapter CL422S für RS-485 in Mode 2 und 3 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL422S für RS-485 in Mode 5 (siehe auch Abb. 2-16):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

Die RS-485-Anschlüsse RTS+ und RTS- dienen als Clock-Ausgang vom TRxC Pin des SCC des zugehörigen Kanals. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

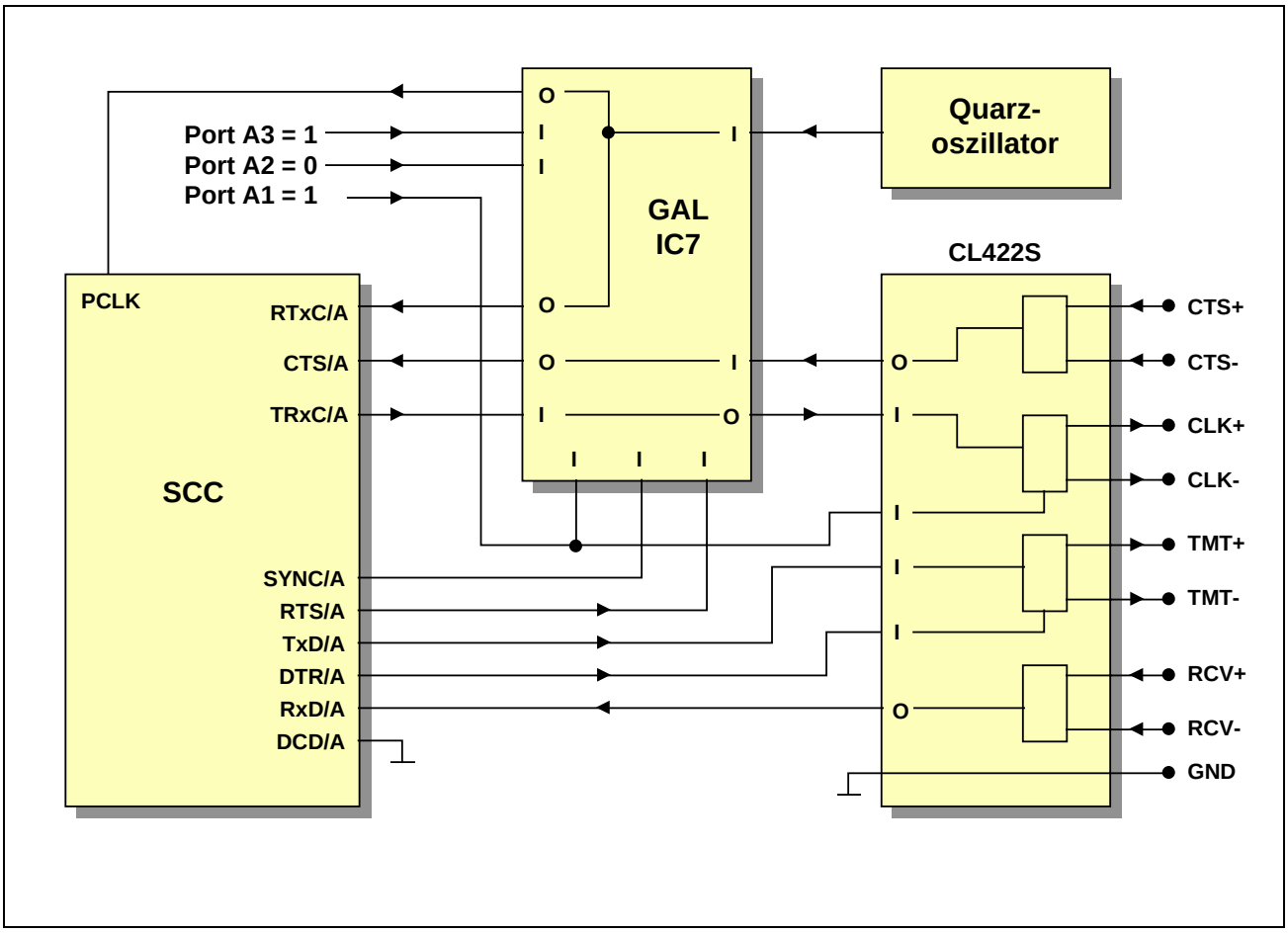


***Abb. 2-16**: C-Link-Adapter CL422S für RS-485 in Mode 5 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

C-Link-Adapter CL422i: galv. getrennte RS-422-Schnittstelle

Verfügbar sind die Modem-Steuersignale RTS und CTS (Mode 1). Per Software kann die CTS-Leitung auch als Takteingang (Mode 3) oder die RTS-Leitung als Taktausgang (Mode 5) geschaltet werden. Die damit möglichen Funktionen der RS-422-Anschlüsse CTS und RTS und der SCC-Pins RTxC und TRxC sind in Tabelle 2-11 angegeben. Beachten Sie bitte, daß bei RS-422-Verbindungen + mit + und - mit - der Gegenstation verbunden wird.

Tabelle 2-11: Zusammenfassung der möglichen Modes mit CL422i und der Funktionen der RS-422-Leitungen (die mit * gekennzeichneten Signale CTS* und RTS* beziehen sich auf die RS-422-Anschlußpins).

Mode	**A3 B3**	**A2 bzw. B2**	**A1 B1**	**Funktion von CTS***	**Funktion von RTS***	**Funktion von RTxC**	**TRxC Ein-/ Ausgang**	**Funktion von TRxC**
1	0	0	1	CTS	RTS	Quarz	Eingang	keine
3	0	1	1	an RTxC	RTS	von CTS*	Eingang	keine
5	1	0	1	CTS	von TRxC	Quarz	Ausgang	an RTS*

Abschlußwiderstände

Das C-Link enthält keine Abschlußwiderstände.

***Tabelle 2-12**:* Pinbelegung von St2 und Funktion der RS-422-Anschlüsse mit C-Link CL422i in Mode 1, 3 und 5 (Angaben für D-Submin. Stecker gelten für SORCUS Kabel K2-2720).

RS-422 Signal	**RS-422 Ein-/ Ausgang**	**Funktion in Mode 1**	**3**	**5**	**Pin (St2) Kanal A**	**B**	**Pin D-Submin. 9-pol.**
RTS-	Ausgang	RTS-	RTS-	CLK_{out}-	11	1	1
RTS+	Ausgang	RTS+	RTS+	CLK_{out}+	12	2	6
CTS-	Eingang	CTS-	CLK_{in}-	CTS-	13	3	2
CTS+	Eingang	CTS+	CLK_{in}+	CTS+	14	4	7
TMT-	Ausgang	TMT-	TMT-	TMT-	15	5	3
TMT+	Ausgang	TMT+	TMT+	TMT+	16	6	8
RCV-	Eingang	RCV-	RCV-	RCV-	17	7	4
RCV+	Eingang	RCV+	RCV+	RCV+	18	8	9
GROUND	Ausgang	GND	GND	GND	19	9	5
-	-	-	-	-	20	10	-

- = Pin ist nicht angeschlossen

CL422i für RS-422 in Mode 1 (siehe auch Abb. 2-17):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

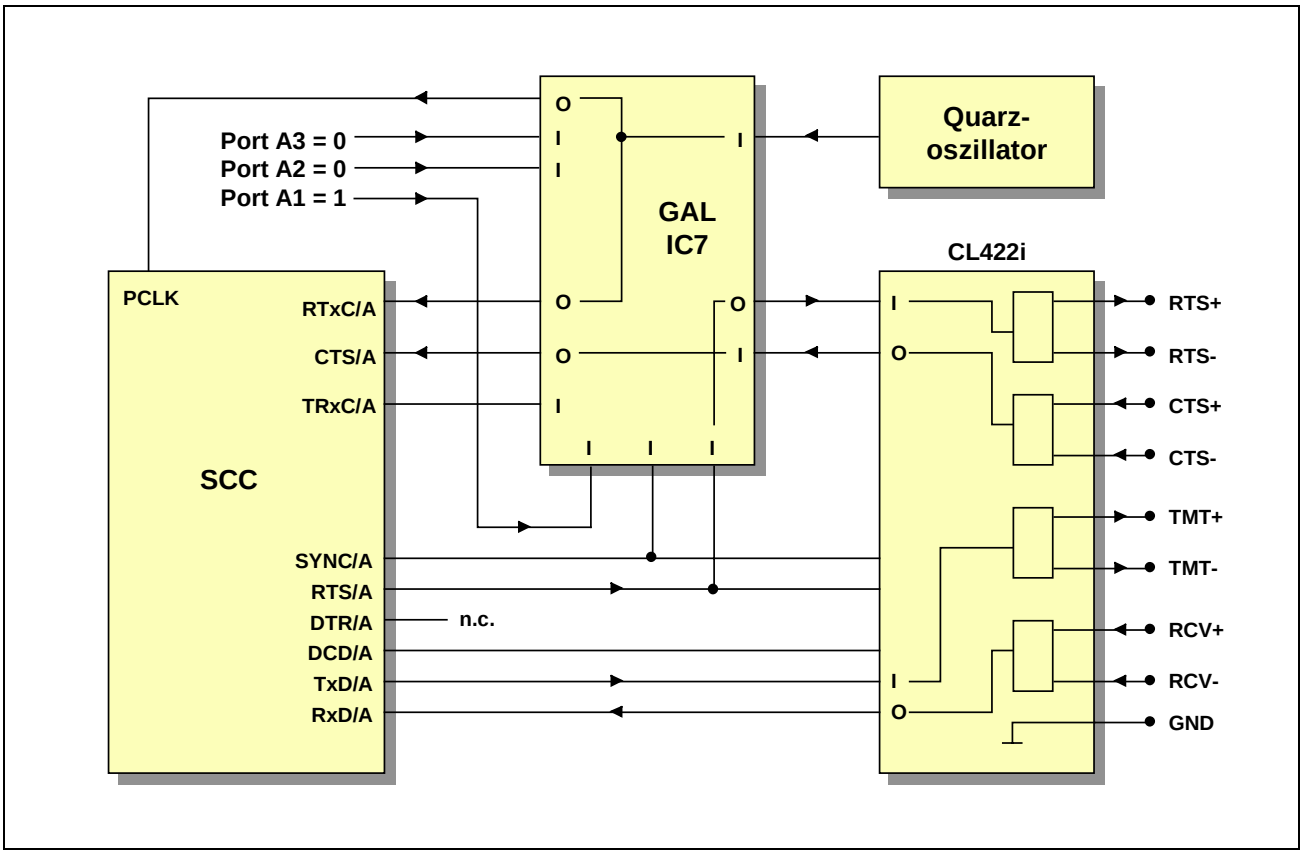


***Abb. 2-17**: C-Link-Adapter CL422i für RS-422 in Mode 1 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL422i für RS-422 in Mode 3 (siehe auch Abb. 2-18):

Der RS-422-Anschluß CTS dient als Clock-Eingang und liegt am RTxC Pin des SCC des zugehörigen Kanals. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden.

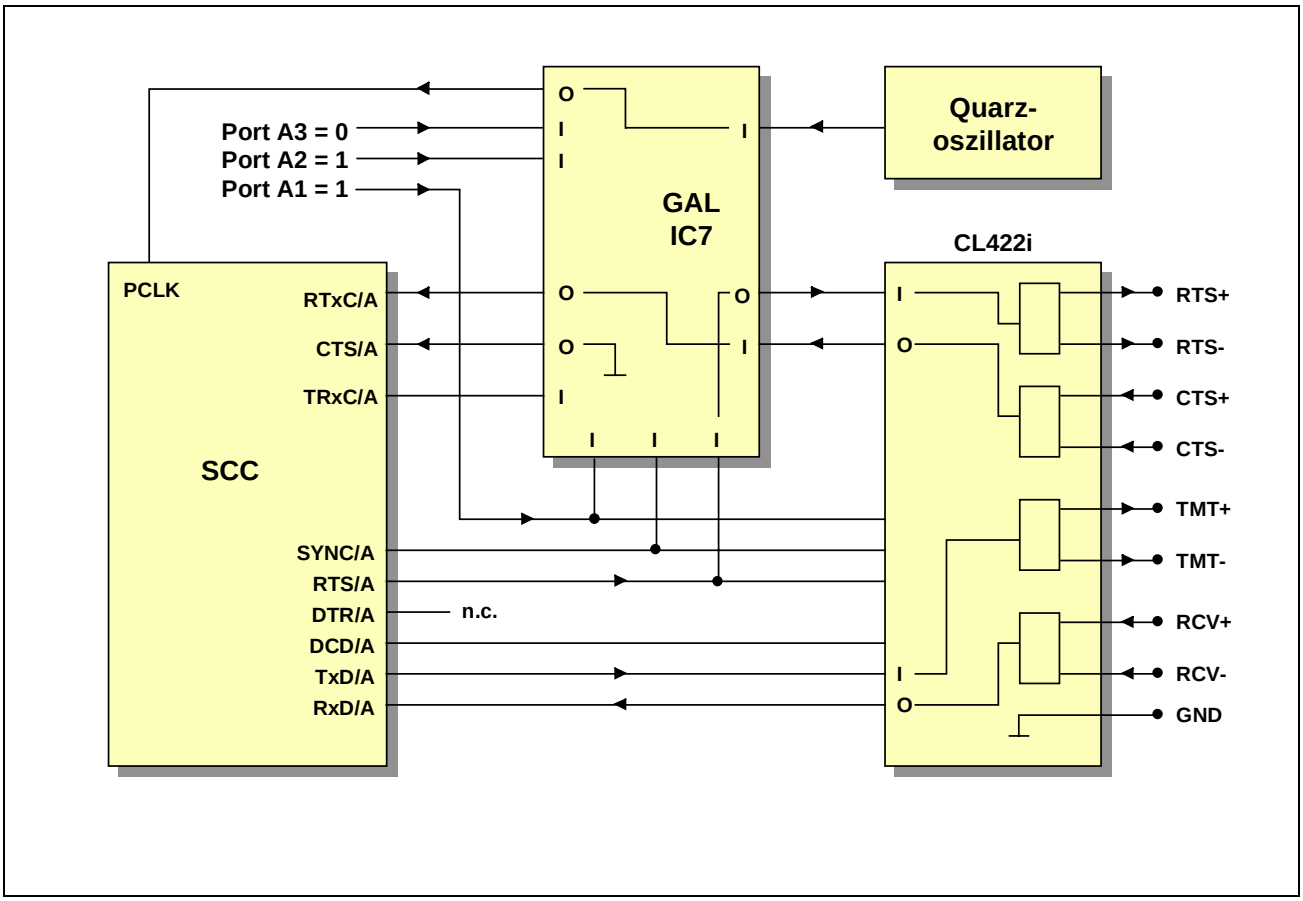


***Abb. 2-18**: C-Link-Adapter CL422i für RS-422 in Mode 3 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

CL422i für RS-422 in Mode 5 (siehe auch Abb. 2-19):

Der Takt des Quarzoszillators liegt auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann bei asynchronem Betrieb so die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

Der RS-422-Anschluß RTS dient als Clock-Ausgang vom TRxC Pin des SCC des zugehörigen Kanals. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

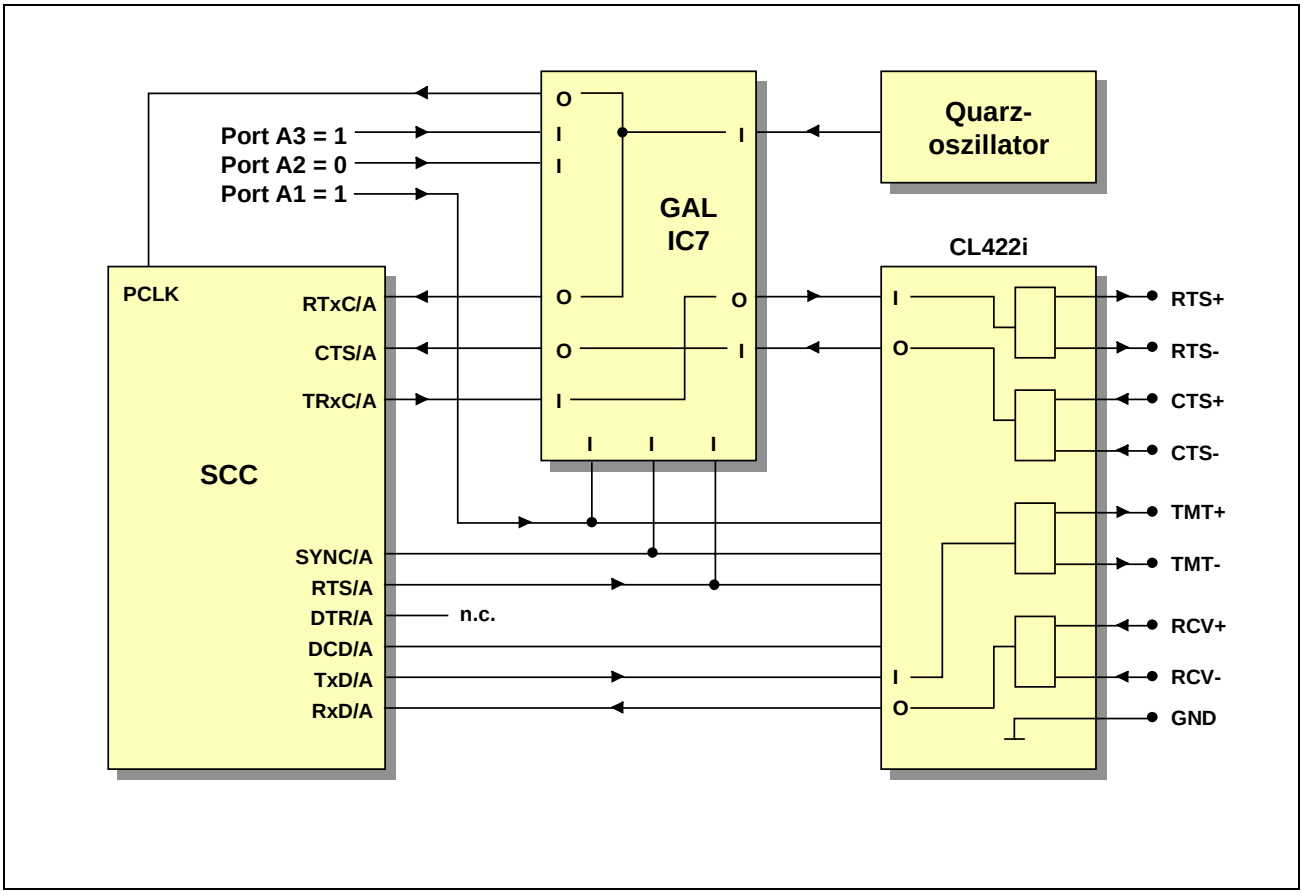


***Abb. 2-19**: C-Link-Adapter CL422i für RS-422 in Mode 5 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

C-Link-Adapter CL485i/P: RS-485-Schnittstelle

Dieser C-Link-Adapter stellt eine galvanisch getrennte RS-485-Schnittstelle für max. Baudraten bis 20 MBaud zur Verfügung und ist damit auch für Profibus bis 12 MBaud geeignet.

Die Umschaltung zwischen Senden und Empfangen geschieht per Software über Port A1 bzw. B1.

Die zusätzliche Leitung DPRTS auf der galvanisch getrennten RS-485-Seite zeigt an, ob die RS-485-Schnittstelle Sender (DPRTS = log. 1) oder Empfänger (DPRTS = log. 0) ist. Diese Leitung wird von einem TTL-Buffer über einen Serienwiderstand von 300 Ω getrieben. Die Datenleitungen DPA und DPB sind überspannungsgeschützt und mit Pull-Up- (an DPB) bzw. Pull-Down-Widerständen (an DPA) von je 47 kΩ versehen. Der RS-485-Empfänger ist immer aktiv, beim Senden erscheint also das gesendete Signal wieder am Empfängereingang des SCC. Beachten Sie bitte, daß bei RS-485-Verbindungen + mit + und - mit - der Gegenstation verbunden wird.

***Tabelle 2-13**:* Zusammenfassung der möglichen Modes mit CL485i/P. Nach einem Reset des Systems ist immer Mode 0 eingestellt. Als Initialisierung sollte zunächst Mode = 2 eingestellt werden (Empfangen) und danach der Pin TRxC des SCC als Ausgang geschaltet werden. TRxC hat hier bei diesem C-Link-Adapter keine Funktion.

Mode	A3 B3	A2 bzw. B2	A1 B1	Pins des SCC CTS	DCD	SYNC	Funktion von RTxC	TRxC Ein-/ Ausgang	DPRTS (RS-485)	Funktion von RS-485
0	0	0	0	= 0	= 0	= 1	keine	Eingang	= 0	Empfang
2	0	1	0	= 0	= 0	= 1	= 0	Ausgang	= 0	Empfang
3	0	1	1	= 0	= 0	= 1	= 0	Ausgang	= 1	Senden

Abschlußwiderstände

Das C-Link enthält keine Abschlußwiderstände.

***Tabelle 2-14**:* Pinbelegung von St2 und Funktion der RS-485-Anschlüsse mit C-Link CL485i/P in Mode 0, 2 und 3 (Angaben für D-Submin. Stecker gelten für SORCUS Kabel K2-2720).

RS-485 Signal	**RS-485 Ein-/Ausgang**	**Funktion in Mode 0**	**2**	**3**	**Pin (St2) Kanal A**	**B**	**Pin D-Submin. 9-pol.**
DPPE	Ausgang	GND	GND	GND	11	1	1
DP5V	Ausgang	+5 V	+5 V	+5 V	12	2	6
-	-	-	-	-	13	3	2
-	-	-	-	-	14	4	7
DPB	Ein-/Aus	-In	-In	-Out	15	5	3
DPA	Ein-/Aus	+In	+In	+Out	16	6	8
DPRTS	Ausgang	= 0	= 0	= 1	17	7	4
-	-	-	-	-	18	8	9
DPGND	Ausgang	GND	GND	GND	19	9	5
-	-	-	-	-	20	10	-

- = Pin ist nicht angeschlossen

CL485i/P für RS-485 in Mode 2 und 3:

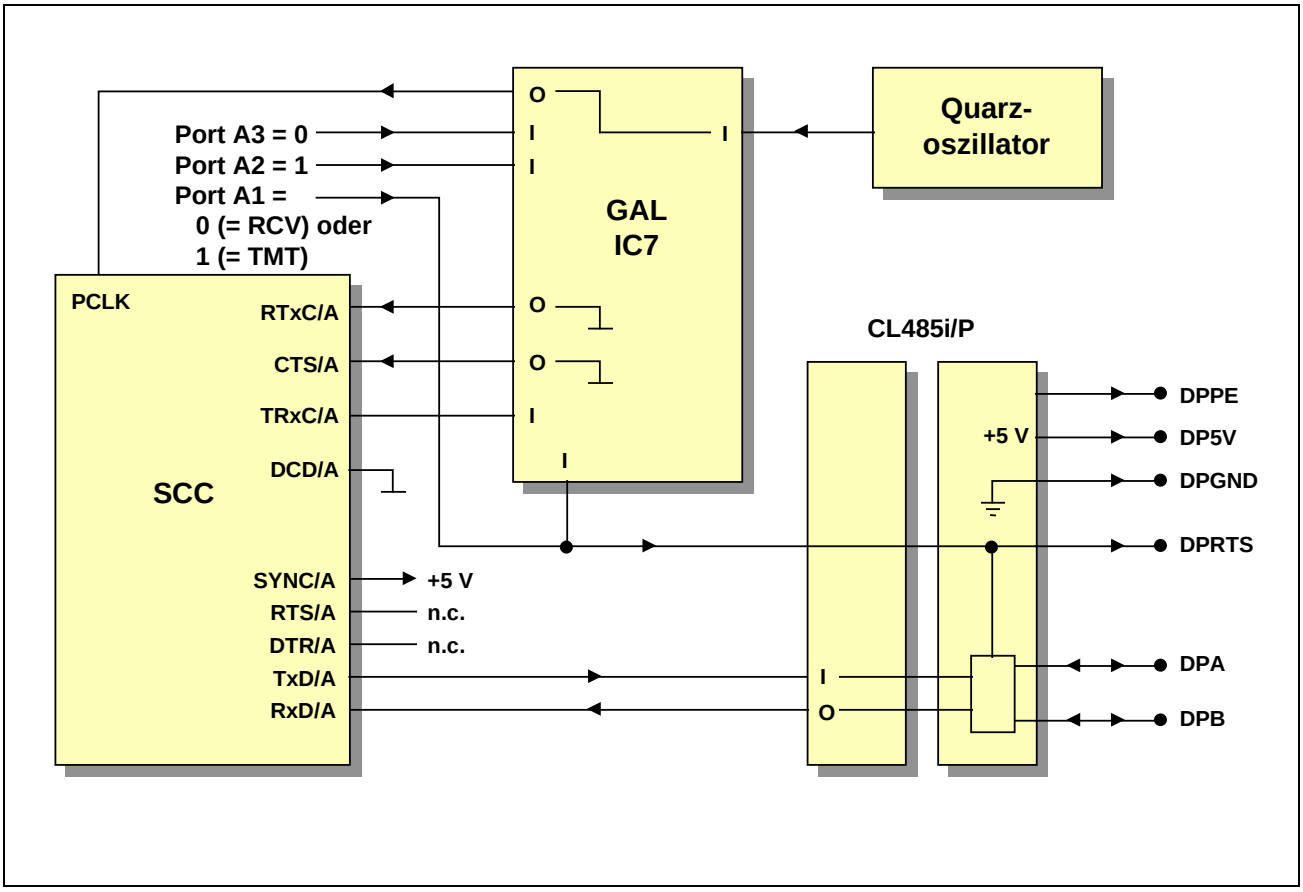


***Abb. 2-20**: C-Link-Adapter CL485i/P für RS-485 in Mode 2 und 3 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

C-Link-Adapter CL485i/U: RS-485-Schnittstelle

Für diesen C-Link-Adapter muß das Modul mit einem GAL "G032x07D" und, um auch SDLC und HDLC in vollem Umfang nutzen zu können, einem ESCC (Z85230) ausgerüstet sein.

Der C-Link-Adapter stellt eine galvanisch getrennte RS-485-Schnittstelle für max. Baudraten bis 20 MBaud zur Verfügung.

Die Umschaltung zwischen Senden und Empfangen geschieht über den Pin RTS/A bzw. RTS/B des SCC. Dies kann per Software oder, sofern RTS/A bzw. RTS/B im SCC entsprechend programmiert ist, auch automatisch geschehen.

Die zusätzliche Leitung DPRTS auf der galvanisch getrennten RS-485-Seite zeigt an, ob die RS-485-Schnittstelle Sender (DPRTS = log. 1) oder Empfänger (DPRTS = log. 0) ist. Diese Leitung wird von einem TTL-Buffer über einen Serienwiderstand von 300 Ω getrieben. Die Datenleitungen DPA und DPB sind überspannungsgeschützt und mit Pull-Up- (an DPB) bzw. Pull-Down-Widerständen (an DPA) von je 47 kΩ versehen. Der RS-485-Empfänger ist immer aktiv, beim Senden erscheint also das gesendete Signal wieder am Empfängereingang des SCC. Beachten Sie bitte, daß bei RS-485-Verbindungen + mit + und - mit - der Gegenstation verbunden wird.

Tabelle 2-15*:* Zusammenfassung der möglichen Modes mit CL485i/U. Nach einem Reset des Systems ist immer Mode 0 eingestellt, die SCC-Pins RTS/A und RTS/B sind = 1 (= Empfangen). Als Initialisierung muß Mode = 7 eingestellt und danach der Pin TRxC/A bzw. TRxC/B des SCC als Ausgang geschaltet werden (TRxC hat im übrigen keine Funktion bei diesem C-Link-Adapter). Mit den SCC-Pins RTS/A bzw. RTS/B kann per Software oder automatisch zwischen Empfangen und Senden umgeschaltet werden. RTS/A ist mit CTS/A und RTS/B mit CTS/B des SCC verbunden, um einen Interrupt auslösen zu können, wenn die Sende-/Empfangsrichtung umgeschaltet wird.

	A3	**A2 bzw.**	**A1**	**Pins des SCC**			**Funktion**	**TRxC Ein-/**	**SCC Pin**	**RS-485**	
Mode	**B3**	**B2**	**B1**	**CTS**	**DCD**	**SYNC**	**RTxC**	**Ausgang**	**RTS**	**DPRTS**	**Funktion**
0	0	0	0	= 0	= 0	= 1	keine	Eingang	0	0	Empfang
7	1	1	1	= 1	= 0	= 1	Quarz	Ausgang	1	0	Empfang
7	1	1	1	= 0	= 0	= 1	Quarz	Ausgang	0	1	Senden

Abschlußwiderstände

Das C-Link enthält keine Abschlußwiderstände.

Tabelle 2-16: Pinbelegung von St2 und Funktion der RS-485-Anschlüsse mit C-Link CL485i/U in Mode 0 und 7. Die Angaben für RTS beziehen sich auf die SCC-Pins RTS/A bzw. RTS/B (Angaben für D-Submin. Stecker gelten für SORCUS Kabel K2-2720).

		Funktion in Mode			**Pin (St2)**		**Pin D-**
RS-485	**RS-485**	**0**	**7**	**7**	**Kanal**		**Submin.**
Signal	**Ein-/Ausgang**		**RTS = 1**	**RTS = 0**	**A**	**B**	**9-pol.**
DPPE	Ausgang	GND	GND	GND	11	1	1
DP5V	Ausgang	+5 V	+5 V	+5 V	12	2	6
-	-	-	-	-	13	3	2
-	-	-	-	-	14	4	7
DPB	Ein-/Aus	-In	-In	-Out	15	5	3
DPA	Ein-/Aus	+In	+In	+Out	16	6	8
DPRTS	Ausgang	= 0	= 0	= 1	17	7	4
-	-	-	-	-	18	8	9
DPGND	Ausgang	GND	GND	GND	19	9	5
-	-	-	-	-	20	10	-

- = Pin ist nicht angeschlossen

CL485i/U für RS-485 in Mode 7:

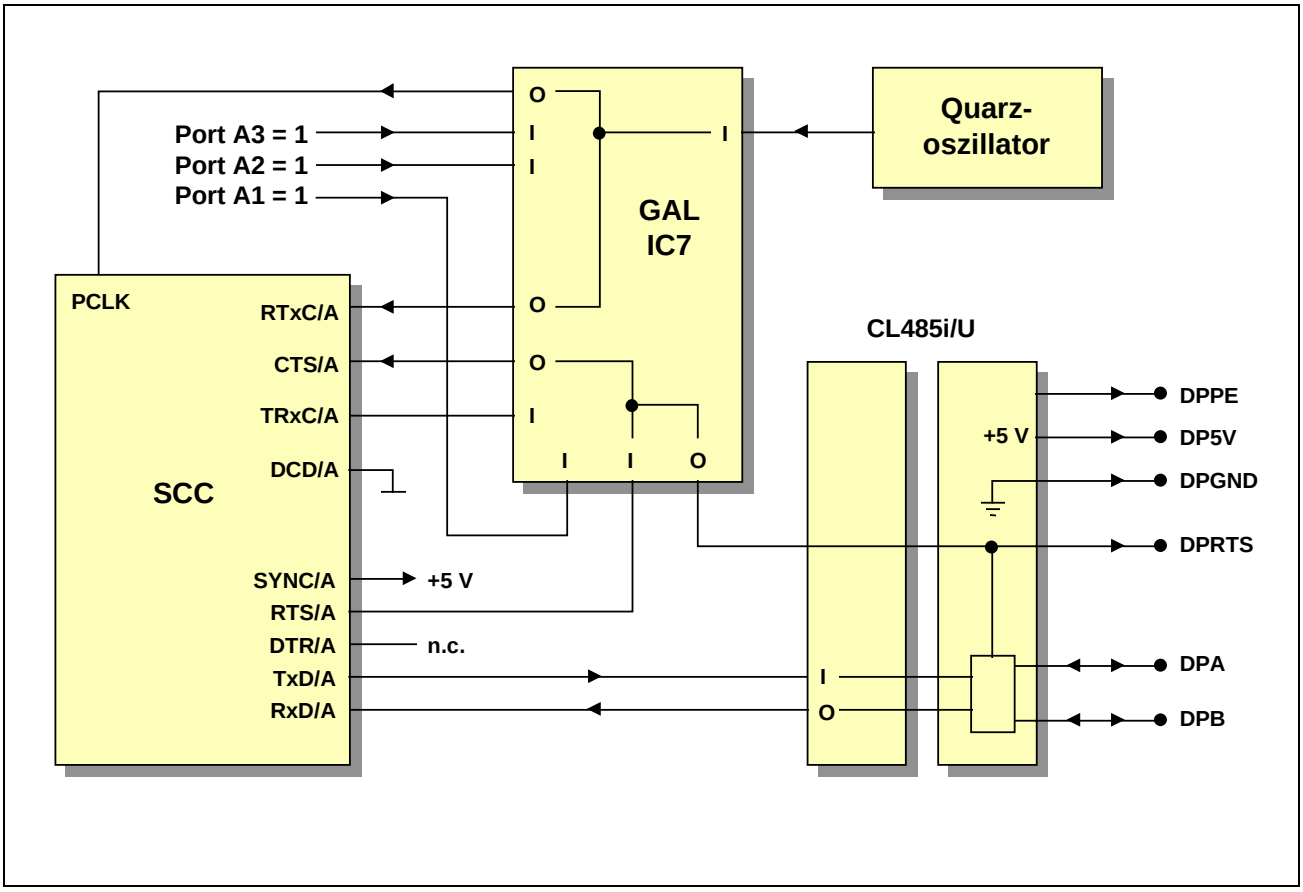


***Abb. 2-21**: C-Link-Adapter CL485i/U für RS-485 in Mode 7 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07D". Die Angaben gelten entsprechend auch für Kanal B.*

Hinweis: Bei RS-485-Teilnehmern anderer Hersteller sind die Bezeichnungen A (DPA) und B (DPB) evtl. vertauscht. Es muß dann A mit B und B mit A verbunden werden.

SORCUS verwendet die Bezeichnungen folgendermaßen: B > A => log. 1, B < A => log. 0 auf dem RS-485 Bus.

C-Link-Adapter CL200A: 20 mA Current Loop

Auf dem C-Link CL200A sind zwei 20 mA Konstantstromquellen vorhanden, je eine für Senden (TMT) und Empfangen (RCV). Ihre Verwendung ist optional. 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 C-Link galvanisch getrennt. Die Konfiguration (aktiv/passiv) wird über Verbindungen am externen Anschlußstecker (z. B. D-Submin.) eingestellt, kann also auch nach dem Einbau von Karte und Modul noch geändert werden.

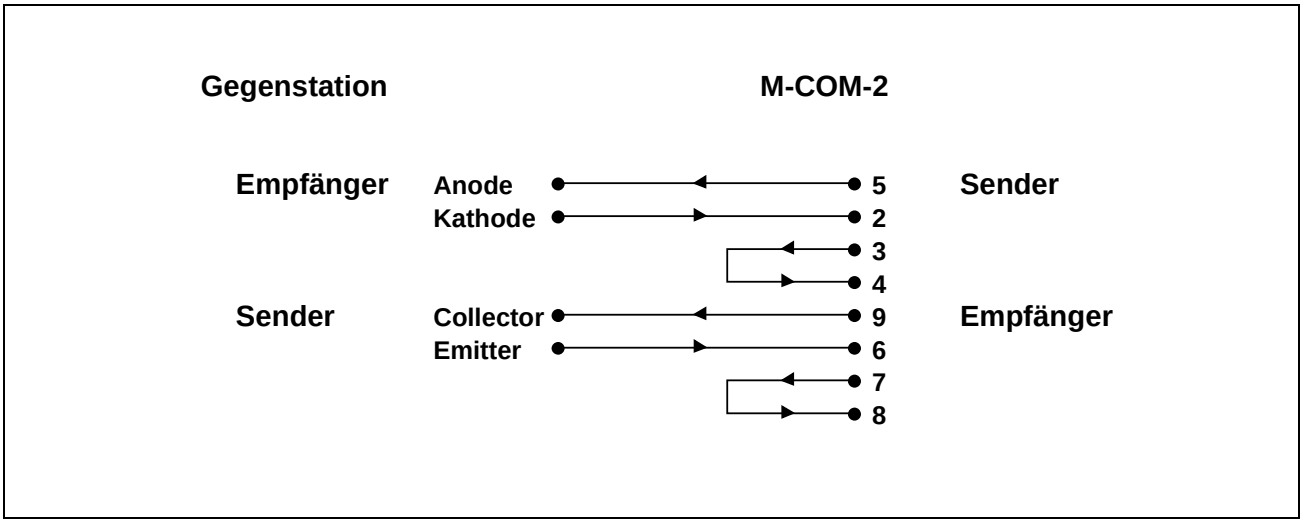
Beachten Sie bitte, daß bei 20 mA Verbindungen + mit - und - mit + der Gegenstation verbunden wird. Q1 und Q2 sind die Ausgänge der beiden 20 mA Konstantstromquellen.

Tabelle 2-17: 20 mA Current Loop mit C-Link CL200A

Signal C-Link	**Pin (St2) Kanal A**	**Pin (St2) Kanal B**	**RCV: TMT:**	**aktiv aktiv**	**aktiv passiv**	**passiv aktiv**	**passiv passiv**
-12 V	11	1		-	-	-	-
TMT-	12	2		TMT-	TMT-	TMT-	TMT-
TMT+	13	3		┐	TMT+	┐	TMT+
GND	14	4		┘*	-	┘*	-
Q1+	15	5		TMT+	-	TMT+	-
RCV-	16	6		RCV-	RCV-	RCV-	RCV-
RCV+	17	7		┐	┐	RCV+	RCV+
GND	18	8		┘*	┘*	-	-
Q2+	19	9		RCV+	RCV+	-	-
-	20	10		-	-	-	-

- := Pin ist nicht angeschlossen bzw. nicht vorhanden
* := Lötverbindungen am externen Anschlußstecker (z. B. D-Submin.)

Beispiel: Empfänger und Sender bei Modul M-COM-2, Kanal B aktiv (nicht galvanisch getr.) und mit Optokopplern einer Gegenstation verbunden.



Die Stromquellen auf dem Modul bzw. C-Link werden von +12 Volt gespeist. Um den Spannungshub zu vergrößern, kann die Rückführung statt an Ground (wie in Tabelle 2-18 angegeben) auch an -12 Volt erfolgen:

Tabelle 2-18: Vergrößerter Spannungshub bei 20 mA Current Loop mit C-Link CL200A

Signal C-Link	**Pin (St2) Kanal A**	**Pin (St2) Kanal B**	**RCV: TMT:**	**aktiv aktiv**	**aktiv passiv**	**passiv aktiv**	**passiv passiv**
-12 V	11	1					-
TMT-	12	2		TMT- *	TMT-	TMT- *	TMT-
TMT+	13	3			TMT+		TMT+
GND	14	4		-	- *	-	-
Q1+	15	5		TMT+ *	-	TMT+	-
RCV-	16	6		RCV-	RCV-	RCV-	RCV-
RCV+	17	7				RCV+	RCV+
GND	18	8		-	-	-	-
Q2+	19	9		RCV+	RCV+	-	-
-	20	10		-	-	-	-

- = Pin ist nicht angeschlossen bzw. nicht vorhanden
* = Lötverbindungen am D-Submin.-Stecker

Hinweise zur Programmierung (mit C-Link CL200A):

Für einen mit dem C-Link-Adapter CL200A ausgerüsteten Kanal muß Mode 1 eingestellt werden. Der Takt des Quarzoszillators liegt damit auch direkt am RTxC Pin dieses Kanals des SCC. Intern im SCC kann RTxC als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden.

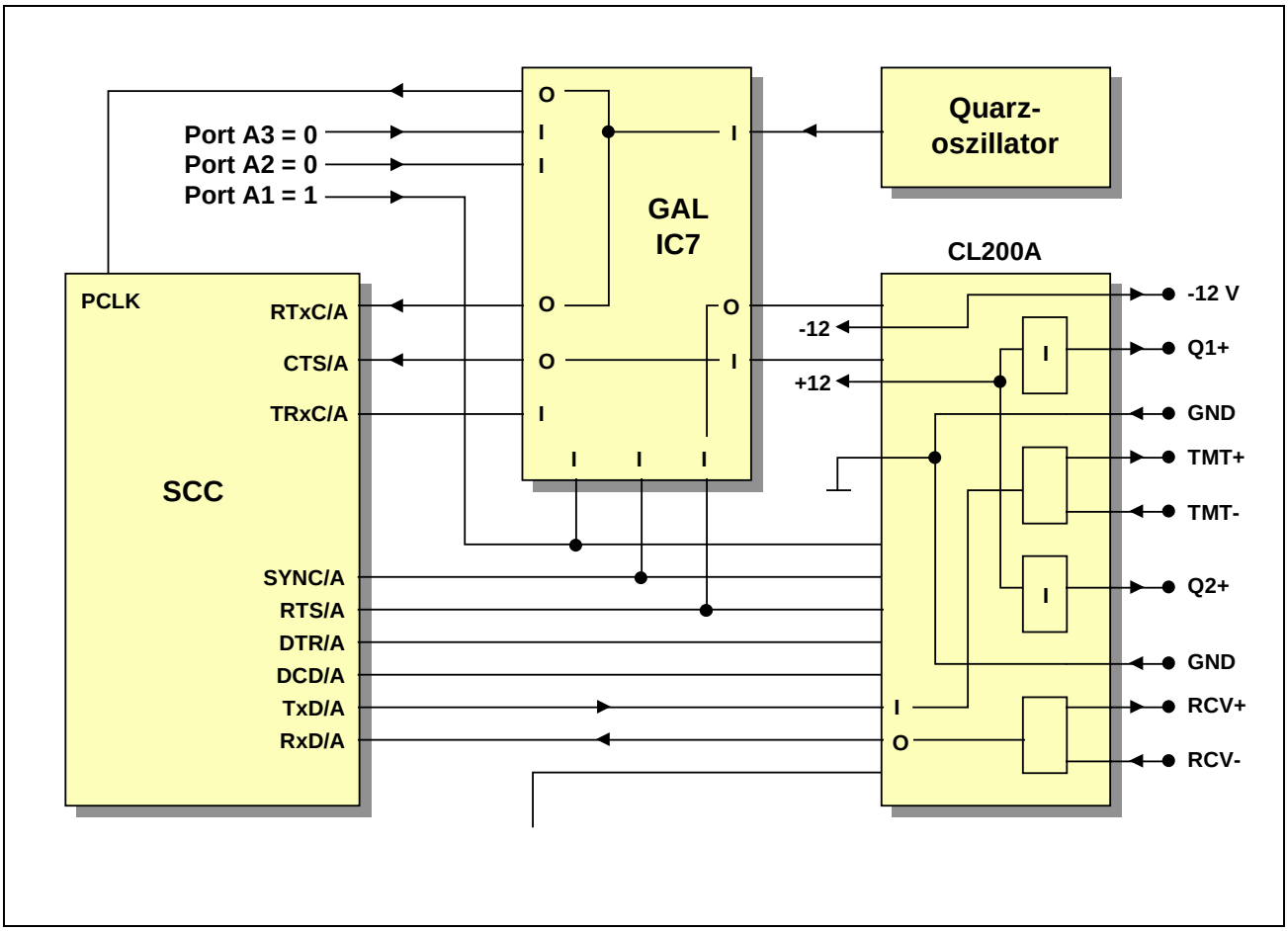


***Abb. 2-22**: C-Link-Adapter CL200A in Mode 1 für Kanal A des Moduls M-COM-2 (Rev. F) mit GAL "G032F07C". Die Angaben gelten entsprechend auch für Kanal B.*

Programmierung

Auf dem Modul müssen folgende Funktionsgruppen programmiert werden:

a) Anwahl einer Interrupt-Leitung (vom Modul zur Basiskarte)

b) Programmierung des SCC-Bausteins (Kanal A und B)

c) Konfiguration der Schnittstellen A und B (außerhalb SCC)

Zusätzlich zu den in dieser Beschreibung angegebenen Konfigurationsmöglichkeiten bestehen noch weitere Möglichkeiten, die ggf. den Austausch eines IC (GAL IC7) erforderlich machen. Falls Sie also spezielle Konfigurationen benötigen, sollten Sie dies anfragen. Alle in dieser Beschreibung gemachten Angaben beziehen sich auf eine Bestückung mit den ausgelieferten Standard-GALs. Beide Kanäle können unabhängig voneinander konfiguriert werden.

Anwahl einer Interrupt-Leitung zur Basiskarte

Das Modul ist interruptfähig. Die Interrupt-Leitung des Moduls kann per Software mit einem der Interrupt-Eingänge der Basiskarte verbunden werden. Während der Anwahl einer Interrupt-Leitung darf das Modul keinen Interrupt anfordern, d. h., im SCC müssen (vorübergehend) alle Interrupts maskiert werden, z. B. durch Reset des SCC. Die Anwahl einer Interrupt-Leitung geschieht durch Setzen der drei Interrupt-Anwahlleitungen C1, C2 und C3:

Interrupt-Leitung des Moduls an MODULAR-4/486	**C1**	**C2**	**C3**
keine	0	0	0
IRQ-A	0	1	0
IRQ-F	0	1	1
IRQ-B	0	0	1
IRQ-E	1	1	1
IRQ-C	1	1	0
IRQ-D	1	0	1

Programmierung des SCC-Bausteins

Da dieser Baustein etwas komplex ist, wird hierzu auf die Literatur von Zilog (zum Baustein Z8530, Z85C30 und Z85230) bzw. Intel (82530) verwiesen. Dabei sind die Bausteine Z8530, Z85C30 und 82530 untereinander kompatibel, der Baustein Z85230 stellt eine neuere und erweiterte Version dar (u. a. größere FIFOs).

Ein ausführliches Programmierhandbuch (in engl. Sprache) zu diesem Baustein ist bei SORCUS erhältlich (Best.-Nr.: MA-1529).

Bedingt durch die Konstruktion des Moduls sind bei der Programmierung des SCC einige Dinge zu berücksichtigen:

1. Die Pins "SYNC/A" und "SYNC/B" des SCC müssen im SCC als Eingänge konfiguriert werden. Sie werden für die Modem-Steuerleitungen "Ri/A" bzw. "Ri/B" verwendet, sofern diese Modem-Steuerleitung vom aufgesteckten C-Link zur Verfügung gestellt wird (wie z. B. beim C-Link CL232S, CL232A/i und CL232A/o). "SYNC/A" und "SYNC/B" werden auf dem Modul nicht für einen Quarzoszillator eingesetzt (siehe SCC-Beschreibung).

2. Die Pins "RTxC/A" und "RTxC/B" des SCC müssen im SCC als Eingänge konfiguriert werden. Sie werden auf dem Modul nicht für einen Quarzoszillator eingesetzt (siehe SCC-Beschreibung). Das an diesen Pins anliegende Signal kann innerhalb des SCC für den jeweiligen Kanal für folgende Zwecke verwendet werden:

Pin	**In-/Output**	**Verwendbar als**
RTxC/A	Input	RCV-Clock/A TMT-Clock/A Eingang der DPLL/A Eingang von Baudratengenerator/A
RTxC/B	Input	RCV-Clock/B TMT-Clock/B Eingang der DPLL/B Eingang von Baudratengenerator/B

3. Die Pins "TRxC/A" und "TRxC/B" des SCC müssen abhängig vom jeweils aufgesteckten C-Link-Adapter als Ein- oder Ausgang konfiguriert werden. Als Ausgang können sie eines der folgenden Signale des zugehörigen Kanals nach außen liefern:

Pin	**In-/Output**	**kann liefern**
TRxC/A	Output	TMT-Clock von Kanal A Ausgang Baudratengenerator von Kanal A Ausgang von RCV DPLL von Kanal A Signal an Pin "RTxC/A"
TRxC/B	Output	TMT-Clock von Kanal B Ausgang Baudratengenerator von Kanal B Ausgang von RCV DPLL von Kanal B Signal an Pin "RTxC/B"

Als Eingang kann TRxC folgendermaßen eingesetzt werden:

Pin	**In-/Output**	**Verwendbar als**
TRxC/A	Input	RCV-Clock für Kanal A TMT-Clock für Kanal A
TRxC/B	Input	RCV-Clock für Kanal B TMT-Clock für Kanal B

4. Am Pin "PCLK" des SCC liegt der Takt des Quarzoszillators des Moduls (standardmäßig 7,3728 MHz, optional eine andere Frequenz, z. B. 4,9152 MHz). Dieses Signal kann z. B. als Eingangstakt für die Baudratengeneratoren verwendet werden. Je nach Betriebsart (Mode) kann dieser Takt auch an RTxC angelegt werden und damit direkt als Eingangstakt für Sender und/oder Empfänger dienen, um hohe Baudraten zu erreichen.

Tabelle: Reihenfolge der Initialisierung für den SCC

Register	Data	Kommentar
1. Stufe: Betriebsarten und Konstanten definieren		
WR9	1100 0000	Hardware Reset
WR0	0000 00xx	Select Shift mode (nur bei Z8030)
WR4	xxxx xxxx	Transmit/Receive Control: Asyn- oder Synchrone Betriebsart anwählen
WR1	0xx0 0x00	W/REQ anwählen (optional)
WR2	xxxx xxxx	Interruptvektor programmieren
WR3	xxxx xxx0	Receiver control, Bit D0 (Rx enable) muß hier auf 0 gesetzt werden.
WR5	xxxx 0xxx	Transmit control, Bit D3 (Tx enable) muß hier auf 0 gesetzt werden.
WR6	xxxx xxxx	SYNC-Zeichen programmieren
WR7	xxxx xxxx	SYNC-Zeichen programmieren
WR9	000x 0xxx	Interrupt control anwählen. Bit D3 (MIE) muß auf 0 gesetzt werden.
WR10	xxxx xxxx	Kontrollwort (optional)
WR11	xxxx xxxx	Clock control
WR12	xxxx xxxx	Zeitkonstante, Lowbyte (optional)
WR13	xxxx xxxx	Zeitkonstante, Highbyte (optional)
WR14	xxxx xxx0	Kontrollwort, Bit 0 (BR enable) muß hier auf 0 gesetzt werden.
WR14	xxxS SSSS	Register kann mehrfach beschrieben werden, wenn mehr als ein Kommando geschickt werden soll
2. Stufe: Enables		
WR3	SSSS SSS1	Bit D0 (Rx Enable) auf 1 setzen.
WR5	SSSS 1SSS	Bit D3 (Tx Enable) auf 1 setzen
WR0	1000 0000	Reset TxCRC
WR14	000S SSS1	Baudraten Generator Enable, Bit D0 auf 1 setzen, Enable DPLL
WR1	xSS0 0S00	D7 auf 1 setzen, wenn DMA enabled werden soll.
Stufe 3: Interrupt Enable		
WR15	xxxx xxxx	Enable external interrupts
WR0	0001 0000	Reset EXT/STATUS
WR0	0001 0000	Reset EXT/STATUS
WR1	SSSx xSxx	Enable receive, transmit and external interrupt master.
WR9	000S xSSS	Enable Master Interrupt bit D3.

1 = Auf 1 setzen, 0 = auf 0 setzen, x = nach Wunsch setzen, S = so setzen wie bereits gesetzt

Arbeitsblatt:

Stufen	Register	Hex	Bit	Kommentar
1. Betriebsarten				
	WR9	_C_ _0_	1100 0000	Software Reset
	WR0	_0_ ____	0000 00__	_____
	WR4	___ ___	____ ____	_____
	WR1	___ ___	0__0 0_00	_____
	WR2	___ ___	____ ____	_____
	WR3	___ ___	____ ___0	_____
	WR5	___ ___	____ 0___	_____
	WR6	___ ___	____ ____	_____
	WR7	___ ___	____ ____	_____
	WR9	___ ___	000_ 0___	_____
	WR10	___ ___	____ ____	_____
	WR11	___ ___	____ ____	_____
	WR12	___ ___	____ ____	_____
	WR13	___ ___	____ ____	_____
	WR14	___ ___	____ ___0	_____
	WR14	___ ___	____ ___0	_____
2. Enables				
	WR3	___ ___	____ ___1	_____
	WR5	___ ___	____ 1___	_____
	WR0	_8_ _0_	1000 0000	Reset TxCRC
	WR14	___ ___	000_ ___1	_____
	WR1	___ ___	____ ____	_____
3. Interrupt				
	WR15	___ ___	____ ____	_____
	WR0	_1_ _0_	0001 0000	Reset Ext/Status
	WR0	_1_ _0_	0001 0000	Reset Ext/Status
	WR1	___ ___	____ ____	_____
	WR9	___ ___	000_ ____	_____

Initialisierungswerte nach Reset:

Register	**Hardware-Reset**	**Channel-Reset**
WR0	0000 0010	0000 0000
WR1	00_0 0_00	00_0 0_00
WR2	____ ____	____ ____
WR3	____ ___0	____ ___0
WR4	____ _1__	____ _1__
WR5	0__0 000_	0__0 0000
WR6	____ ____	____ ____
WR7	____ ____	____ ____
WR9	1100 00__	__0_ ____
WR10	0000 0000	0__0 0000
WR11	0000 1000	____ ____
WR12	____ ____	____ ____
WR13	____ ____	____ ____
WR14	__10 0000	__10 00__
WR15	1111 1000	1111 1000
RR0	01__ _100	01___100
RR1	0000 0111	0000 0111
RR3	0000 0000	0000 0000
RR10	0000 0000	0000 0000

Hochsprachenbibliothek

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M032_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Prozedur **m032_bib_startup** einmal aufgerufen werden.

m032_bib_startup — Initialisiere Modulbibliothek

Pascal PROCEDURE m032_bib_startup (micro_slot: byte);

C void EXPORT m032_bib_startup (byte micro_slot);

Funktion Diese Prozedur initialisiert die Modulbibliothek M032_LIB. Es werden u.a. die Initialisierungsdaten aus den EEPROMs aller M-COM-2 Module übernommen, die sich auf der Basiskarte befinden. Die Register des Gate-Arrays werden gemäß den EEPROM-Inhalten gesetzt.

m032_scc_reset — Führe SCC-Hardware-Reset durch

Pascal PROCEDURE m032_scc_reset (micro_slot);

C void EXPORT m032_scc_reset (byte micro_slot);

Funktion Diese Prozedur führt einen Hardware-Reset des SCC durch.

m032_init_scc — Initialisiere SCC-Kanal

Pascal: FUNCTION m032_init_scc (micro_slot, channel, mode, intmode, int_x_mode: byte; timeconst: word; use_brg, clkmode, databits, stopbits, parity_type: byte): byte;

C: byte EXPORT m032_init_scc (byte micro_slot, byte channel, byte mode, byte intmode, byte int_x_mode, ushort timeconst, byte use_brg, byte clkmode, byte databits, byte stopbits, byte parity_type);

Funktion: Diese Funktion initialisiert einen Kanal des Moduls.

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

mode: Einstellung der Taktquellen für Receiver und Transmitter

mode	CTS	RTS	Funktion von RTxC	Funktion von TRxC	TRxC Ein-/Ausgang
0	CTS	RTS	Keine / EXT	CLKin von Ri	Eingang
1	CTS	RTS	Quarz	keine	Eingang
2	an RTxC	disabled	von CTS	keine	Eingang
3	an RTxC	RTS	von CTS	keine	Eingang
4	res.	res.	res.	res.	res.
5	CTS	von TRxC/RTS	Quarz	an RTS/EXT	Ausgang
6	res.	res.	res.	res.	res.
7	CTS	RTS	Quarz	keine	Ausgang

int_mode: Interrupt-Betriebsart festlegen

7	6	5	4	3	2	1	0	
								Interrupt Enable
0	0	0						Reserviert
			1	0				Interrupt On All Rx Character or Special Condition
			0	0				Rx Interrupt Disable
					1			Parity is Special Condition
						1		Tx Interrupt Enable
							1	Ext. Interrupt Enable

int_x_mode: Erweiterte Interrupt-Betriebsart festlegen

7	6	5	4	3	2	1	0	
								Ext./Status Interrupt Select
1								Break/Abort Interrupt Enable
	0							reserviert
		1						CTS Interrupt Enable
			0					reserviert
				1				DCD Interrupt Enable
					0	0	0	reserviert

timeconst: Zeitkonstante für Baudratengenerator (0 bis ffffh)

clkmode: Clock-Mode (1, 16, 32, 64) für Receiver und Transmitter

use_brg: Baudratengenerator verwenden (1) oder nicht (0)

databits: Anzahl Datenbits (5 bis 8)

stopbits: Anzahl Stopbits (0 = 0, 1 = 1, 2 = 1,5 3 = 2 Stopbits)

parity_type: Parität (0 = keine Parität, 1 = odd, 2 = even)

m033_define_intrp_connection Interruptanwahl

Pascal PROCEDURE m033_define_intrp_connection (micro_slot, intrp: byte);

C void EXPORT m033_define_intrp_connection (byte micro_slot, byte intrp);

Funktion Diese Prozedur wählt einen Interruptleitung zur Basiskarte an.

Parameter *intrp*: Nummer der Interrupt-Leitung (0 = kein Interrupt, 1 = IRQ-A, 2 = IRQ-F, 3 = IRQ-B, 4 = IRQ-E, 5 = IRQ-C und 6 = IRQ-D)

m032_transmit_scc_character Sende Zeichen

Pascal FUNCTION m032_transmit_scc_character (micro_slot, channel, data: byte): byte;

C byte EXPORT m032_transmit_scc_character (byte micro_slot, byte channel, byte data);

Funktion Diese Funktion sendet ein Zeichen. Vor dem Senden wird der Zustand des Sendepuffers abgefragt. Ist der Sendepuffer voll (TBE = 0), so wird 1 zurückgeliefert. Wurde das Zeichen erfolgreich in den Sendepuffer geschrieben, liefert die Funktion 0 zurück.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

data: Zu sendendes Datenbyte

m032_receive_scc_character — Empfange Zeichen

Pascal: FUNCTION m032_receive_scc_character (micro_slot, channel: byte; var data: byte): byte;

C: byte EXPORT m032_receive_scc_character (byte micro_slot, byte channel, byte *data);

Funktion: Diese Funktion liest den Empfangspuffer eines Kanals. Vor dem Empfangen wird der Zustand des Empfangspuffers abgefragt. Ist dieser leer (RBF = 0), so wird 1 zurückgeliefert. Wurde ein Zeichen empfangen, liefert die Funktion 0 zurück.

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

data: Empfangenes Datenbyte

m032_scc_rbf — Frage Empfangspuffer-Status (RBF) ab

Pascal: FUNCTION m032_scc_rbf (micro_slot, channel: byte): byte;

C: byte EXPORT m032_scc_rbf (byte micro_slot, byte channel);

Funktion: Diese Funktion prüft den Status des Empfangspuffers eines Kanals. Der Zustand des RBF-Bits wird zurückgegeben (RBF = 1: receive buffer full, RBF = 0 receive buffer empty).

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

m032_scc_tbe — Frage Sendepuffer-Status (TBE) ab

Pascal: FUNCTION m032_scc_tbe (micro_slot, channel: byte): byte;

C: byte EXPORT m032_scc_tbe (byte micro_slot, byte channel);

Funktion: Diese Funktion prüft den Status des Sendepuffers eines Kanals. Der Zustand des TBE-Bits wird zurückgegeben (TBE = 1: transmit buffer empty, TBE = 0 transmit buffer full).

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

m032_scc_cts — Frage Clear To Send (CTS) ab

Pascal FUNCTION m032_scc_cts (micro_slot, channel: byte): byte;

C byte EXPORT m032_scc_cts (byte micro_slot, byte channel);

Funktion Diese Funktion liefert den aktuellen Zustand der CTS-Leitung eines Kanals.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

m032_set_rts — Setze Request To Send (RTS)

Pascal PROCEDURE m032_set_rts (micro_slot, channel, rts: byte);

C void EXPORT m032_set_rts (byte micro_slot, byte channel, byte rts);

Funktion Diese Prozedur setzt die RTS-Leitung eines Kanals auf den angegebenen Wert.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

rts: Zu setzender Wert (0 oder 1)

m032_scc_intvec — Lies Interrupt-Vektor

Pascal PROCEDURE m032_scc_intvec (micro_slot: byte; var int_vector, rr0, rr1, rr3: byte);

C void EXPORT m032_scc_intvec (byte micro_slot, byte *int_vector, byte *rr0, byte *rr1, byte *rr3);

Funktion Diese Funktion kann dazu verwendet werden, nach einer SCC-Interrupt-Anforderung, den Interrupt-Vektor zu lesen.

Parameter *vector*: Interrupt-Vektor

Bit 3 bis 1 (V3, V2 und V1) enthalten weitere Informationen über den aufgetretenen Interrupt. Die restlichen Bits sind reserviert.

rr0: Aktueller Zustand von Read-Register 0

7	6	5	4	3	2	1	0	
□	□	□	□	□	□	□	□	Read Register 0
□	■	■	■	■	■	■	■	Break/Abort
■	□	■	■	■	■	■	■	Tx Underun/EOM
■	■	□	■	■	■	■	■	CTS
■	■	■	□	■	■	■	■	Sync/Hunt
■	■	■	■	□	■	■	■	DCD
■	■	■	■	■	□	■	■	Tx Buffer Empty
■	■	■	■	■	■	□	■	Zero Count
■	■	■	■	■	■	■	□	Rx Character Available

rr1: Aktueller Zustand von Read-Register 1

7	6	5	4	3	2	1	0	
□	□	□	□	□	□	□	□	Read Register 1
□	■	■	■	■	■	■	■	End of Frame
■	□	■	■	■	■	■	■	CRC/Framing Error
■	■	□	■	■	■	■	■	Rx Overrun Error
■	■	■	□	■	■	■	■	Parity Error
■	■	■	■	□	■	■	■	Residue Code 0
■	■	■	■	■	□	■	■	Residue Code 1
■	■	■	■	■	■	□	■	Residue Code 2
■	■	■	■	■	■	■	□	All sent

rr3: Aktueller Zustand von Read-Register 3

7	6	5	4	3	2	1	0	
□	□	□	□	□	□	□	□	Read Register 3 (Interrupt Pending (IP) Register)
□	■	■	■	■	■	■	■	0
■	□	■	■	■	■	■	■	0
■	■	□	■	■	■	■	■	Channel A Rx IP
■	■	■	□	■	■	■	■	Channel A Tx IP
■	■	■	■	□	■	■	■	Channel A Ext/Status IP
■	■	■	■	■	□	■	■	Channel B Rx IP
■	■	■	■	■	■	□	■	Channel B Tx IP
■	■	■	■	■	■	■	□	Channel B Ext/Status IP

m032_baudrate_calc Ermittle Einstellungen für Baudrate

Pascal
PROCEDURE m032_baudrate_calc (baudrate, quartz: longint;
var bauderror: longint;var basereg: byte; var clk_mode: byte;
var timeconst: word, var use_brg: byte);

C
void EXPORT m032_baudrate_calc (long baudrate, long quartz,
long *bauderror, byte *basereg, byte *clk_mode, ushort *timeconst,
byte *use_brg);

Funktion
Diese Prozedur ermittelt die notwendigen Einstellungen für eine gewünschte Baudrate bei einer bestimmten Frequenz des Quarzoszillators. Als Ergebnisse liefert die Prozedur die SCC-Parameter für den Baudratengenerator.

Parameter

baudrate: Gewünschte Baudrate in bd

quartz: Gibt die Taktfrequenz des Quarzoszillators in Hz an (Standard: 7,3728 MHz ≙ 7370000)

bauderror: Absoluter Fehler des Basistaktes bzw. der Baudrate der sich bei dieser Einstellung ergibt

basereg: Reserviert

timeconst: Einstellung für die Zeitkonstante des Baudratengenerators

clk_mode: Einstellung für Clock-Mode des Receivers/Transmitters

use_brg: Verwendung des Baudratengenerators (0=nein, 1=ja)

m032_set_channel — Konfiguriere Kanal

Pascal
FUNCTION m032_set_channel (micro_slot, channel: byte; baudrate: word; flag, scc_mode, scc_intmode, scc_int_x_mode, databits, stopbits, parity_type: byte; var bauderror: longint; var used_clkmode: byte): byte);

C
byte EXPORT m032_set_channel (byte micro_slot, byte channel, long baudrate, byte flag, byte mode, byte intmode, byte int_x_mode, byte databits, byte stopbits, byte parity_type, long *bauderror, byte *used_clkmode);

Funktion
Diese Funktion stellt eine Kombination aus **m032_baudrate_calc** und **m032_init_scc** dar (siehe auch Parameter dieser Funktionen).

Parameter
flag: Reserviert

Programmierung mit I/O-Zugriffen

Lokale I/O-Adressen

Adresse	Zugr.	Funktion							
MBA+00h	RW8	SCC: Kanal B, Control							
MBA+01h	RW8	Kanal B, Data							
MBA+02h	RW8	Kanal A, Control							
MBA+03h	RW8	Kanal A, Data							
MBA+07h	W8	Interrupt anwählen: Port C1 [1]							
MBA+08h	W8	Port C2 [1]							
MBA+09h	W8	Port C3 [1]							
		Interrupt:	Keiner	IRQ-A	IRQ-B	IRQ-C	IRQ-D	IRQ-E	IRQ-F
		C3, C2, C1:	000	010	100	011	101	111	110
MBA+0ah	W8	Konfiguration Schnittstelle A: Port A1 [1]							
MBA+0bh	W8	Port A2 [1]							
MBA+0ch	W8	Port A3 [1]							
MBA+0dh	W8	Konfiguration Schnittstelle B: Port B1 [1]							
MBA+0eh	W8	Port B2 [1]							
MBA+0fh	W8	Port B3 [1]							
MBA+08h	R8	Status lesen: DSR/A [2]							
MBA+0ah	R8	DSR/B [2]							
MBA+0ch	R8	Port C1 [2, 3]							

[1] Beim Schreiben wird der Port entsprechend Bit 0 gesetzt, Bit 1 bis Bit 7 sind ohne Bedeutung.

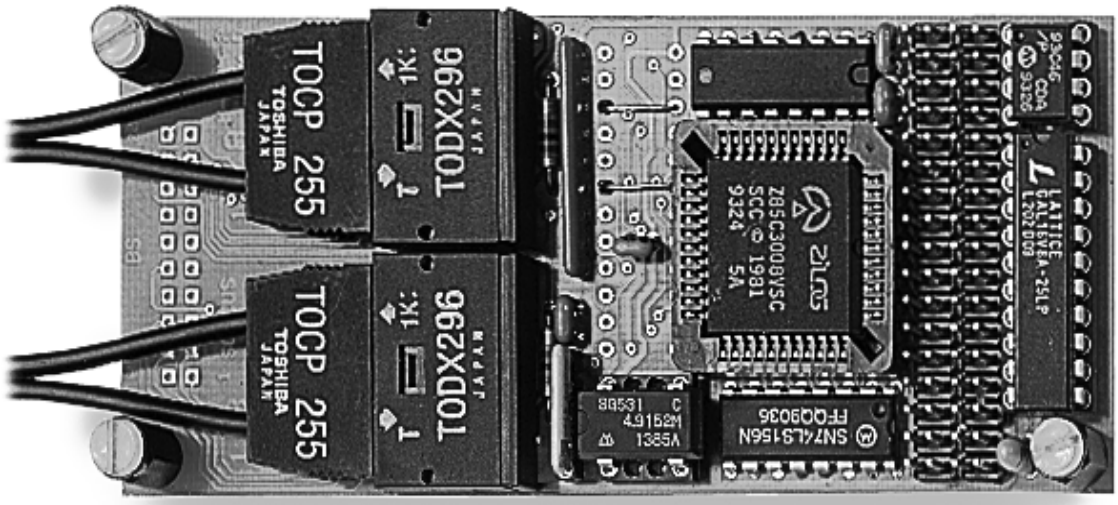
[2] Beim Lesen steht das Ergebnis in Bit 0, die anderen Bit sind ungültig.

[3] Nur für Testzwecke.

Notizen

3. M-COM-2/P

Zwei serielle Schnittstellen für Lichtwellenleiter



Funktionsbeschreibung

Diese Beschreibung bezieht sich auf Rev. D, E und F der Module M-COM-2 mit Lichtwellenleiter-Interface System Toshiba.

Die Module M-COM-2/P/4, M-COM-2/G/4, M-COM-2/P/8 und M-COM-2/G/8 enthalten jeweils zwei identische serielle synchrone/asynchrone Schnittstellen für den Anschluß von Lichtwellenleitern (im folgenden mit LWL abgekürzt). Entsprechend der verwendeten Basiskarte und dem Typ der LWL (Glas oder Plastik) sind folgende Versionen lieferbar (der Vollständigkeit halber sind auch die beiden Universalversionen des Moduls für den Einsatz mit C-Link Adaptern aufgeführt):

Modulversion	**Typ**	**für Basiskarte**	**Bestückungsvariante**
M-COM-2/P/4	33	MODULAR-4/Z80 und /Z280	Plastik-LWL für beide Kanäle
M-COM-2/G/4	33	MODULAR-4/Z80 und /Z280	Glas-LWL für beide Kanäle
M-COM-2/P/8	33	MODULAR-4/486	Plastik-LWL für beide Kanäle
M-COM-2/G/8	33	MODULAR-4/486	Glas-LWL für beide Kanäle
M-COM-2/4	32	MODULAR-4/Z80 und /Z280	C-Link-Adapter je Kanal
M-COM-2/8	32	MODULAR-4/486	C-Link-Adapter je Kanal

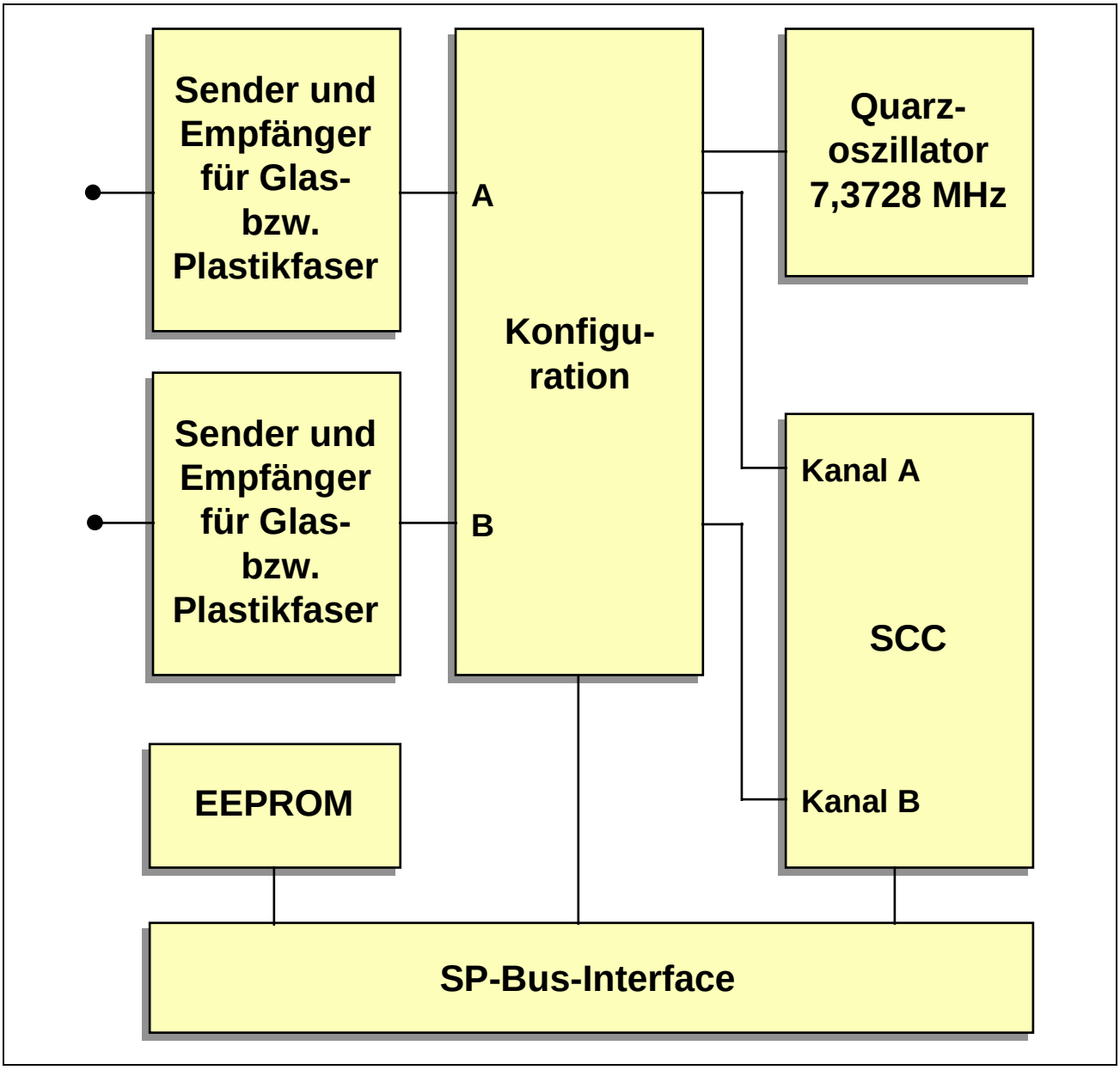
Als Standardbestückung sind die Module mit einem Quarzoszillator von 7,3728 MHz ausgerüstet, sie können auf Wunsch aber auch mit einer anderen Frequenz, z. B. 4,9152 MHz geliefert werden.

Bei den Versionen M-COM-2/4 und /8 erfolgt die Konfiguration der physikalischen Schnittstellen über je einen steckbaren C-Link Adapter. Sie sind verfügbar für RS-232, RS-232 isoliert, RS-422, RS-422 isoliert, RS-485, RS-485 isoliert und 20 mA isoliert.

Die beiden seriellen Schnittstellen des Moduls sind unabhängig voneinander und unterschiedlich konfigurierbar. Sie sind mit einem SCC-Baustein Z8530 (bzw. 85C30 = CMOS-Version) oder optional Z85230 (= verbesserte und erweiterte Version mit größeren FIFOs) ausgerüstet. Außerdem enthält das Modul einen eigenen Quarzoszillator und zwei Baudratengeneratoren (einen je Kanal), ist also unabhängig vom CPU-Takt oder von Timern der Basiskarte.

Das Modul ist Interrupt- und DMA-fähig, der Interrupt vom Modul zur Basiskarte kann per Software angewählt und an IRQ-A bis IRQ-F der Basiskarte gelegt werden.

Blockschaltbild



Technische Daten

Parameter	Wert	Einheit
Anzahl serieller Schnittstellen	2	-
Serieller Kommunikationsbaustein (Option:	Z8530-8 Z85230-20)	-
Interruptfähig zur Basiskarte [1]	ja	-
Modem-Steuerleitungen je Schnittstelle	keine	-
Baudratengeneratoren Quarzoszillator: 7,3728 MHz (Option: 4,9152 MHz)	2	-
Versorgungsspannungen (von der Basiskarte)	+5	V
Stromaufnahme +5 V, typ. (Sender und Empfänger stromlos)	90	mA
Betriebstemperatur	0 bis 70	°C
Abmessungen (L x B x H)	106 x 45 x 15	mm

Lieferumfang

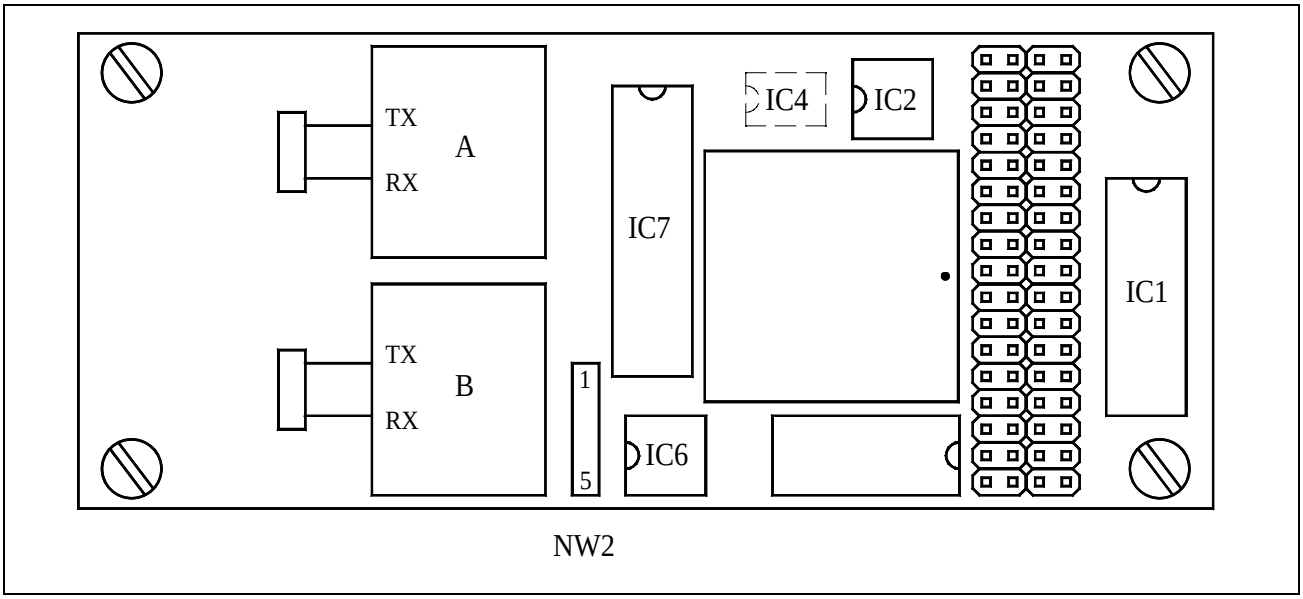
- Modul M-COM-2/...
- Diese Beschreibung
- Datenträger mit Programmbibliotheken (Pascal und C)

[1] Per Software ist bei MODULAR-4/486 einer von 6 Interrupt-Eingängen der Basiskarte wählbar. Der Interrupt-Ausgang des Moduls ist aktiv Low. Da die Interrupt-Eingänge der Basiskarte flankengesteuert sind, muß der entsprechende Interrupt-Eingang also auf negative Flanke programmiert werden.

Konfiguration und Einbau

Das Modul enthält keine Jumper, alle Einstellungen werden nach dem Einbau per Software vorgenommen.

Lageplan, Rev. F



Das Netzwerk NW2 ist nicht bestückt. Das gestrichelt eingezeichnete IC4 ist auf der Lötseite angebracht.

EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0110	0010 0001	2621h	Modultyp M-COM-2/P
1	0000 0000	0000 0001	0001h	Initialisierung
2	0010 0010	0001 0000	2210h	Bestückung
3	0000 0111	0011 0111	0737h	Quarzfrequenz
4	0000 0100	0000 0100	0404h	Bestückung und Umbau (Kanal A)
5	0000 0100	0000 0100	0404h	Bestückung und Umbau (Kanal B)
6	0000 0000	0000 1010	000ah	Physikalisches Interface (Kanal A)
7	0000 0000	0000 1010	000ah	Physikalisches Interface (Kanal B)
8	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	WORT-0: Kennung
								0	0	1	0	0	0	0	1	Modultyp: 33 = M-COM-2/P
				0	1	1	0									Revision: 1 = A, 2 = B, 3 = C,..., 6 = F
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort kann eingestellt werden, ob das Modul nach dem Einschalten und bei einem Hardware-Reset entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															1	Init nach Hardware-Reset: 0 = nein, 1 = ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-2: Bestückung

Bei der Angabe der GAL-Versionen (GALs sind vorprogrammierte ICs) steht das "x" für die Revision des Moduls. Bei einem Modul M-COM-2/P, Rev. F, trägt das GAL für IC1 also z. B. ein Etikett mit der Aufschrift "G033F01B".

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	WORT-2: Bestückung (werks. Einst.)
																geändert am: von:
												0	0	0	0	Typ SCC: 0 = SCC (8530) 1 = ESCC (85130) 2 = ESCC (85230)
								0	0	0	1					Max. Takt (SCC): 0 = 6 MHz 1 = 8 MHz 2 = 10 MHz 3 = 16 MHz 4 = 20 MHz
				0	0	1	0									GAL-Version IC1: 0 = unbekannt 1 = Vers. A ("G033x01A") 2 = Vers. B ("G033x01B")
0	0	1	0													GAL-Version IC7: 0 = unbekannt 1 = Vers. A ("G033x07A") 2 = Vers. B ("G033x07B") 3 = Vers. C ("G033x07C")

WORT-3: Quarzfrequenz

Die Angabe erfolgt mit 4 Ziffern in MHz mit 2 Vorkomma- und 2 Nachkommastellen. 4,9152 MHz würde also aufgerundet auf 4,92 und mit 4 Dezimalziffern 0492 (= 0492h) eingegeben. 0000h bedeutet, daß der Quarzoszillator nicht bestückt ist, die folgende Angabe 0737h bedeutet 7,37 MHz.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	1	WORT-3: Quarzfrequenz (werks. Einst.)
																geändert am: von:
												0	1	1	1	2. Ziffer nach Komma
								0	0	1	1					1. Ziffer nach Komma
				0	1	1	1									2. Ziffer
0	0	0	0													1. Ziffer

WORT-4 und WORT-5: Bestückung und Umbau

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	WORT-4: Hardware Kanal A (werks. Einst.)
																geändert am: von:
															0	1 = C-Link möglich 0 = nicht vorgesehen
														0		1 = 24-poliger Sockel 0 = nicht vorbereitet
													1			1 = C-Link eingelötet 0 = nicht eingelötet
								0	0	0	0	0				Reserviert
							0									1 = Interface möglich 0 = nicht vorgesehen
						0										1 = Sockel 0 = nicht vorbereitet
					1											1 = fest eingelötet 0 = nicht eingelötet
				0												1 = Modul umgebaut auf Revision F 0 = nicht umgebaut
0	0	0	0													Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	WORT-5: Hardware Kanal B (werks. Einst.)
																geändert am: von:
															0	1 = C-Link möglich 0 = nicht vorgesehen
														0		1 = 24-poliger Sockel 0 = nicht vorbereitet
													1			1 = C-Link eingelötet 0 = nicht eingelötet
								0	0	0	0	0				Reserviert
							0									1 = Interface möglich 0 = nicht vorgesehen
						0										1 = Sockel 0 = nicht vorbereitet
					1											1 = fest eingelötet 0 = nicht eingelötet
				0												1 = Modul umgebaut auf Revision F 0 = nicht umgebaut
0	0	0	0													Reserviert

WORT-6 und WORT-7: Physikalisches Interface (Kanal A und B)

In WORT-6 steht ein Code für das physikalische Interface für Kanal A, in diesem Fall also für das eingelötete LWL-Interface (Herstellerfirma und Typ des LWL-Kabels), in WORT-7 für Kanal B.

Code		Physikalisches Interface	isol.	z. B. C-Link
0	(00h)	keines	-	-
1	(01h)	RS-232, mit allen Modem-Leitungen	nein	CL232S
2	(02h)	RS-232, nur 5 V Versorgung	nein	CL232V
3	(03h)	RS-232	nein	CL232A
4	(04h)	RS-232 (RCV, TMT, RTS/CLK_{out}, CTS/CLK_{in})	ja	CL232i
5	(05h)	20 mA	ja	CL200A
6	(06h)	RS-422 / RS-485	nein	CL422S
7	(07h)	RS-422 / RS-485	ja	CL422i
8	(08h)	Lichtleiter HP-System, seitlich	ja	CL800Q
9	(09h)	Lichtleiter HP-System, oben	ja	CL800L
10	(0ah)	Lichtleiter Toshiba Glas PCS (TODX296)	ja	CL800Q
11	(0bh)	Lichtleiter Toshiba Plastik APF (TODX297)	ja	CL800L
12	(0ch)	RS-232, Pin EXT als CLK_{in}	nein	CL232A/i
13	(0dh)	RS-232, Pin EXT als CLK_{out}	nein	CL232A/o

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	WORT-6: Interface Kanal A (werks. Einst.)
																geändert am: von:
								0	0	0	0	1	0	1	0	Code für Interface A
0	0	0	0	0	0	0	0									Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	WORT-7: Interface Kanal B (werks. Einst.)
																geändert am: von:
								0	0	0	0	1	0	1	0	Code für Interface B
0	0	0	0	0	0	0	0									Reserviert

Programmierung

Auf dem Modul müssen folgende Funktionsgruppen programmiert werden:

a) Anwahl einer Interrupt-Leitung (vom Modul zur Basiskarte)

b) Programmierung des SCC-Bausteins (Kanal A und B)

c) Konfiguration der Schnittstellen A und B (außerhalb SCC)

Beide Kanäle können unabhängig voneinander konfiguriert werden. Zusätzlich zu den in dieser Beschreibung angegebenen Konfigurationsmöglichkeiten bestehen noch weitere Möglichkeiten, die ggf. den Austausch eines IC (GAL IC7) erforderlich machen. Falls Sie also spezielle Konfigurationen benötigen, sollten Sie dies anfragen. Alle in dieser Beschreibung gemachten Angaben beziehen sich auf eine Bestükkung mit dem GAL "G033x07A" (x = D, E oder F).

Anwahl einer Interrupt-Leitung

Das Modul ist interruptfähig. Die Interrupt-Leitung des Moduls kann per Software mit einem der Interrupt-Eingänge der Basiskarte verbunden werden. Während der Anwahl einer Interrupt-Leitung darf das Modul keinen Interrupt anfordern, d. h., im SCC müssen (vorübergehend) alle Interrupts maskiert werden, z. B. durch Reset des SCC. Die Anwahl einer Interrupt-Leitung geschieht durch Setzen der drei Interrupt-Anwahlleitungen C1, C2 und C3 (siehe Abschnitt 'Programmierung mit I/O-Zugriffen'):

Interrupt-Leitung des Moduls an MODULAR-4/486	**C1**	**C2**	**C3**
keine	0	0	0
IRQ-A	0	1	0
IRQ-F	0	1	1
IRQ-B	0	0	1
IRQ-E	1	1	1
IRQ-C	1	1	0
IRQ-D	1	0	1

Programmierung des SCC-Bausteins

Da dieser Baustein etwas komplex ist, wird hierzu auf die Literatur von Zilog (zum Baustein Z8530, Z85C30 und Z85230) bzw. Intel (82530) verwiesen. Dabei sind die Bausteine Z8530, Z85C30 und 82530 untereinander kompatibel, der Baustein Z85230 stellt eine neuere und erweiterte Version dar (u. a. größere FIFOs).

Ein ausführliches Programmierhandbuch (in engl. Sprache) zu diesen Bausteinen ist bei SORCUS erhältlich (Best.-Nr.: MA-1529).

Bedingt durch die Konstruktion des Moduls sind bei der Programmierung des SCC einige Dinge zu berücksichtigen:

1. Die Pins "SYNC/A" und "SYNC/B" des SCC müssen im SCC als Eingänge konfiguriert werden. Sie werden nicht für einen Quarzoszillator eingesetzt (siehe SCC-Beschreibung).

2. Die Pins "RTxC/A" und "RTxC/B" des SCC müssen im SCC als Eingänge konfiguriert werden. Sie werden nicht für einen Quarzoszillator eingesetzt (siehe SCC-Beschreibung). Das an diesen Pins anliegende Signal kann innerhalb des SCC für den jeweiligen Kanal für folgende Zwecke verwendet werden:

Pin	**In-/Output**	**Verwendbar als**
RTxC/A	Input	RCV-Clock/A TMT-Clock/A Eingang der DPLL/A Eingang von Baudratengenerator/A
RTxC/B	Input	RCV-Clock/B TMT-Clock/B Eingang der DPLL/B Eingang von Baudratengenerator/B

3. Die Pins "TRxC/A" und "TRxC/B" des SCC müssen immer als Eingänge konfiguriert werden (Bit-2 in SCC-Write-Register 11 von beiden Kanälen = 0). Beide Pins haben hier keine Funktion, mit einer Ausnahme: Für besondere Anwendungen liegt an Pin "TRxC/A" das Signal des Empfängers von Kanal B. Es wäre also denkbar, hierüber einen Takt für Kanal A einzuspeisen.

4. Am Pin "PCLK" des SCC liegt der Takt des Quarzoszillators des Moduls (standardmäßig 7,3728 MHz, optional 4,9152 MHz). Dieses Signal kann z. B. als Eingangstakt für die Baudratengeneratoren verwendet werden. Je nach Betriebsart (Mode) kann dieser Takt auch an RTxC angelegt werden und damit direkt als Ein-

gangstakt für Sender und/oder Empfänger dienen, um hohe Baudraten zu erreichen.

Konfigurationsmöglichkeiten mit GAL "G033x07A"

Die Lichtwellenleiter-Interfaces sind fest eingelötet (Toshiba-Steckersystem TOS-LINK). Lieferbar sind für beide Modulversionen auch fertig konfektionierte Lichtwellenleiterkabel mit Glas- bzw. Plastikfaser.

Drei Ports, also 3 Bit stehen zur Konfiguration für jede Schnittstelle zur Verfügung: Port A1, A2 und A3 für Schnittstelle A, Port B1, B2 und B3 für Schnittstelle B. Damit lassen sich je Schnittstelle per Software 8 Modes einstellen. Nur Mode 0 und 1 sind belegt, Mode 2 bis 7 sind reserviert. Damit kann je Kanal eingestellt werden, ob die Sende- und Empfangspegel "invertiert" oder "nicht invertiert" sein sollen:

Mode	**A3** **B3**	**A2** **bzw.** **B2**	**A1** **B1**	**Funktion**
0	0	0	0	Sende- und Empfangspegel "invertiert"
1	0	0	1	Sende- und Empfangspegel "nicht invertiert"

Bitte beachten Sie, daß die beiden SCC-Pins "TRxC/A" und "TRxC/B" im SCC als Eingänge konfiguriert werden müssen. Außerdem müssen die CTS-, DCD- und SYNC-Interrupts im SCC bei beiden Kanälen deaktiviert (disabled) sein.

Tabelle: Reihenfolge der Initialisierung für den SCC

Register	Data	Kommentar
1. Stufe: Betriebsarten und Konstanten definieren		
WR9	1100 0000	Hardware Reset
WR0	0000 00xx	Select Shift Mode (nur bei Z8030)
WR4	xxxx xxxx	Transmit/Receive Control: Asyn- oder Synchrone Betriebsart anwählen
WR1	0xx0 0x00	W/REQ anwählen (optional)
WR2	xxxx xxxx	Interrupt-Vektor programmieren
WR3	xxxx xxx0	Receiver Control, Bit D0 (Rx enable) muß hier auf 0 gesetzt werden.
WR5	xxxx 0xxx	Transmit Control, Bit D3 (Tx enable) muß hier auf 0 gesetzt werden.
WR6	xxxx xxxx	SYNC-Zeichen programmieren
WR7	xxxx xxxx	SYNC-Zeichen programmieren
WR9	000x 0xxx	Interrupt Control anwählen. Bit D3 (MIED) muß auf 0 gesetzt werden.
WR10	xxxx xxxx	Kontrollwort (optional)
WR11	xxxx xxxx	Clock Control
WR12	xxxx xxxx	Zeitkonstante, Lowbyte (optional)
WR13	xxxx xxxx	Zeitkonstante, Highbyte (optional)
WR14	xxxx xxx0	Kontrollwort, Bit 0 (BR enable) muß hier auf 0 gesetzt werden.
WR14	xxxS SSSS	Register kann mehrfach beschrieben werden, wenn mehr als ein Kommando geschickt werden soll
2. Stufe: Enables		
WR3	SSSS SSS1	Bit D0 (Rx Enable) auf 1 setzen.
WR5	SSSS 1SSS	Bit D3 (Tx Enable) auf 1 setzen
WR0	1000 0000	Reset TxCRC
WR14	000S SSS1	Baudraten Generator Enable, Bit D0 auf 1 setzen, Enable DPLL
WR1	xSS0 0S00	D7 auf 1 setzen, wenn DMA enabled werden soll.
Stufe 3: Interrupt Enable		
WR15	xxxx xxxx	Enable external interrupts
WR0	0001 0000	Reset EXT/STATUS zweimal
WR0	0001 0000	Reset EXT/STATUS zweimal
WR1	SSSx xSxx	Enable receive, transmit and external interrupt master.
WR9	000S xSSS	Enable Master Interrupt bit D3.

1 = Auf 1 setzen, 0 = auf 0 setzen, x = nach Wunsch setzen, S = so setzen wie bereits gesetzt

Arbeitsblatt:

Stufen	Register	Hex	Bit	Kommentar
1. Betriebsarten				
	WR9	_C_ _0_	1100 0000	Software Reset
	WR0	_0_ ___	0000 00__	_____
	WR4	___ ___	____ ____	_____
	WR1	___ ___	0__0 0_00	_____
	WR2	___ ___	____ ____	_____
	WR3	___ ___	____ ___0	_____
	WR5	___ ___	____ 0___	_____
	WR6	___ ___	____ ____	_____
	WR7	___ ___	____ ____	_____
	WR9	___ ___	000_ 0___	_____
	WR10	___ ___	____ ____	_____
	WR11	___ ___	____ ____	_____
	WR12	___ ___	____ ____	_____
	WR13	___ ___	____ ____	_____
	WR14	___ ___	____ ___0	_____
	WR14	___ ___	____ ___0	_____
2. Enables				
	WR3	___ ___	____ ___1	_____
	WR5	___ ___	____ 1___	_____
	WR0	_8_ _0_	1000 0000	Reset TxCRC
	WR14	___ ___	000_ ___1	_____
	WR1	___ ___	____ ____	_____
3. Interrupt				
	WR15	___ ___	____ ____	_____
	WR0	_1_ _0_	0001 0000	Reset Ext/Status
	WR0	_1_ _0_	0001 0000	Reset Ext/Status
	WR1	___ ___	____ ____	_____
	WR9	___ ___	000_ ____	_____

Initialisierungswerte nach Reset:

Register	**Hardware-Reset**	**Channel-Reset**
WR0	0000 0010	0000 0000
WR1	00_0 0_00	00_0 0_00
WR2	____ ____	____ ____
WR3	____ ___0	____ ___0
WR4	____ _1__	____ _1__
WR5	0__0 000_	0__0 0000
WR6	____ ____	____ ____
WR7	____ ____	____ ____
WR9	1100 00__	__0_ ____
WR10	0000 0000	0__0 0000
WR11	0000 1000	____ ____
WR12	____ ____	____ ____
WR13	____ ____	____ ____
WR14	__10 0000	__10 00__
WR15	1111 1000	1111 1000
RR0	01__ _100	01__ 100
RR1	0000 0111	0000 0111
RR3	0000 0000	0000 0000
RR10	0000 0000	0000 0000

Hochsprachenbibliothek

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M033_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Prozedur **m033_bib_startup** einmal aufgerufen werden.

m033_bib_startup — Initialisiere Modulbibliothek

Pascal PROCEDURE m033_bib_startup (micro_slot: byte);

C void EXPORT m033_bib_startup (byte micro_slot);

Funktion Diese Prozedur initialisiert die Modulbibliothek M033_LIB. Es werden u.a. die Initialisierungsdaten aus den EEPROMs aller M-COM-2/P Module übernommen, die sich auf der Basiskarte befinden. Die Register des Gate-Arrays werden gemäß den EEPROM-Inhalten gesetzt.

m033_scc_reset — Führe SCC-Hardware-Reset durch

Pascal PROCEDURE m033_scc_reset (micro_slot);

C void EXPORT m033_scc_reset (byte micro_slot);

Funktion Diese Prozedur führt einen Hardware-Reset des SCC durch.

m033_init_scc — Initialisiere SCC-Kanal

Pascal
FUNCTION m033_init_scc (micro_slot, channel, mode, intmode, int_x_mode: byte; timeconst: word; use_brg, clkmode, databits, stopbits, parity_type: byte): byte;

C
byte EXPORT m033_init_scc (byte micro_slot, byte channel, byte mode, byte intmode, byte int_x_mode, ushort timeconst, byte use_brg, byte clkmode, byte databits, byte stopbits, byte parity_type);

Funktion
Diese Funktion initialisiert einen Kanal des Moduls.

Parameter

channel: Nummer des Kanals (0 = A, 1 = B)

mode: Einstellung der Taktquellen für Receiver und Transmitter

= 0: Sende- und Empfangspegel „invertiert“

= 1: Sende- und Empfangspegel „nicht invertiert“

int_mode: Interrupt-Betriebsart festlegen

7	6	5	4	3	2	1	0	
								Interrupt Enable
0	0	0						Reserviert
			1	0				Interrupt On All Rx Character or Special Condition
			0	0				Rx Interrupt Disable
					1			Parity is Special Condition
						1		Tx Interrupt Enable
							1	Ext. Interrupt Enable

int_x_mode: Erweiterte Interrupt-Betriebsart festlegen

7	6	5	4	3	2	1	0	
								Ext./Status Interrupt Select
1								Break/Abort Interrupt Enable
	0	0	0	0	0	0	0	reserviert

timeconst: Zeitkonstante für Baudratengenerator (0 bis ffffh)

clkmode: Clock-Mode (1, 16, 32, 64) für Receiver und Transmitter

use_brg: Baudratengenerator verwenden (1) oder nicht (0)

databits: Anzahl Datenbits (5 bis 8)

stopbits: Anzahl Stopbits (0 = 0, 1 = 1, 2 = 1,5 3 = 2 Stopbits)

parity_type: Parität (0 = keine Parität, 1 = odd, 2 = even)

m033_define_intrp_connection Interruptanwahl

Pascal PROCEDURE m033_define_intrp_connection (micro_slot, intrp: byte);

C void EXPORT m033_define_intrp_connection (byte micro_slot, byte intrp);

Funktion Diese Prozedur wählt einen Interruptleitung zur Basiskarte an.

Parameter *intrp*: Nummer der Interrupt-Leitung (0 = kein Interrupt, 1 = IRQ-A, 2 = IRQ-F, 3 = IRQ-B, 4 = IRQ-E, 5 = IRQ-C und 6 = IRQ-D)

m033_transmit_scc_character Sende Zeichen

Pascal FUNCTION m033_transmit_scc_character (micro_slot, channel, data: byte): byte;

C byte EXPORT m033_transmit_scc_character (byte micro_slot, byte channel, byte data);

Funktion Diese Funktion sendet ein Zeichen. Vor dem Senden wird der Zustand des Sendepuffers abgefragt. Ist der Sendepuffer voll (TBE = 0), so wird 1 zurückgeliefert. Wurde das Zeichen erfolgreich in den Sendepuffer geschrieben, liefert die Funktion 0 zurück.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

data: Zu sendendes Datenbyte

m033_receive_scc_character — Empfange Zeichen

Pascal: FUNCTION m033_receive_scc_character (micro_slot, channel: byte; var data: byte): byte;

C: byte EXPORT m033_receive_scc_character (byte micro_slot, byte channel, byte *data);

Funktion: Diese Funktion liest den Empfangspuffer eines Kanals. Vor dem Empfangen wird der Zustand des Empfangspuffers abgefragt. Ist dieser leer (RBF = 0), so wird 1 zurückgeliefert. Wurde ein Zeichen empfangen, liefert die Funktion 0 zurück.

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

data: Empfangenes Datenbyte

m033_scc_rbf — Frage Empfangspuffer-Status (RBF) ab

Pascal: FUNCTION m033_scc_rbf (micro_slot, channel: byte): byte;

C: byte EXPORT m033_scc_rbf (byte micro_slot, byte channel);

Funktion: Diese Funktion prüft den Status des Empfangspuffers eines Kanals. Der Zustand des RBF-Bits wird zurückgegeben (RBF = 1: receive buffer full, RBF = 0 receive buffer empty).

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

m033_scc_tbe — Frage Sendepuffer-Status (TBE) ab

Pascal: FUNCTION m033_scc_tbe (micro_slot, channel: byte): byte;

C: byte EXPORT m033_scc_tbe (byte micro_slot, byte channel);

Funktion: Diese Funktion prüft den Status des Sendepuffers eines Kanals. Der Zustand des TBE-Bits wird zurückgegeben (TBE = 1: transmit buffer empty, TBE = 0 transmit buffer full).

Parameter: *channel*: Nummer des Kanals (0 = A, 1 = B)

m033_scc_cts — Frage Clear To Send (CTS) ab

Pascal FUNCTION m033_scc_cts (micro_slot, channel: byte): byte;

C byte EXPORT m033_scc_cts (byte micro_slot, byte channel);

Funktion Diese Funktion liefert den aktuellen Zustand der CTS-Leitung eines Kanals.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

m033_set_rts — Setze Request To Send (RTS)

Pascal PROCEDURE m033_set_rts (micro_slot, channel, rts: byte);

C void EXPORT m033_set_rts (byte micro_slot, byte channel, byte rts);

Funktion Diese Prozedur setzt die RTS-Leitung eines Kanals auf den angegebenen Wert.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

rts: Zu setzender Wert (0 oder 1)

m033_scc_intvec — Lies Interrupt-Vektor

Pascal PROCEDURE m033_scc_intvec (micro_slot: byte; var int_vector, rr0, rr1, rr3: byte);

C void EXPORT m033_scc_intvec (byte micro_slot, byte *int_vector, byte *rr0, byte *rr1, byte *rr3);

Funktion Diese Funktion kann dazu verwendet werden, nach einer SCC-Interrupt-Anforderung, den Interrupt-Vektor zu lesen.

Parameter *vector*: Interrupt-Vektor

Bit 3 bis 1 (V3, V2 und V1) enthalten weitere Informationen über den aufgetretenen Interrupt. Die restlichen Bits sind reserviert.

rr0: Aktueller Zustand von Read-Register 0

7	6	5	4	3	2	1	0	
								Read Register 0
	■	■	■	■	■	■	■	Break/Abort
■		■	■	■	■	■	■	Tx Underun/EOM
■	■		■	■	■	■	■	CTS
■	■	■		■	■	■	■	Sync/Hunt
■	■	■	■		■	■	■	DCD
■	■	■	■	■		■	■	Tx Buffer Empty
■	■	■	■	■	■		■	Zero Count
■	■	■	■	■	■	■		Rx Character Available

rr1: Aktueller Zustand von Read-Register 1

7	6	5	4	3	2	1	0	
								Read Register 1
	■	■	■	■	■	■	■	End of Frame
■		■	■	■	■	■	■	CRC/Framing Error
■	■		■	■	■	■	■	Rx Overrun Error
■	■	■		■	■	■	■	Parity Error
■	■	■	■		■	■	■	Residue Code 0
■	■	■	■	■		■	■	Residue Code 1
■	■	■	■	■	■		■	Residue Code 2
■	■	■	■	■	■	■		All sent

rr3: Aktueller Zustand von Read-Register 3

7	6	5	4	3	2	1	0	
□	□	□	□	□	□	□	□	Read Register 3 (Interrupt Pending (IP) Register)
□	■	■	■	■	■	■	■	0
■	□	■	■	■	■	■	■	0
■	■	□	■	■	■	■	■	Channel A Rx IP
■	■	■	□	■	■	■	■	Channel A Tx IP
■	■	■	■	□	■	■	■	Channel A Ext/Status IP
■	■	■	■	■	□	■	■	Channel B Rx IP
■	■	■	■	■	■	□	■	Channel B Tx IP
■	■	■	■	■	■	■	□	Channel B Ext/Status IP

m033_baudrate_calc Ermittle Einstellungen für Baudrate

Pascal
PROCEDURE m033_baudrate_calc (baudrate, quartz: longint;
var bauderror: longint; var basereg: byte; var clk_mode: byte;
var timeconst: word, var use_brg: byte);

C
void EXPORT m033_baudrate_calc (long baudrate, long quartz,
long *bauderror, byte *basereg, byte *clk_mode, ushort *timeconst,
byte *use_brg);

Funktion
Diese Prozedur ermittelt die notwendigen Einstellungen für eine gewünschte Baudrate bei einer bestimmten Frequenz des Quarzoszillators. Als Ergebnisse liefert die Prozedur die SCC-Parameter für den Baudratengenerator.

Parameter

baudrate: Gewünschte Baudrate in bd

quartz: Taktfrequenz des Quarzoszillators in Hz (Standard: 7,3728 MHz)

bauderror: Absoluter Fehler des Basistaktes bzw. der Baudrate der sich bei dieser Einstellung ergibt

basereg: Reserviert

timeconst: Einstellung für die Zeitkonstante des Baudratengenerators

clk_mode: Einstellung für Clock-Mode des Receivers/Transmitters

use_brg: Verwendung des Baudratengenerators (0 = nein, 1 = ja)

m033_set_channel — Konfiguriere Kanal

Pascal
FUNCTION m033_set_channel (micro_slot, channel: byte; baudrate: word; flag, scc_mode, scc_intmode, scc_int_x_mode, databits, stopbits, parity_type: byte; var bauderror: longint; var used_clkmode: byte): byte);

C
byte EXPORT m033_set_channel (byte micro_slot, byte channel, long baudrate, byte flag, byte mode, byte intmode, byte int_x_mode, byte databits, byte stopbits, byte parity_type, long *bauderror, byte *used_clkmode);

Funktion
Diese Funktion stellt eine Kombination aus **m033_baudrate_calc** und **m033_init_scc** dar (siehe auch Parameter dieser Funktionen).

Parameter
flag: Reserviert

Programmierung mit I/O-Zugriffen

Lokale I/O-Adressen

Adresse	Zugr.	Funktion							
MBA+00h	RW8	SCC: Kanal B, Control							
MBA+01h	RW8	Kanal B, Data							
MBA+02h	RW8	Kanal A, Control							
MBA+03h	RW8	Kanal A, Data							
MBA+07h	W8	Interrupt anwählen: Port C1 [1]							
MBA+08h	W8	Port C2 [1]							
MBA+09h	W8	Port C3 [1]							
		Interrupt:	Keiner	IRQ-A	IRQ-B	IRQ-C	IRQ-D	IRQ-E	IRQ-F
		C3, C2, C1:	000	010	100	011	101	111	110
MBA+0ah	W8	Konfiguration Schnittstelle A: Port A1 [1]							
MBA+0bh	W8	Port A2 [1]							
MBA+0ch	W8	Port A3 [1]							
MBA+0dh	W8	Konfiguration Schnittstelle B: Port B1 [1]							
MBA+0eh	W8	Port B2 [1]							
MBA+0fh	W8	Port B3 [1]							
MBA+0ch	R8	Status lesen: Port C1 [2, 3]							

[1] Beim Schreiben wird der Port entsprechend Bit 0 gesetzt, Bit 1 bis Bit 7 sind ohne Bedeutung.

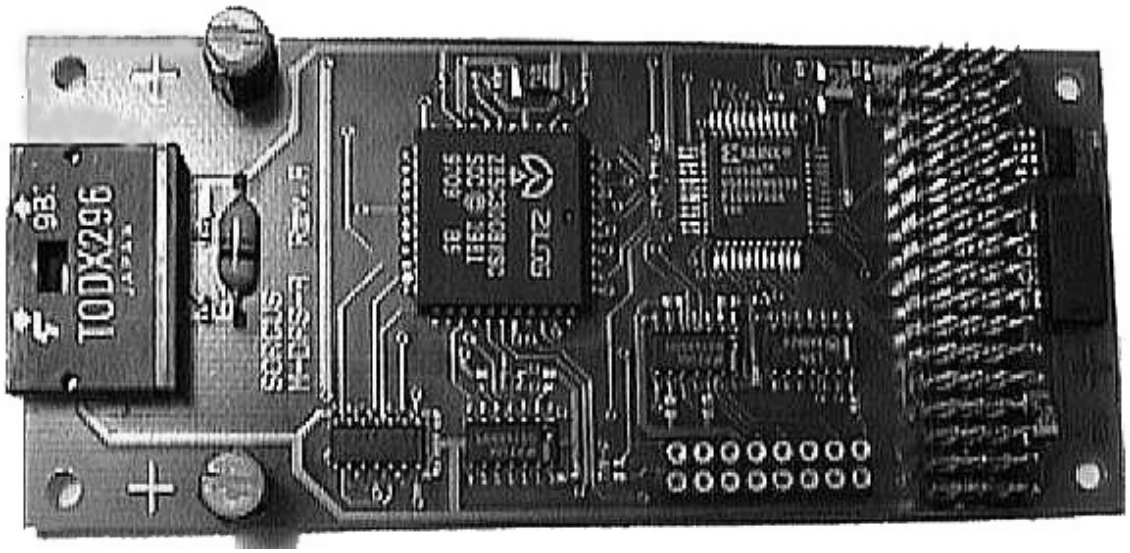
[2] Beim Lesen steht das Ergebnis in Bit 0, die anderen Bit sind ungültig.

[3] Nur für Testzwecke.

Notizen

4. M-DAS-A

Zwei universelle serielle Schnittstellen



Funktionsbeschreibung

Diese Beschreibung bezieht sich auf Rev. A des Moduls M-DAS-A (Kanal A mit Lichtwellenleiter-Interface System Toshiba, Kanal B mit zusätzlichem CLK-Input und CLK-Output, entsprechend den C-Link-Adaptern CL232A/i und CL232A/o beim Modul M-COM-2).

Das Modul M-DAS-A enthält zwei serielle synchrone/asynchrone Schnittstellen. Es ist für die Basiskarte MODULAR-4/486 in zwei Bestückungsvarianten lieferbar (siehe folgende Tabelle). Als Standardbestückung ist das Modul mit einem Quarzoszillator von 7,3728 MHz ausgerüstet, alle Bestückungsvarianten können auf Wunsch auch mit einer anderen Quarzfrequenz, z. B. 4,9152 MHz geliefert werden. Entsprechend der verwendeten Basiskarte und dem Typ der LWL (Glas oder Plastik) sind außerdem folgende Versionen lieferbar (der Vollständigkeit halber sind auch die beiden Universalversionen des Moduls M-COM-2 für den Einsatz mit C-Link Adaptern aufgeführt):

***Tabelle 10-1**:* Übersicht M-COM-2 und M-DAS-A Module

Modulversion	**Typ**	**für Basiskarte**	**Bestückungsvariante**
M-COM-2/P/8	33	MODULAR-4/486	Plastik-LWL für beide Kanäle
M-COM-2/G/8	33	MODULAR-4/486	Glas-LWL für beide Kanäle
M-COM-2/8	32	MODULAR-4/486	C-Link-Adapter je Kanal
M-DAS-A/G	50	MODULAR-4/486	Glas-LWL für Kanal A RS-232 mit zus. CLK_{in} und CLK_{out} für Kanal B
M-DAS-A/P	50	MODULAR-4/486	Plastik-LWL für Kanal A RS-232 mit zus. CLK_{in} und CLK_{out} für Kanal B

Tabelle 10-2: Übersicht C-Link-Adapter für Modul M-COM-2

C-Link-Adapter	**Physikalische Schnittstelle**	**Kurzbeschreibung**
CL232S	RS-232 bis 120 kBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS, DTR, DSR, RI, DCD Zusätzliche Funktionen: Mode 0: RI als Clock-Eingang Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang
CL232A/i	RS-232 bis 120 kBaud	Modem-Steuerleitungen (in Mode 0): TMT, RCV, RTS, CTS, DTR, DSR, RI, DCD Zusätzliche Funktionen: Mode 0: Zusätzliche RS-232-Leitung EXT als Clock-Eingang 1 [1] Mode 0: RI als Clock-Eingang 2
CL232A/o	RS-232 bis 120 kBaud	Modem-Steuerleitungen (in Mode 5): TMT, RCV, RTS, CTS, DTR, DSR, RI, DCD Zusätzliche Funktionen: Mode 5: Zusätzliche RS-232-Leitung EXT als Clock-Ausgang [1]
CL232i	RS-232 isol. bis 120 kBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang
CL422S	RS-422 bis 10 MBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang

[1] EXT ist kein RS-232-Standardsignal. Es ist auch bei den üblichen 9-poligen oder 25-poligen D-Submin.-Steckern bzw. Buchsen nicht vorhanden. Die 9-polige Schnittstelle wurde hier um einen Pin erweitert, um bei RS-232 alle Modem-Steuersignale und zusätzlich einen Taktein- oder -ausgang zu ermöglichen. Bei jenen C-Links, die diesen Pin verwenden, kann er, wenn das entsprechende Signal nicht benötigt wird, frei bleiben (= nicht angeschlossen).

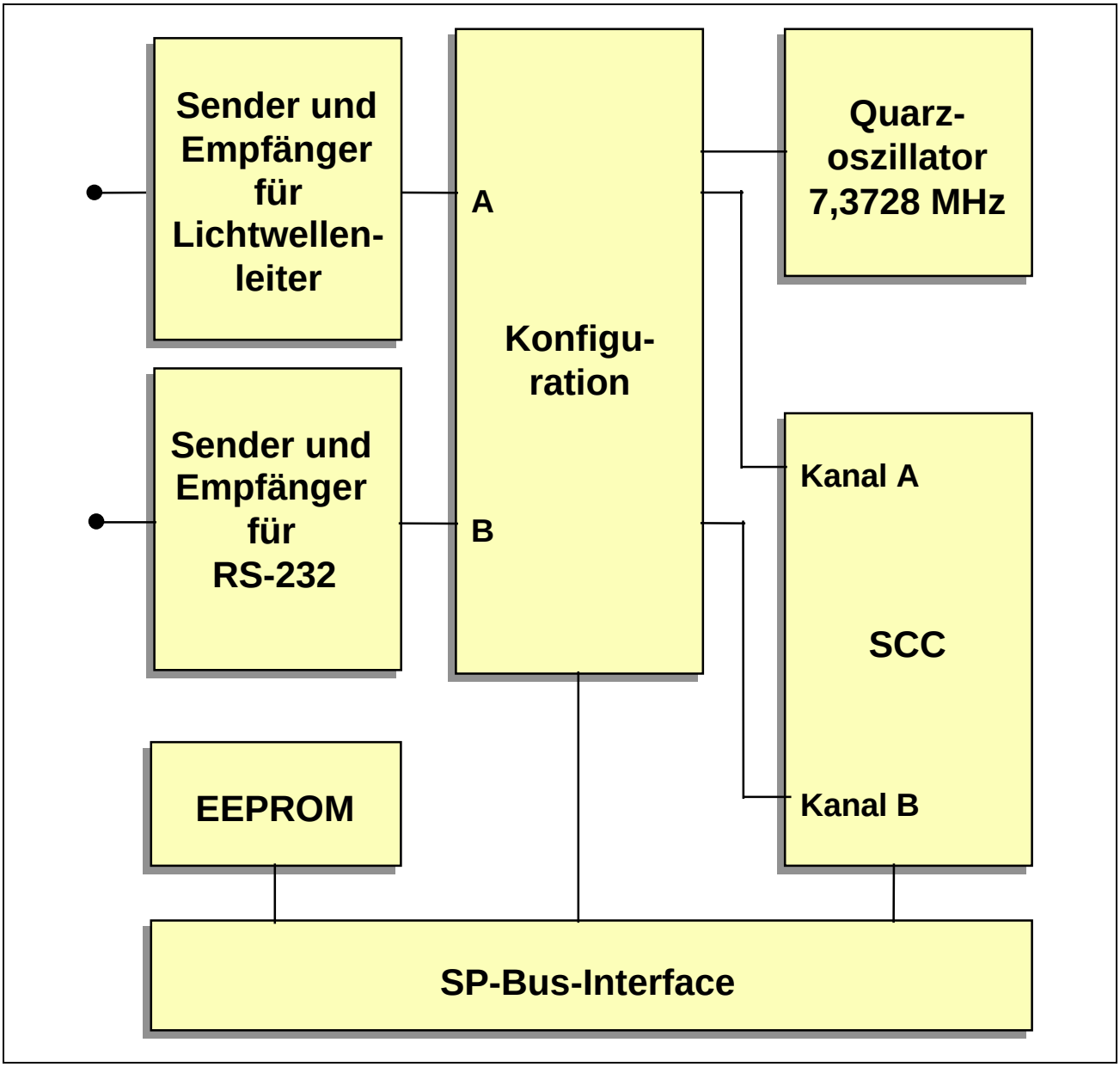
C-Link-Adapter	Physikalische Schnittstelle	Kurzbeschreibung
CL422S	RS-485 bis 10 MBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 0 und 2: RTS-Treiber disabled Mode 2: CTS als Clock-Eingang Mode 3: CTS als Clock-Eingang, RTS enabled Mode 5: RTS als Clock-Ausgang
CL422i	RS-422 isol. bis 10 MBaud	Modem-Steuerleitungen (in Mode 1): TMT, RCV, RTS, CTS Zusätzliche Funktionen: Mode 3: CTS als Clock-Eingang Mode 5: RTS als Clock-Ausgang
CL485i/U	RS-485 isol. bis 20 MBaud	Bis 20 MBaud geeignet, Umschaltung von Senden auf Empfangen per Software oder automatisch (z. B. für SDLC/HDLC)
CL200A	20 mA isol. bis 120 kBaud	20 mA Current Loop, zwei Konstantstromquellen auf dem C-Link. Passiv oder aktiv konfigurierbar (wenn passiv, dann galvanisch getrennt)

Das Modul M-DAS-A

Die beiden seriellen Schnittstellen des Moduls M-DAS-A sind unabhängig voneinander und unterschiedlich konfigurierbar. Sie sind mit einem SCC-Baustein 85C30 oder optional Z85230 (= verbesserte und erweiterte Version mit größeren FIFOs) ausgerüstet. Außerdem enthält das Modul einen eigenen Quarzoszillator und zwei Baudratengeneratoren, ist also unabhängig vom CPU-Takt oder von Timern der Basiskarte. Für besondere Einsatzbedingungen ist das Modul auch so konfigurierbar, daß die Baudraten für Senden und Empfangen bei einem Kanal unterschiedlich eingestellt werden können. Außerdem kann je nach Konfiguration der Sende- und/oder Empfangstakt von außen bzw. ein Takt nach außen geliefert werden. Die Richtung ist per Software umschaltbar.

Das Modul ist interruptfähig, der Interrupt vom Modul zur Basiskarte kann per Software angewählt und an IRQ-A bis IRQ-F der Basiskarte gelegt werden.

Blockschaltbild



Technische Daten

Parameter	Wert	Einheit
Anzahl serieller Schnittstellen	2	-
Serieller Kommunikationsbaustein (Option:	Z85C30-8 Z85230-20)	-
Interruptfähig zur Basiskarte [1]	ja	-
Modem-Steuerleitungen Kanal A	Keine	-
Modem-Steuerleitungen Kanal B	RTS, CTS, DCD, DTR, Ri, DSR, EXTI[2], EXTO[2], CLK_{in}, CLK_{out}	
Baudratengeneratoren Quarzoszillator: 7,3728 MHz (Option: 4,9152 MHz)	2	-
Versorgungsspannungen:	+5 +12 -12	V V V
Stromaufnahme +5/+12/-12 Volt (typ.):	130/15/14	mA
Betriebstemperatur	0 bis 70	°C
Abmessungen (L x B x H)	106 x 45 x 15	mm

Lieferumfang

- Modul M-DAS-A
- 20-poliger Pfostenstecker für Flachbandkabel
- Datenträger und Programmbibliotheken (Pascal und C)

---

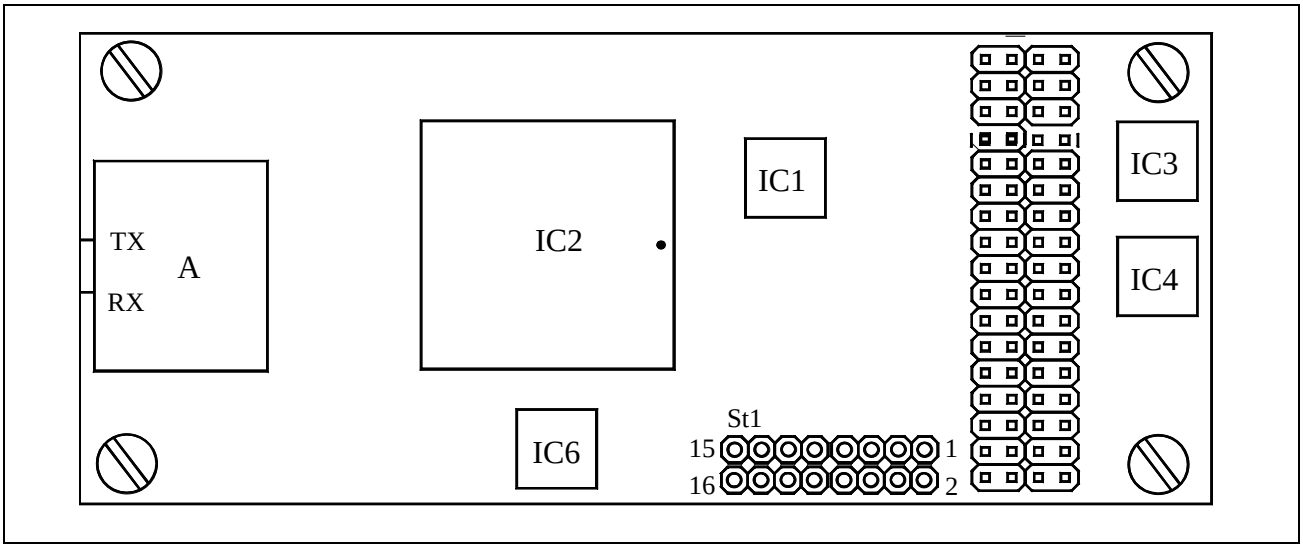
[1] Per Software ist einer von 6 Interrupt-Eingängen der Basiskarte wählbar. Der Interrupt-Ausgang des Moduls ist aktiv Low. Da die Interrupt-Eingänge der Basiskarte flankengesteuert sind, muß der entsprechende Interrupt-Eingang also auf der Basiskarte auf negative Flanke programmiert werden.

[2] Erklärung zum Signal EXTI, EXTO siehe unten bei der Beschreibung der Konfiguration

Konfiguration und Einbau

Das Modul enthält keine Jumper, alle Einstellungen werden nach dem Einbau per Software vorgenommen.

Lageplan, Rev. A



EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0001	0011 0010	2132h	Modultyp M-DAS-A
1	0000 0000	0000 0001	0001h	Initialisierung
2	0000 0001	0001 0000	0110h	Bestückung
3	0000 0111	0011 0111	0737h	Quarzfrequenz
4	0000 0100	0000 0100	0404h	Bestückung und Umbau (Kanal A)
5	0000 0101	0000 0100	0504h	Bestückung und Umbau (Kanal B)
6	0000 0000	0000 1010	000ah	Physikalisches Interface (Kanal A)
7	0000 0000	0000 1110	000eh	Physikalisches Interface (Kanal B)
8	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	1	0	0	1	0	WORT-0: Kennung
								0	0	1	1	0	0	1	0	Modultyp: 50 = M-DAS-A
				0	0	0	1									Revision: 1 = A, 2 = B, 3 = C, ...
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort kann eingestellt werden, ob das Modul nach dem Einschalten und bei einem Hardware-Reset entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															1	Init nach Hardware-Reset: 0 = nein, 1 = ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-2: Bestückung

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	WORT-2: Bestückung (werks. Einst.)
																geändert am: von:
												0	0	0	0	Typ SCC: 0 = SCC (85C30) 1 = ESCC (85130) 2 = ESCC (85230)
								0	0	0	1					Max. Takt (SCC): 0 = 6 MHz 1 = 8 MHz 2 = 10 MHz 3 = 16 MHz 4 = 20 MHz
				0	0	0	1									FPGA-Version IC1: 0 = unbekannt 1 = Vers. A ("MDASA1A") 2 = Vers. B ("MDASA1B")
0	0	0	0													Reserviert

M-DAS-A

WORT-3: Quarzfrequenz

Die Angabe erfolgt mit 4 Ziffern in MHz mit 2 Vorkomma- und 2 Nachkommastellen. 4,9152 MHz würde also aufgerundet auf 4,92 und mit 4 Dezimalziffern 0492 (= 0492h) eingegeben. 0000h bedeutet, daß der Quarzoszillator nicht bestückt ist, die folgende Angabe 0737h bedeutet 7,37 MHz.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	1	WORT-3: Quarzfrequenz (werks. Einst.)
																geändert am: von:
												0	1	1	1	2. Ziffer nach Komma
								0	0	1	1					1. Ziffer nach Komma
				0	1	1	1									2. Ziffer
0	0	0	0													1. Ziffer

WORT-4 und WORT-5: Bestückung und Umbau

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	WORT-4: Hardware Kanal A (werks. Einst.)
																geändert am: von:
															0	1 = C-Link möglich 0 = nicht vorgesehen
														0		1 = 24-poliger Sockel 0 = nicht vorbereitet
													1			1 = C-Link eingelötet 0 = nicht eingelötet
								0	0	0	0	0				Reserviert
							0									1 = Interface möglich 0 = nicht vorgesehen
						0										1 = Sockel 0 = nicht vorbereitet
					1											1 = fest eingelötet 0 = nicht eingelötet
				0												1 = Modul umgebaut 0 = nicht umgebaut
0	0	0	0													Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	WORT-5: Hardware Kanal B (werks. Einst.)
																geändert am: von:
															0	1 = C-Link möglich 0 = nicht vorgesehen
														0		1 = 24-poliger Sockel 0 = nicht vorbereitet
													1			1 = C-Link eingelötet 0 = nicht eingelötet
								0	0	0	0	0				Reserviert
							1									1 = Interface möglich 0 = nicht vorgesehen
						0										1 = Sockel 0 = nicht vorbereitet
					1											1 = fest eingelötet 0 = nicht eingelötet
0	0	0	0	0												Reserviert

WORT-6 und WORT-7: Physikalisches Interface (Kanal A und B)

In WORT-6 steht ein Code für das physikalische Interface für Kanal A, in WORT-7 für Kanal B (siehe auch WORT-4 und WORT-5).

Code		Physikalisches Interface	isol.	z. B. C-Link
0	(00h)	keines	-	-
1	(01h)	RS-232, mit allen Modem-Leitungen	nein	CL232S
2	(02h)	RS-232, nur 5 V Versorgung	nein	CL232V
3	(03h)	RS-232	nein	CL232A
4	(04h)	RS-232 (RCV, TMT, RTS/CLK_{out}, CTS/CLK_{in})	ja	CL232i
5	(05h)	20 mA	ja	CL200A
6	(06h)	RS-422 / RS-485	nein	CL422S
7	(07h)	RS-422 / RS-485	ja	CL422i
8	(08h)	Lichtleiter HP-System, seitlich	ja	CL800Q
9	(09h)	Lichtleiter HP-System, oben	ja	CL800L
10	(0ah)	Lichtleiter Toshiba Glas PCS (TODX296)	ja	CL800Q
11	(0bh)	Lichtleiter Toshiba Plastik APF (TODX297)	ja	CL800L
12	(0ch)	RS-232, Pin EXTI als CLK_{in}	nein	CL232A/i
13	(0dh)	RS-232, Pin EXTO als CLK_{out}	nein	CL232A/o
14	(0eh)	RS-232, mit EXTO als CLK_{out} und EXTI als CLK_{in}	nein	-

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	WORT-6: Interface Kanal A (werks. Einst.)
																geändert am: von:
								0	0	0	0	1	0	1	0	Code für Interface A
0	0	0	0	0	0	0	0									Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	WORT-7: Interface Kanal B (werks. Einst.)
																geändert am: von:
								0	0	0	0	1	1	1	0	Code für Interface B
0	0	0	0	0	0	0	0									Reserviert

Konfiguration und Steckerbelegung

Übersicht: Konfigurationsmöglichkeiten bei M-DAS-A (Rev. A) mit FPGA "MDASA1A"

Drei Ports, d.h. 3 Bit stehen zur Konfiguration für jede Schnittstelle zur Verfügung: Port A1, A2 und A3 für Schnittstelle A, Port B1, B2 und B3 für Schnittstelle B. Damit lassen sich je Schnittstelle per Software 8 Modes einstellen (Mode 0, 1, 2, 3, 4, 5 und 7 sind belegt, Mode 6 ist reserviert). Sie passen das Modul M-DAS-A an die jeweilig gewünschte Konfiguration an.

Abb. 4-1 zeigt die prinzipielle Verschaltung von SCC und FPGA für Kanal B des Moduls M-DAS-A (Rev. A). In allen folgenden Abbildungen bedeutet: I := Eingang, O := Ausgang.

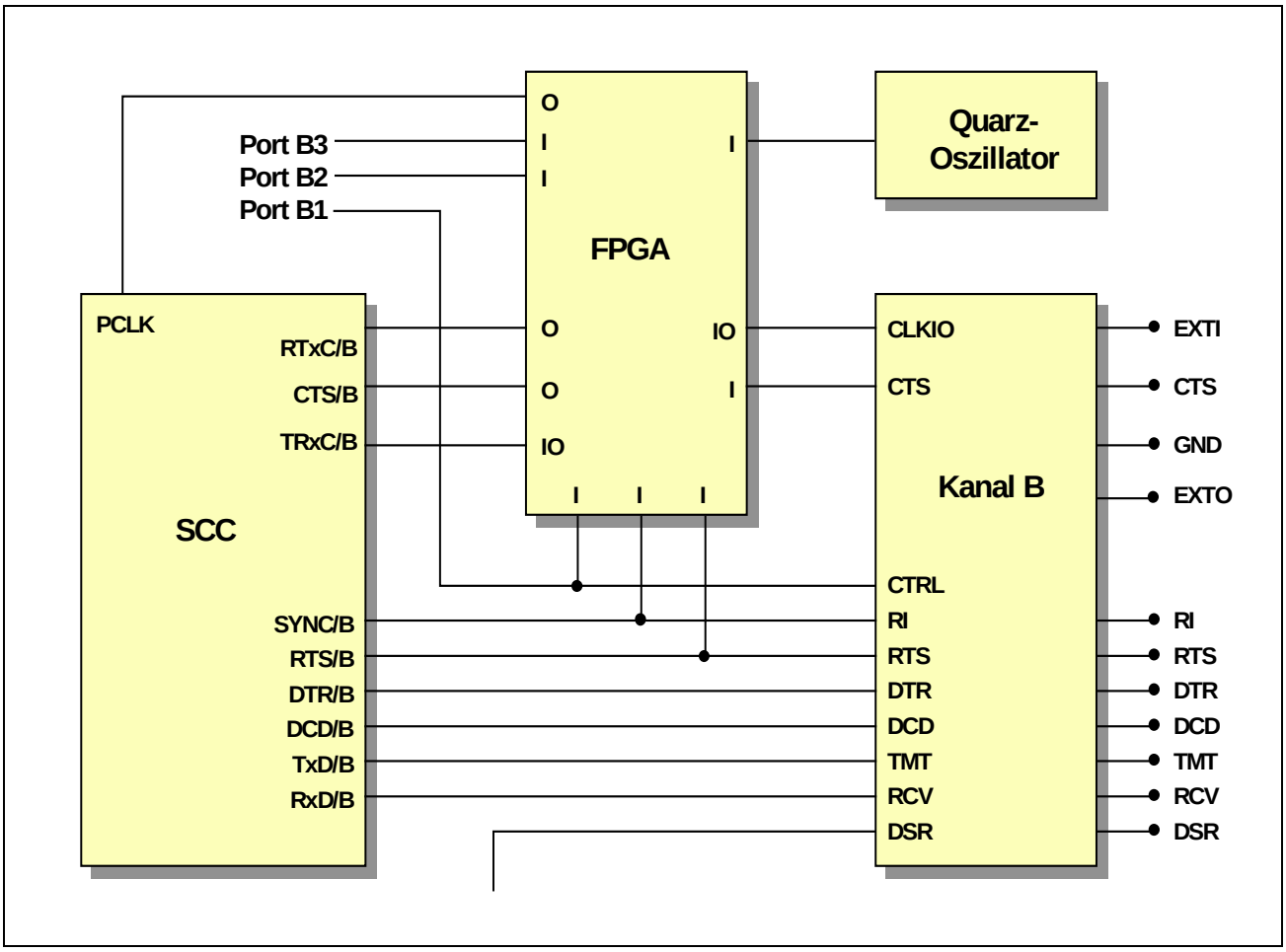


***Abb. 4-1**: Prinzipschaltbild der Konfigurationsmöglichkeiten für Schnittstelle B des Moduls M-DAS-A (Rev. A). Quarzoszillator und PCLK des SCC werden von beiden Kanälen genutzt. Vereinfachte Darstellung ohne die Konfigurationsmöglichkeiten im SCC. Die DSR-Leitung kann keinen Interrupt auslösen, der Status dieser Leitung kann aber gelesen werden (Logik hier nicht gezeigt).*

Anschlußstecker St1

Das Modul M-DAS-A wird über einen LWL-Anschluß (Kanal A) und einen 16-poligen Pfostensteckverbinder St1 und ein entsprechendes Flachbandkabel mit der Außenwelt verbunden (Kanal B).

Kanal A: LWL-Anschluß

Über das FPGA kann für diesen Kanal eingestellt werden, ob die Sende- und Empfangspegel "invertiert" oder "nicht invertiert" sein sollen:

Mode	A3	A2	A1	Funktion
0	0	0	0	Sende- und Empfangspegel "invertiert"
1	0	0	1	Sende- und Empfangspegel "nicht invertiert"

Bitte beachten Sie, daß der SCC-Pin "TRxC/A" im SCC als Eingang konfiguriert werden muß. Außerdem müssen die CTS-, DCD- und SYNC-Interrupts im SCC bei diesem Kanal deaktiviert (disabled) sein.

Kanal B: RS-232 mit zusätzlichem CLK-Input (Mode 0 und Mode 4) und zusätzlichem CLK-Output (Mode 5)

Für die Konfiguration mit zusätzlichem CLK-Input auf dem Modul M-DAS-A muß Mode 0 (d. h. für Kanal B: Port B3=0, B2=0, B1=0) oder Mode 4 eingestellt und im SCC der zugehörige TRxC Pin als Eingang konfiguriert werden.

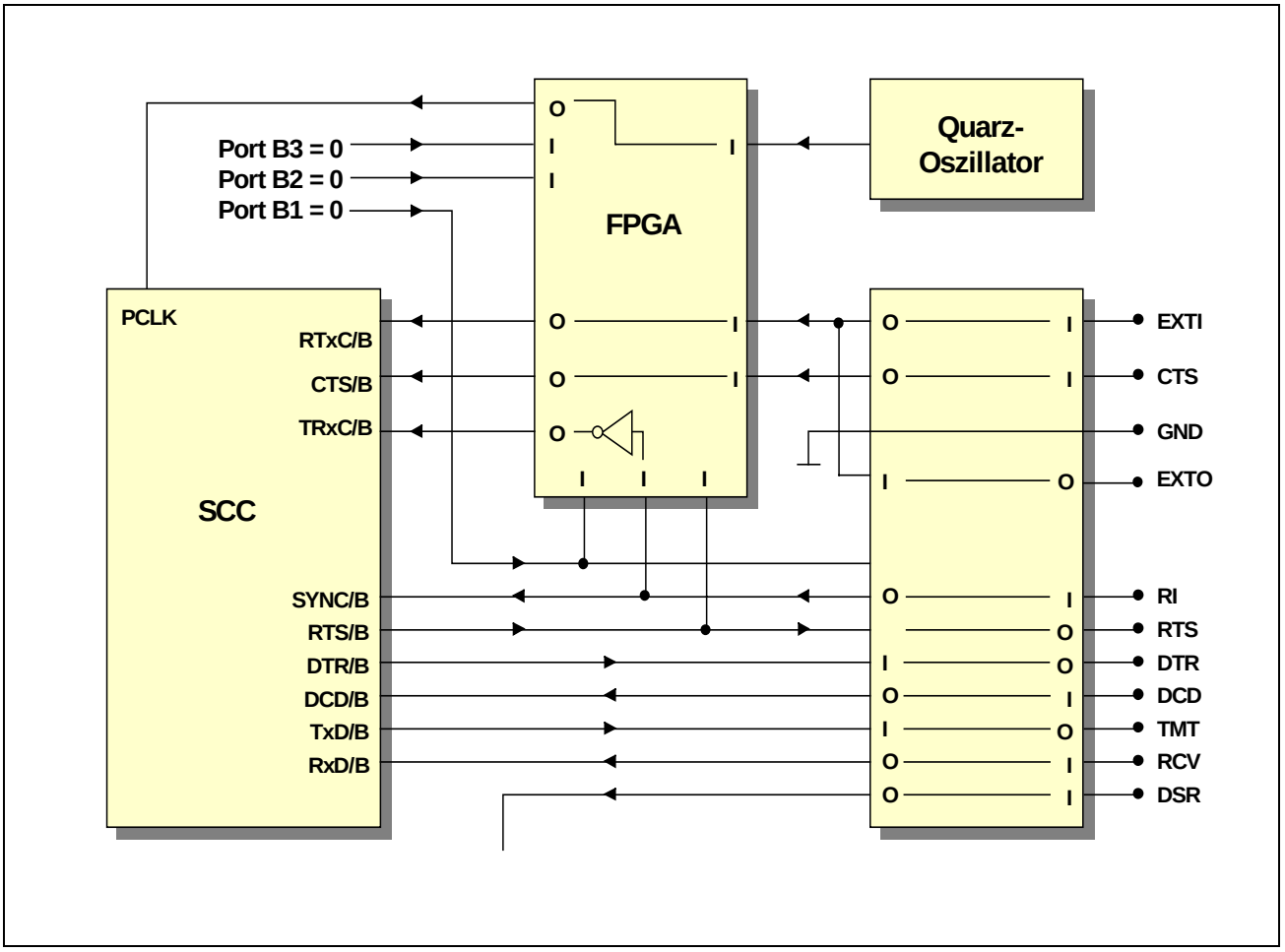


***Abb. 4-2**: Verschaltung bei Mode 0 von Kanal B des Moduls M-DAS-A (Rev. A).*

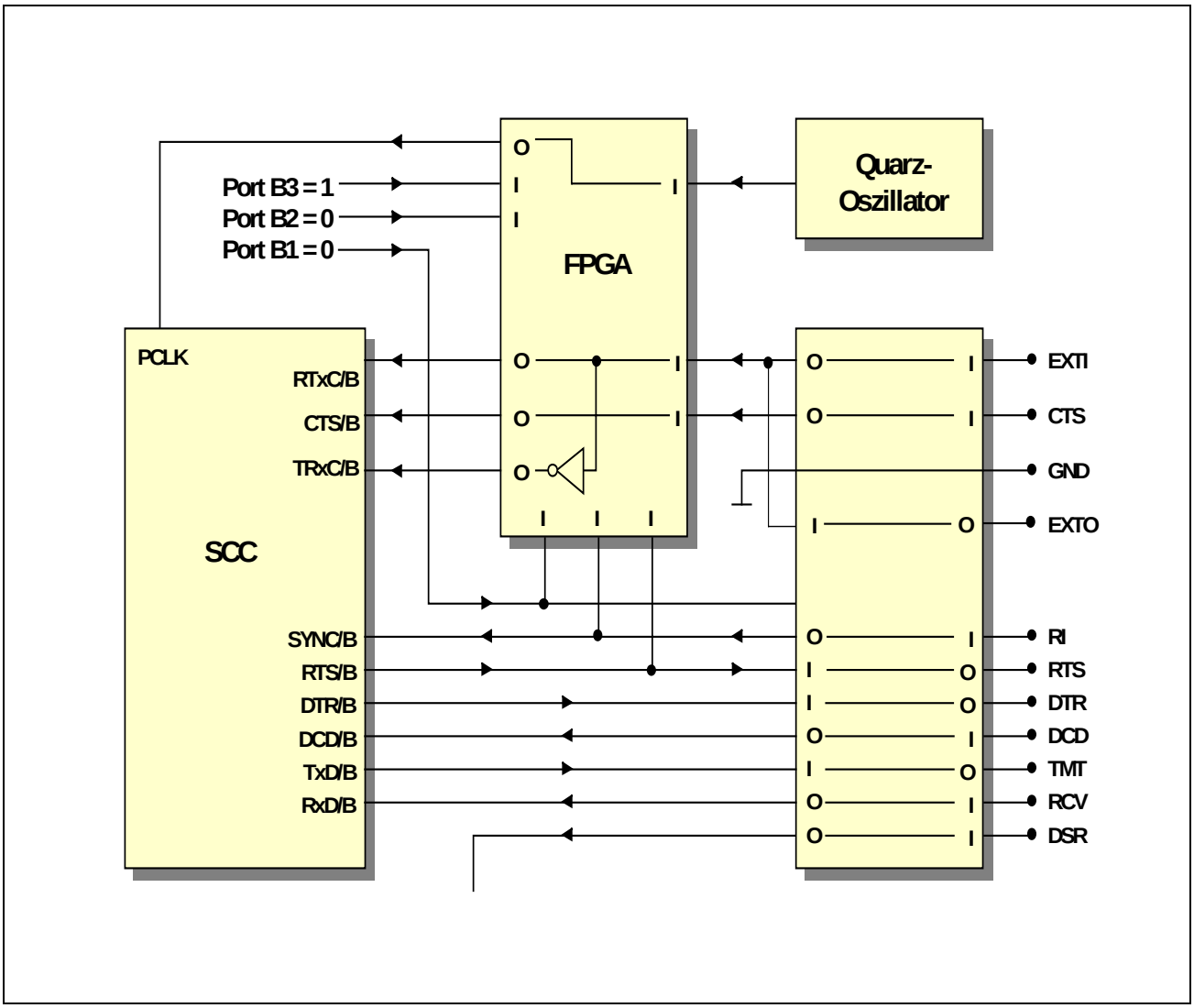


***Abb. 4-3**: Verschaltung bei Mode 4 von Kanal B des Moduls M-DAS-A (Rev. A)*

Für die Konfiguration mit zusätzlichem CLK-Output auf dem Modul M-DAS-A sollte Mode 5 (d.h. für Kanal B: Port B3=1, B2=0, B1=1) eingestellt und im SCC der zugehörige TRxC Pin als Ausgang konfiguriert werden.

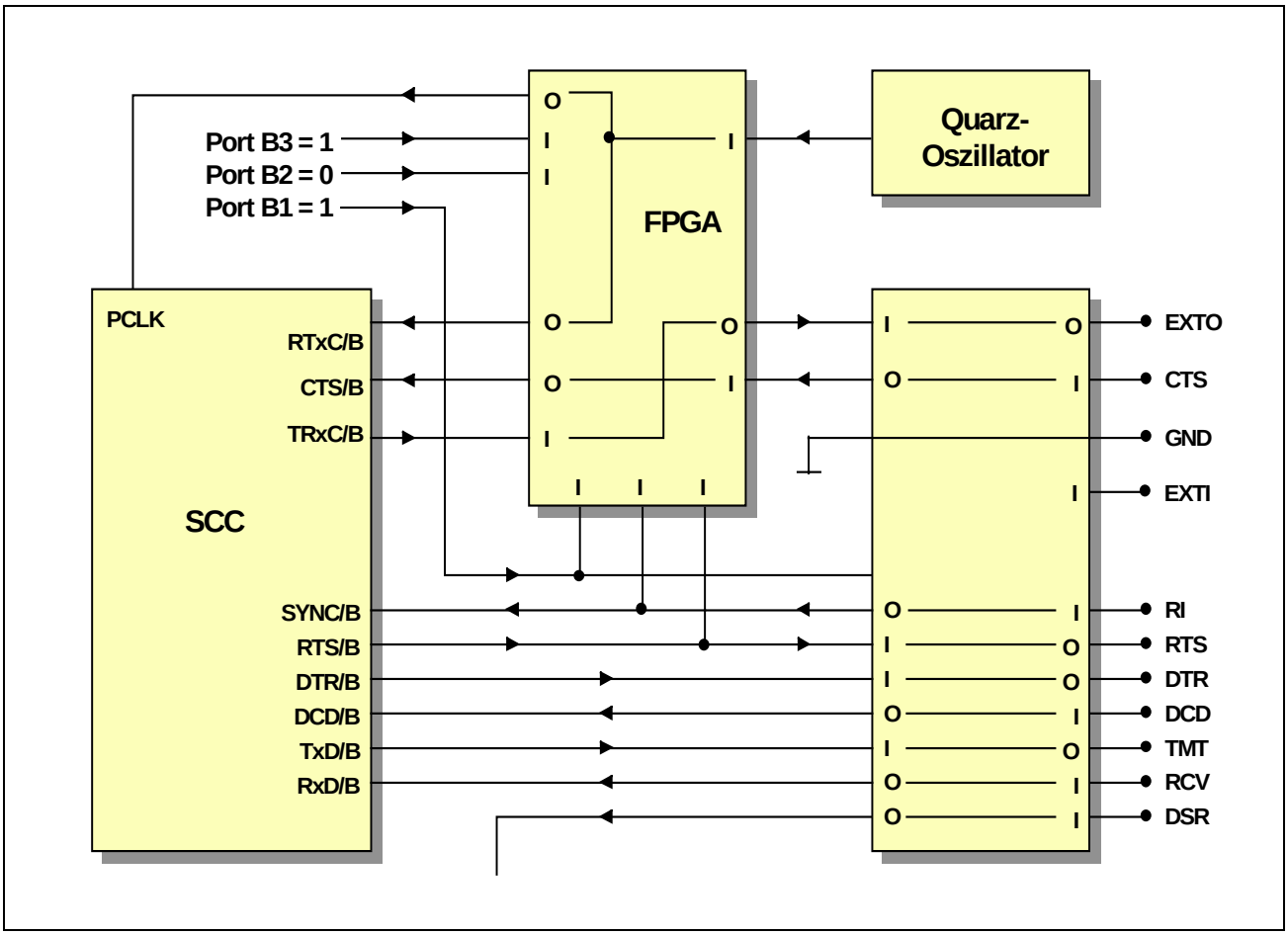


***Abb. 4-4**: Verschaltung bei Mode 5 von Kanal A des Moduls M-DAS-A (Rev. A).*

Die RS-232-Schnittstelle weist folgende Besonderheiten auf:

1. Über den RS-232-Anschluß EXTI (Eingang) kann ein Takt an Pin RTxC/B des SCC gelegt werden. Intern im SCC kann dieser dann als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden.

2. Über den RS-232-Anschluß EXTO (Ausgang) kann dann das Signal am TRxC/B Pin dieses Kanals des SCC nach außen geliefert werden. Durch Programmierung des SCC wird festgelegt, ob das der Empfangstakt, der Sendetakt, der Ausgang der DPLL oder der Ausgang des Baudratengenerators dieses Kanals ist.

3. Der Takt des Quarzoszillators liegt beim Mode 5 auch direkt am RTxC/B Pin dieses Kanals des SCC. Intern im SCC kann er als Empfangstakt, als Sendetakt, als Eingangstakt für die DPLL und/oder als Eingangstakt für den Baudratengenerator konfiguriert werden. Wenn er als Empfangs- und/oder Sendetakt verwendet wird, kann so bei asynchronem Betrieb die höchstmögliche Baudrate eingestellt werden, z. B. 460,8 KBaud mit einem Quarz von 7,3728 MHz bei Clock Mode x16.

4. Der RS-232-Anschluß RI kann entweder als Modem-Steuerleitung RI dienen (an den zugehörigen SYNC/B Pin des SCC und damit interruptfähig), oder es kann darüber ein weiterer Takt von außen an Pin TRxC/B des SCC gelegt werden. Intern im SCC kann dieser dann als Empfangs- und/oder Sendetakt konfiguriert werden.

***Tabelle 10-3**:* St1 Pinbelegung

RS-232 Signal	**RS-232 Ein- / Ausgang**	**Funktion**	**Pin am 15-pol. D-Sub**	**Pin (St1) Kanal B**
DCD	Eingang	DCD	1	1
DSR	Eingang	DSR	9	2
RxD	Eingang	RxD	2	3
RTS	Ausgang	RTS	10	4
TxD	Ausgang	TxD	3	5
CTS	Eingang	CTS	11	6
DTR	Ausgang	DTR	4	7
Ri	Eingang	Ri oder CLK_{in} (an SYNC und in Mode 0 zusätzlich invertiert an TRxC)	12	8
GND	Ausgang	Ground	5	9
EXTO	Ausgang	CLK_{out} (von TRxC des SCC)	13	10
EXTI	Eingang	CLK_{in} (an RTxC und in Mode 4 zusätzlich invertiert an TRxC)	6	11
-	-	nicht angeschlossen	7,8,14,15	12..16

Programmierung

Auf dem Modul müssen folgende Funktionsgruppen programmiert werden:

a) Anwahl einer Interrupt-Leitung (vom Modul zur Basiskarte)

b) Programmierung des SCC-Bausteins (Kanal A und B)

c) Konfiguration der Schnittstellen A und B (außerhalb SCC)

Zusätzlich zu den in dieser Beschreibung angegebenen Konfigurationsmöglichkeiten bestehen noch weitere Möglichkeiten, die ggf. den Austausch eines IC (FPGA) erforderlich machen. Falls Sie also spezielle Konfigurationen benötigen, sollten Sie dies anfragen. Alle in dieser Beschreibung gemachten Angaben beziehen sich auf eine Bestückung mit den ausgelieferten Standard-FPGA. Beide Kanäle können unabhängig voneinander konfiguriert werden.

Anwahl einer Interrupt-Leitung zur Basiskarte

Das Modul ist interruptfähig. Die Interrupt-Leitung des Moduls kann per Software mit einem der Interrupt-Eingänge der Basiskarte verbunden werden. Während der Anwahl einer Interrupt-Leitung darf das Modul keinen Interrupt anfordern, d. h., im SCC müssen (vorübergehend) alle Interrupts maskiert werden, z. B. durch Reset des SCC. Die Anwahl einer Interrupt-Leitung geschieht durch Setzen der drei Interrupt-Anwahlleitungen C1, C2 und C3:

Interrupt-Leitung des Moduls an MODULAR-4/486	**C1**	**C2**	**C3**
keine	0	0	0
IRQ-A	0	1	0
IRQ-F	0	1	1
IRQ-B	0	0	1
IRQ-E	1	1	1
IRQ-C	1	1	0
IRQ-D	1	0	1

Programmierung des SCC-Bausteins

Da dieser Baustein etwas komplex ist, wird hierzu auf die Literatur von Zilog (zum Baustein Z8530, Z85C30 und Z85230) bzw. Intel (82530) verwiesen. Dabei sind die Bausteine Z8530, Z85C30 und 82530 untereinander kompatibel, der Baustein Z85230 stellt eine neuere und erweiterte Version dar (u. a. größere FIFOs).

Ein ausführliches Programmierhandbuch (in engl. Sprache) zu diesem Baustein ist bei SORCUS erhältlich (Best.-Nr.: MA-1529).

Bedingt durch die Konstruktion des Moduls sind bei der Programmierung des SCC einige Dinge zu berücksichtigen:

1. Die Pins "SYNC/A" und "SYNC/B" des SCC müssen im SCC als Eingänge konfiguriert werden. "SYNC/B" wird für die Modem-Steuerleitung "Ri/B" verwendet. "SYNC/A" und "SYNC/B" werden auf dem Modul nicht für einen Quarzoszillator eingesetzt (siehe SCC-Beschreibung).

2. Die Pins "RTxC/A" und "RTxC/B" des SCC müssen im SCC als Eingänge konfiguriert werden. Sie werden auf dem Modul nicht für einen Quarzoszillator eingesetzt (siehe SCC-Beschreibung). Das an diesen Pins anliegende Signal kann innerhalb des SCC für den jeweiligen Kanal für folgende Zwecke verwendet werden:

Pin	**In-/Output**	**Verwendbar als**
RTxC/A	Input	RCV-Clock/A TMT-Clock/A Eingang der DPLL/A Eingang von Baudratengenerator/A
RTxC/B	Input	RCV-Clock/B TMT-Clock/B Eingang der DPLL/B Eingang von Baudratengenerator/B

3. Der Pin "TRxC/A" des SCC muß immer als Eingang konfiguriert werden. Er hat hier keine Funktion, mit einer Ausnahme: Für besondere Anwendungen liegt an Pin "TRxC/A" das Signal des Empfängers von Kanal B. Er wäre also denkbar, hierüber einen Takt für Kanal A einzuspeisen. Der Pin "TRxC/B" des SCC muß abhängig von der jeweiligen Konfiguration als Ein- oder Ausgang konfiguriert werden. Als Ausgang kann er eines der folgenden Signale nach außen liefern:

Pin	**In-/Output**	**kann liefern**
TRxC/B	Output	TMT-Clock von Kanal B Ausgang Baudratengenerator von Kanal B Ausgang von RCV DPLL von Kanal B Signal an Pin "RTxC/B"

Als Eingang kann TRxC/B folgendermaßen eingesetzt werden:

Pin	**In-/Output**	**Verwendbar als**
TRxC/B	Input	RCV-Clock für Kanal B TMT-Clock für Kanal B

4. Am Pin "PCLK" des SCC liegt der Takt des Quarzoszillators des Moduls (standardmäßig 7,3728 MHz, optional eine andere Frequenz, z. B. 4,9152 MHz). Dieses Signal kann z. B. als Eingangstakt für die Baudratengeneratoren verwendet werden. Je nach Betriebsart (Mode) kann dieser Takt auch an RTxC angelegt werden und damit direkt als Eingangstakt für Sender und/oder Empfänger dienen, um hohe Baudraten zu erreichen.

Tabelle: Reihenfolge der Initialisierung für den SCC

Register	**Data**	**Kommentar**
1. Stufe: Betriebsarten und Konstanten definieren		
WR9	1100 0000	Hardware Reset
WR0	0000 00xx	Select Shift mode (nur bei Z8030)
WR4	xxxx xxxx	Transmit/Receive Control: Asyn- oder Synchrone Betriebsart anwählen
WR1	0xx0 0x00	W/REQ anwählen (optional)
WR2	xxxx xxxx	Interruptvektor programmieren
WR3	xxxx xxx0	Receiver control, Bit D0 (Rx enable) muß hier auf 0 gesetzt werden.
WR5	xxxx 0xxx	Transmit control, Bit D3 (Tx enable) muß hier auf 0 gesetzt werden.
WR6	xxxx xxxx	SYNC-Zeichen programmieren
WR7	xxxx xxxx	SYNC-Zeichen programmieren
WR9	000x 0xxx	Interrupt control anwählen. Bit D3 (MIE) muß auf 0 gesetzt werden.
WR10	xxxx xxxx	Kontrollwort (optional)
WR11	xxxx xxxx	Clock control
WR12	xxxx xxxx	Zeitkonstante, Lowbyte (optional)
WR13	xxxx xxxx	Zeitkonstante, Highbyte (optional)
WR14	xxxx xxx0	Kontrollwort, Bit 0 (BR enable) muß hier auf 0 gesetzt werden.
WR14	xxxS SSSS	Register kann mehrfach beschrieben werden, wenn mehr als ein Kommando geschickt werden soll
2. Stufe: Enables		
WR3	SSSS SSS1	Bit D0 (Rx Enable) auf 1 setzen.
WR5	SSSS 1SSS	Bit D3 (Tx Enable) auf 1 setzen
WR0	1000 0000	Reset TxCRC
WR14	000S SSS1	Baudraten Generator Enable, Bit D0 auf 1 setzen, Enable DPLL
WR1	xSS0 0S00	D7 auf 1 setzen, wenn DMA enabled werden soll.
Stufe 3: Interrupt Enable		
WR15	xxxx xxxx	Enable external interrupts
WR0	0001 0000	Reset EXT/STATUS
WR0	0001 0000	Reset EXT/STATUS
WR1	SSSx xSxx	Enable receive, transmit and external interrupt master.
WR9	000S xSSS	Enable Master Interrupt bit D3.

1 = Auf 1 setzen, 0 = auf 0 setzen, x = nach Wunsch setzen, S = so setzen wie bereits gesetzt

Arbeitsblatt:

Stufen	Register	Hex	Bit	Kommentar
1. Betriebsarten				
	WR9	_C_ _0_	1100 0000	Software Reset
	WR0	_0_ ____	0000 00__	_____
	WR4	___ ___	____ ____	_____
	WR1	___ ___	0__0 0_00	_____
	WR2	___ ___	____ ____	_____
	WR3	___ ___	____ ___0	_____
	WR5	___ ___	____ 0___	_____
	WR6	___ ___	____ ____	_____
	WR7	___ ___	____ ____	_____
	WR9	___ ___	000_ 0___	_____
	WR10	___ ___	____ ____	_____
	WR11	___ ___	____ ____	_____
	WR12	___ ___	____ ____	_____
	WR13	___ ___	____ ____	_____
	WR14	___ ___	____ ___0	_____
	WR14	___ ___	____ ___0	_____
2. Enables				
	WR3	___ ___	____ ___1	_____
	WR5	___ ___	____ 1___	_____
	WR0	_8_ _0_	1000 0000	Reset TxCRC
	WR14	___ ___	000_ ___1	_____
	WR1	___ ___	____ ____	_____
3. Interrupt				
	WR15	___ ___	____ ____	_____
	WR0	_1_ _0_	0001 0000	Reset Ext/Status
	WR0	_1_ _0_	0001 0000	Reset Ext/Status
	WR1	___ ___	____ ____	_____
	WR9	___ ___	000_ ____	_____

Initialisierungswerte nach Reset:

Register	**Hardware-Reset**	**Channel-Reset**
WR0	0000 0010	0000 0000
WR1	00_0 0_00	00_0 0_00
WR2	____ ____	____ ____
WR3	____ ___0	____ ___0
WR4	____ _1__	____ _1__
WR5	0__0 000_	0__0 0000
WR6	____ ____	____ ____
WR7	____ ____	____ ____
WR9	1100 00__	__0_ ____
WR10	0000 0000	0__0 0000
WR11	0000 1000	____ ____
WR12	____ ____	____ ____
WR13	____ ____	____ ____
WR14	__10 0000	__10 00__
WR15	1111 1000	1111 1000
RR0	01__ _100	01___100
RR1	0000 0111	0000 0111
RR3	0000 0000	0000 0000
RR10	0000 0000	0000 0000

Hochsprachenbibliothek

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M050_LIB**, Sie finden sie im Verzeichnis (*pathname*) **SPB_MOD \ BIB \ M-DAS-A**. Vor allen anderen Routinen muß die Prozedur **m050_bib_startup** einmal aufgerufen werden.

m050_bib_startup — Initialisiere Modulbibliothek

Pascal PROCEDURE m050_bib_startup (micro_slot: byte);

C void EXPORT m050_bib_startup (byte micro_slot);

Funktion Diese Prozedur initialisiert die Modulbibliothek M050_LIB. Es werden u.a. die Initialisierungsdaten aus den EEPROMs aller M-DAS-A Module übernommen, die sich auf der Basiskarte befinden. Die Register des Gate-Arrays werden gemäß den EEPROM-Inhalten gesetzt.

m050_scc_reset — Führe SCC-Hardware-Reset durch

Pascal PROCEDURE m050_scc_reset (micro_slot);

C void EXPORT m050_scc_reset (byte micro_slot);

Funktion Diese Prozedur führt einen Hardware-Reset des SCC durch.

m050_init_scc Initialisiere SCC-Kanal

Pascal FUNCTION m050_init_scc (micro_slot, channel, mode, intmode, int_x_mode: byte; timeconst: word; use_brg, clkmode, databits, stopbits, parity_type: byte): byte;

C byte EXPORT m050_init_scc (byte micro_slot, byte channel, byte mode, byte intmode, byte int_x_mode, ushort timeconst, byte use_brg, byte clkmode, byte databits, byte stopbits, byte parity_type);

Funktion Diese Funktion initialisiert einen Kanal des Moduls.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

mode: **Kanal A**: = 0: Sende- und Empfangspegel „invertiert“

= 1: Sende- und Empfangspegel „nicht invertiert“

Kanal B: Einstellung der Taktquellen für Receiver und Transmitter:

mode	CTS	RTS	Funktion von RTxC	Funktion von TRxC	TRxC Ein-/Ausgang
0	CTS	RTS	CLKin von EXTI	CLKin von Ri	Eingang
1	CTS	RTS	Quarz	keine	Eingang
2,3	res.	res.	res.	res.	res.
4	CTS	RTS	CLKin von EXTI	CLKin von EXTI	Eingang
5	CTS	RTS	Quarz	CLKout an EXTO	Ausgang
6	res.	res.	res.	res.	res.
7	CTS	RTS	Quarz	keine	Ausgang

int_mode: Interrupt-Betriebsart festlegen

7	6	5	4	3	2	1	0	
								Interrupt Enable
0	0	0						Reserviert
			1	0				Interrupt On All Rx Character or Special Condition
			0	0				Rx Interrupt Disable
					1			Parity is Special Condition
						1		Tx Interrupt Enable
							1	Ext. Interrupt Enable

int_x_mode: Erweiterte Interrupt-Betriebsart festlegen

7	6	5	4	3	2	1	0	
								Ext./Status Interrupt Select
1								Break/Abort Interrupt Enable
	0							reserviert
		1						CTS Interrupt Enable (nur Kanal B)
			0					reserviert
				1				DCD Interrupt Enable (nur Kanal B)
					0	0	0	reserviert

timeconst: Zeitkonstante für Baudratengenerator (0 bis ffffh)

clkmode: Clock-Mode (1, 16, 32, 64) für Receiver und Transmitter

use_brg: Baudratengenerator verwenden (1) oder nicht (0)

databits: Anzahl Datenbits (5 bis 8)

stopbits: Anzahl Stopbits (0 = 0, 1 = 1, 2 = 1,5 3 = 2 Stopbits)

parity_type: Parität (0 = keine Parität, 1 = odd, 2 = even)

m050_define_intrp_connection Interruptanwahl

Pascal PROCEDURE m050_define_intrp_connection (micro_slot, intrp: byte);

C void EXPORT m050_define_intrp_connection (byte micro_slot, byte intrp);

Funktion Diese Prozedur wählt eine Interruptleitung zur Basiskarte an.

Parameter *intrp*: Nummer der Interrupt-Leitung (0 = kein Interrupt, 1 = IRQ-A, 2 = IRQ-F, 3 = IRQ-B, 4 = IRQ-E, 5 = IRQ-C und 6 = IRQ-D)

m050_transmit_scc_character Sende Zeichen

Pascal FUNCTION m050_transmit_scc_character (micro_slot, channel, data: byte): byte;

C byte EXPORT m050_transmit_scc_character (byte micro_slot, byte channel, byte data);

Funktion Diese Funktion sendet ein Zeichen. Vor dem Senden wird der Zustand des Sendepuffers abgefragt. Ist der Sendepuffer voll (TBE = 0), so wird 1 zurückgeliefert. Wurde das Zeichen erfolgreich in den Sendepuffer geschrieben, liefert die Funktion 0 zurück.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

data: Zu sendendes Datenbyte

m050_receive_scc_character Empfange Zeichen

Pascal FUNCTION m050_receive_scc_character (micro_slot, channel: byte; var data: byte): byte;

C byte EXPORT m050_receive_scc_character (byte micro_slot, byte channel, byte *data);

Funktion Diese Funktion liest den Empfangspuffer eines Kanals. Vor dem Empfangen wird der Zustand des Empfangspuffers abgefragt. Ist dieser leer (RBF = 0), so wird 1 zurückgeliefert. Wurde ein Zeichen empfangen, liefert die Funktion 0 zurück.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

data: Empfangenes Datenbyte

m050_scc_rbf Frage Empfangspuffer-Status (RBF) ab

Pascal FUNCTION m050_scc_rbf (micro_slot, channel: byte): byte;

C byte EXPORT m050_scc_rbf (byte micro_slot, byte channel);

Funktion Diese Funktion prüft den Status des Empfangspuffers eines Kanals. Der Zustand des RBF-Bits wird zurückgegeben (RBF = 1: receive buffer full, RBF = 0 receive buffer empty).

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

m050_scc_tbe Frage Sendepuffer-Status (TBE) ab

Pascal FUNCTION m050_scc_tbe (micro_slot, channel: byte): byte;

C byte EXPORT m050_scc_tbe (byte micro_slot, byte channel);

Funktion Diese Funktion prüft den Status des Sendepuffers eines Kanals. Der Zustand des TBE-Bits wird zurückgegeben (TBE = 1: transmit buffer empty, TBE = 0 transmit buffer full).

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

m050_scc_cts Frage Clear To Send (CTS) ab

Pascal FUNCTION m050_scc_cts (micro_slot, channel: byte): byte;

C byte EXPORT m050_scc_cts (byte micro_slot, byte channel);

Funktion Diese Funktion liefert den aktuellen Zustand der CTS-Leitung eines Kanals.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

m050_set_rts Setze Request To Send (RTS)

Pascal PROCEDURE m050_set_rts (micro_slot, channel, rts: byte);

C void EXPORT m050_set_rts (byte micro_slot, byte channel, byte rts);

Funktion Diese Prozedur setzt die RTS-Leitung eines Kanals auf den angegebenen Wert.

Parameter *channel*: Nummer des Kanals (0 = A, 1 = B)

rts: Zu setzender Wert (0 oder 1)

m050_scc_intvec Lies Interrupt-Vektor

Pascal PROCEDURE m050_scc_intvec (micro_slot: byte; var int_vector, rr0, rr1, rr3: byte);

C void EXPORT m050_scc_intvec (byte micro_slot, byte *int_vector, byte *rr0, byte *rr1, byte *rr3);

Funktion Diese Funktion kann dazu verwendet werden, nach einer SCC-Interrupt-Anforderung, den Interrupt-Vektor zu lesen.

Parameter *vector*: Interrupt-Vektor

Bit 3 bis 1 (V3, V2 und V1) enthalten weitere Informationen über den aufgetretenen Interrupt. Die restlichen Bits sind reserviert.

rr0: Aktueller Zustand von Read-Register 0

7	6	5	4	3	2	1	0	
								Read Register 0
								Break/Abort
								Tx Underun/EOM
								CTS
								Sync/Hunt
								DCD
								Tx Buffer Empty
								Zero Count
								Rx Character Available

rr1: Aktueller Zustand von Read-Register 1

7	6	5	4	3	2	1	0	
								Read Register 1
								End of Frame
								CRC/Framing Error
								Rx Overrun Error
								Parity Error
								Residue Code 0
								Residue Code 1
								Residue Code 2
								All sent

rr3: Aktueller Zustand von Read-Register 3

7	6	5	4	3	2	1	0	
								Read Register 3 (Interrupt Pending (IP) Register)
								0
								0
								Channel A Rx IP
								Channel A Tx IP
								Channel A Ext/Status IP
								Channel B Rx IP
								Channel B Tx IP
								Channel B Ext/Status IP

m050_baudrate_calc Ermittle Einstellungen für Baudrate

Pascal
PROCEDURE m050_baudrate_calc (baudrate, quartz: longint;
var bauderror: longint;var basereg: byte; var clk_mode: byte;
var timeconst: word, var use_brg: byte);

C
void EXPORT m050_baudrate_calc (long baudrate, long quartz,
long *bauderror, byte *basereg, byte *clk_mode, ushort *timeconst,
byte *use_brg);

Funktion
Diese Prozedur ermittelt die notwendigen Einstellungen für eine gewünschte Baudrate bei einer bestimmten Frequenz des Quarzoszillators. Als Ergebnisse liefert die Prozedur die SCC-Parameter für den Baudratengenerator.

Parameter

baudrate: Gewünschte Baudrate in bd

quartz: Gibt die Taktfrequenz des Quarzoszillators in Hz an (Standard: 7,3728 MHz ≙ 7370000)

bauderror: Absoluter Fehler des Basistaktes bzw. der Baudrate, der sich bei dieser Einstellung ergibt

basereg: Reserviert

timeconst: Einstellung für die Zeitkonstante des Baudratengenerators

clk_mode: Einstellung für Clock-Mode des Receivers/Transmitters

use_brg: Verwendung des Baudratengenerators (0=nein, 1=ja)

m050_set_channel Konfiguriere Kanal

Pascal FUNCTION m050_set_channel (micro_slot, channel: byte; baudrate: word; flag, scc_mode, scc_intmode, scc_int_x_mode, databits, stopbits, parity_type: byte; var bauderror: longint; var used_clkmode: byte): byte);

C byte EXPORT m050_set_channel (byte micro_slot, byte channel, long baudrate, byte flag, byte mode, byte intmode, byte int_x_mode, byte databits, byte stopbits, byte parity_type, long *bauderror, byte *used_clkmode);

Funktion Diese Funktion stellt eine Kombination aus **m050_baudrate_calc** und **m050_init_scc** dar (siehe auch Parameter dieser Funktionen).

Parameter *flag*: Reserviert

Programmierung mit I/O-Zugriffen

Lokale I/O-Adressen

Adresse	Zugr.	Funktion
MBA+00h	RW8	SCC: Kanal B, Control
MBA+01h	RW8	Kanal B, Data
MBA+02h	RW8	Kanal A, Control
MBA+03h	RW8	Kanal A, Data
MBA+07h	W8	Interrupt anwählen: Port C1 [1]
MBA+08h	W8	Port C2 [1]
MBA+09h	W8	Port C3 [1]

Interrupt:	Keiner	IRQ-A	IRQ-B	IRQ-C	IRQ-D	IRQ-E	IRQ-F
C3, C2, C1:	000	010	100	011	101	111	110

MBA+0ah	W8	Konfiguration Schnittstelle A: Port A1 [1]
MBA+0bh	W8	Port A2 [1]
MBA+0ch	W8	Port A3 [1]
MBA+0dh	W8	Konfiguration Schnittstelle B: Port B1 [1]
MBA+0eh	W8	Port B2 [1]
MBA+0fh	W8	Port B3 [1]
MBA+0ah	R8	Status lesen: DSR/B[2]
MBA+0ch	R8	Port C1 [2, 3]
MBA+1nh (MBA+18h – MBA+1fh)	R8	Lesen der FPGA-Version von IC1, 8 Bit, je Zugriff liefert Bit 0 ein Bit der Versions-Nr., wobei n = 08h..0fh und (n-8) = Wertigkeit des Bit ist (aktuelle Version = 1)

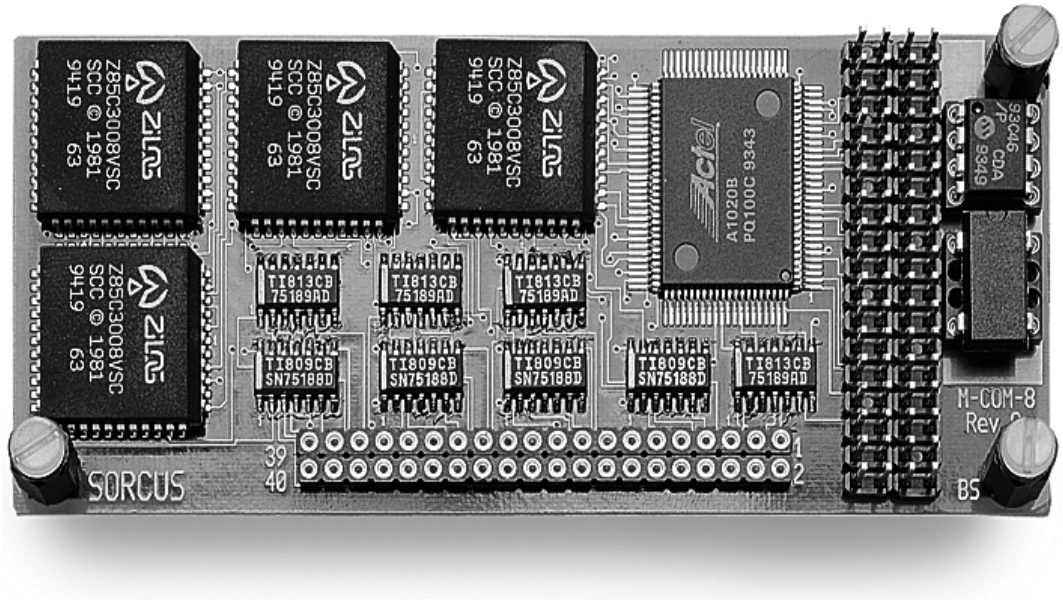
[1] Beim Schreiben wird der Port entsprechend Bit 0 gesetzt, Bit 1 bis Bit 7 sind ohne Bedeutung.

[2] Beim Lesen steht das Ergebnis in Bit 0, die anderen Bit sind ungültig.

[3] Nur für Testzwecke.

5. M-COM-8

Acht serielle RS-232-Schnittstellen



Funktionsbeschreibung

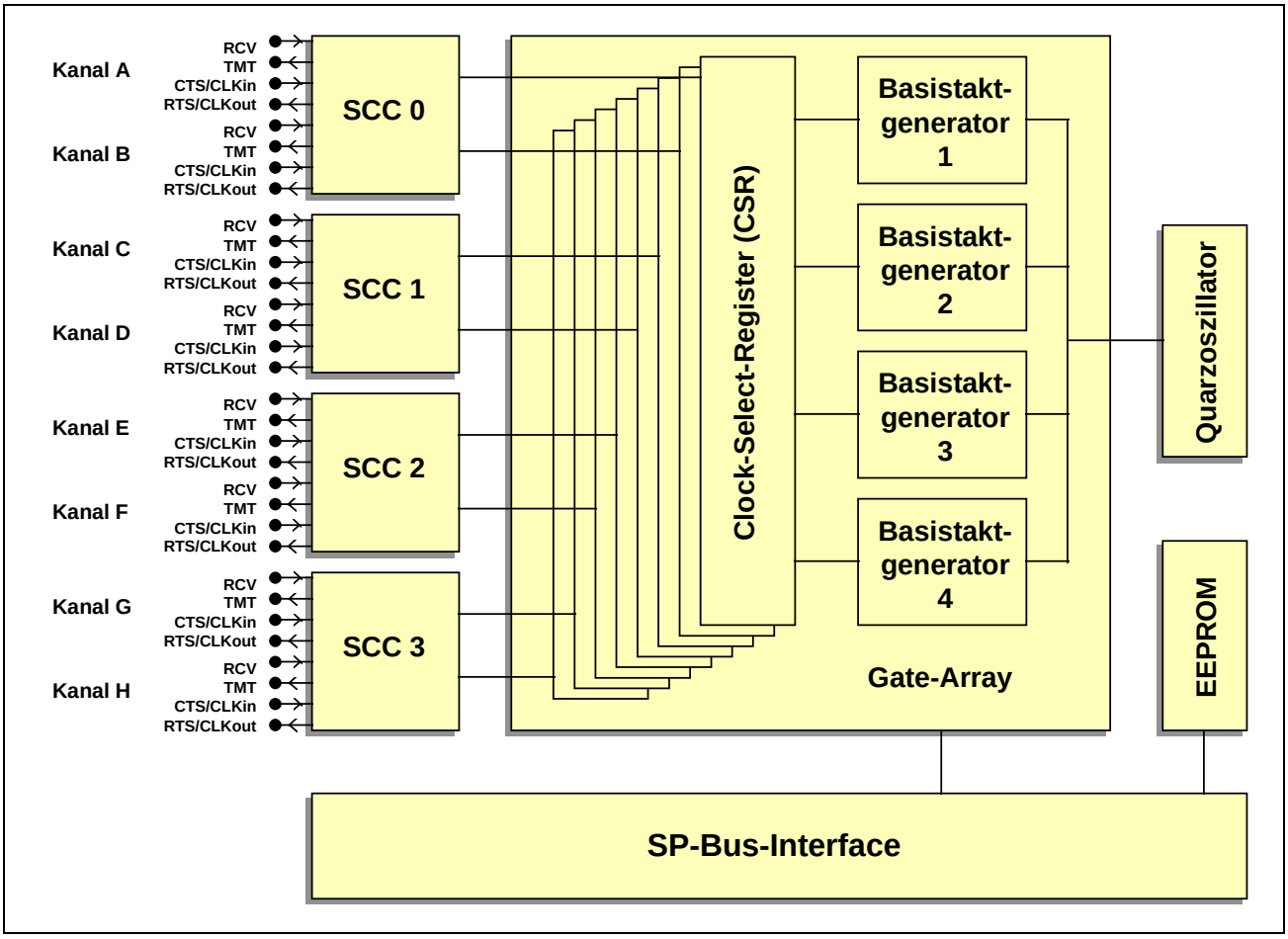
Das Modul M-COM-8 ist ein Kommunikationsmodul für das MODULAR-4 System. Es enthält acht serielle synchrone bzw. asynchrone RS-232-Schnittstellen (Kanal A bis H). Die Kanäle sind vollständig über Software konfigurierbar, das Modul hat keine Jumper.

Vier SCC-Bausteine vom Typ Z85x30 (SCC = Serial Communication Controller) stellen die Schnittstellen zur Verfügung. Ein Gate-Array dient als Schnittstelle zwischen den SCCs und dem SP-Bus (SORCUS-Prozeß-Bus) der Basiskarte.

Das Gate-Array enthält zusätzlich vier programmierbare Basistaktgeneratoren (Vorteiler), die jeweils einen sog. Basistakt erzeugen. Alle acht Kanäle können diese Basistakte nutzen, z. B. für die Baudratengeneratoren der SCCs.

Durch die Programmierbarkeit der Basistaktgeneratoren und die SCC-internen Baudratengeneratoren können neben den gängigen Baudraten eine Vielzahl von weiteren Baudraten eingestellt werden.

Blockschaltbild



Technische Daten

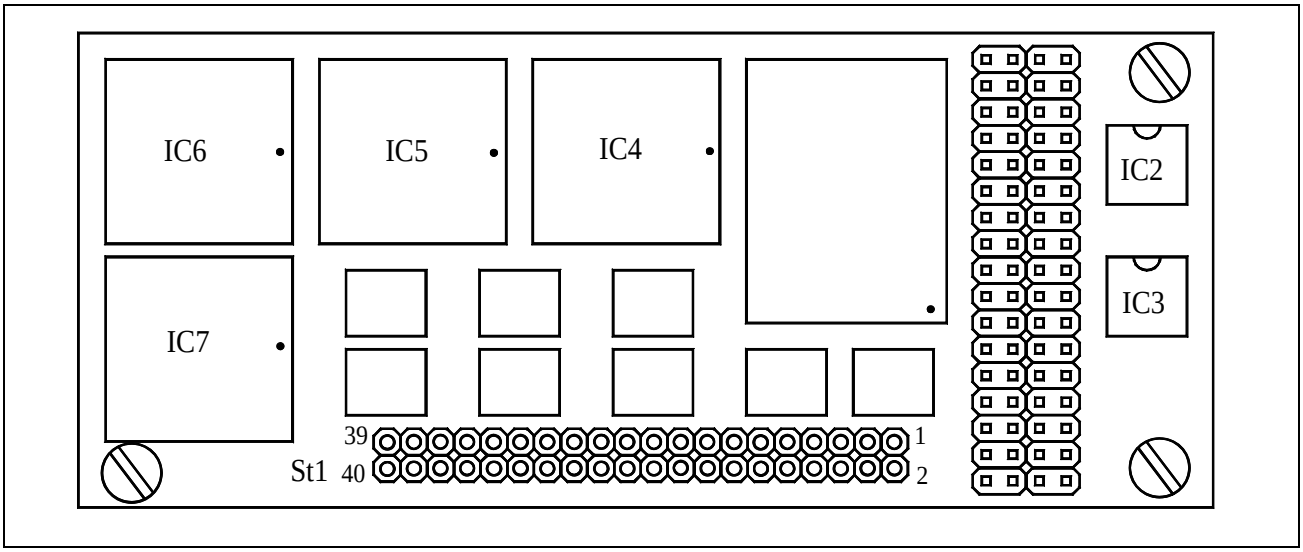
Parameter		Wert	Einheit
Anzahl serieller Schnittstellen, RS-232		8	-
Serieller Kommunikationsbaustein		Z85C30 oder Z85230	-
Interruptfähig zur Basiskarte (Interrupt-Kanal per Software anwählbar)		ja	-
Modem-Steuerleitungen je Schnittstelle		2	-
1 Eingang, verwendbar als		CTS, CLK_{in}	-
1 Ausgang, verwendbar als		RTS, CLK_{out}	-
Baudratengeneratoren (per Software einstellbar)		8 (1 je Kanal)	-
Basistaktgeneratoren (per Software einstellbar)		4	-
Versorgungsspannungen (von der Basiskarte)		+5, ±12	V
Stromaufnahme	+5 V	120	mA
(typ., extern nichts angeschlossen)	+12 V	50	mA
	-12 V	50	mA
Betriebstemperatur		0 bis 60	°C
Abmessungen (L x B x H)		106 x 45 x 15	mm

Lieferumfang

- Modul M-COM-8
- 40-poliger Pfostenstecker für Flachbandkabel
- Datenträger mit Programmbibliotheken und Software zur gepufferten Kommunikation CQ8

Konfiguration und Einbau

Lageplan



EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0001	0011 0000	2130h	Modultyp M-COM-8
1	0000 0000	0000 0000	0000h	Initialisierung
2	0010 0001	0010 0001	2121h	Bestückung IC4 und IC5
3	0010 0001	0010 0001	2121h	Bestückung IC6 und IC7
4	0011 0010	0000 0000	2400h	Quarzoszillator-Frequenz
5	0000 0000	0000 0000	0000h	Interrupt-Kanal zur Basiskarte
6	0000 0000	0000 0000	0000h	PCLK-Select
7	0000 0000	0000 0000	0000h	Clock-Select Kanal A
8	0000 0000	0000 0000	0000h	Clock-Select Kanal B
9	0000 0000	0000 0000	0000h	Clock-Select Kanal C
10	0000 0000	0000 0000	0000h	Clock-Select Kanal D
11	0000 0000	0000 0000	0000h	Clock-Select Kanal E
12	0000 0000	0000 0000	0000h	Clock-Select Kanal F
13	0000 0000	0000 0000	0000h	Clock-Select Kanal G
14	0000 0000	0000 0000	0000h	Clock-Select Kanal H
15	0000 0000	0000 0000	0000h	Basistakt 1
16	0000 0000	0000 0000	0000h	Basistakt 2
17	0000 0000	0000 0000	0000h	Basistakt 3
18	0000 0000	0000 0000	0000h	Basistakt 4
19	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0	WORT-0: Kennung
								0	0	1	1	0	0	0	0	Modultyp: 48 = M-COM-8
				0	0	0	1									Revision: 1 = A, 2 = B.
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort kann eingestellt werden, ob das Modul nach dem Einschalten und bei einem Hardware-Reset entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															0	Init nach Hardreset: 0 = nein, 1 = ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-2: Bestückung IC 4 und IC 5

EEPROM-Wort 2 enthält werkseitig voreingestellte Angaben über die SCC-Bestückung für IC 4 (Kanäle A und B) und IC 5 (Kanäle C und D).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	WORT-2: Bestückung (werks. Einst.)
																geändert am: von:
												0	0	0	1	SCC-Typ IC4: 0 = nicht bestückt 1 = SCC (8530) 2 = ESCC (85130) 3 = ESCC (85230)
								0	0	1	0					Max. Takt für IC4: 0 = nicht bestückt 1 = 6 MHz 2 = 8 MHz 3 = 10 MHz 4 = 16 MHz 5 = 20 MHz
				0	0	0	1									SCC-Typ IC5: 0 = nicht bestückt 1 = SCC (8530) 2 = ESCC (85130) 3 = ESCC (85230)
0	0	1	0													Max. Takt für IC5: 0 = nicht bestückt 1 = 6 MHz 2 = 8 MHz 3 = 10 MHz 4 = 16 MHz 5 = 20 MHz

WORT-3: Bestückung IC 6 und IC 7

EEPROM-Wort 3 enthält werkseitig voreingestellte Angaben über die SCC-Bestückung für IC 6 (Kanäle E und F) und IC 7 (Kanäle G und H).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	WORT-3 Bestückung (werks. Einst.)
																geändert am: von:
												0	0	0	1	SCC-Typ IC6: 0 = nicht bestückt 1 = SCC (8530) 2 = ESCC (85130) 3 = ESCC (85230)
								0	0	1	0					Max. Takt für IC6: 0 = nicht bestückt 1 = 6 MHz 2 = 8 MHz 3 = 10 MHz 4 = 16 MHz 5 = 20 MHz
				0	0	0	1									SCC-Typ IC7: 0 = nicht bestückt 1 = SCC (8530) 2 = ESCC (85130) 3 = ESCC (85230)
0	0	1	0													Max. Takt für IC7: 0 = nicht bestückt 1 = 6 MHz 2 = 8 MHz 3 = 10 MHz 4 = 16 MHz 5 = 20 MHz

WORT-4: Bestückung: Quarzoszillator

EEPROM-Wort 4 gibt die werkseitige Bestückung des Quarzoszillators an. Die Angabe erfolgt mit 4 Ziffern in MHz mit 2 Vorkomma- und 2 Nachkommastellen. 4,9152 MHz würde also aufgerundet auf 4,92 und mit 4 Dezimalziffern 0492 (= 0492h) eingegeben. 0000h bedeutet, daß der Quarzoszillator nicht bestückt ist, die folgende Angabe 2400h bedeutet 24,00 MHz.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	WORT-4: Quarzoszillator-Frequenz (werks. Einst.)
																geändert am: von:
												0	0	0	0	2. Ziffer nach Komma
								0	0	0	0					1. Ziffer nach Komma
				0	1	0	0									2. Ziffer
0	0	1	0													1. Ziffer

Die EEPROM-Inhalte der Wörter 5 bis 13 dienen zur Abspeicherung einer anwenderspezifischen Modulkonfiguration. Die EEPROM-Inhalte werden nicht direkt den entsprechenden Registern des Moduls zugeordnet, sondern können vom Anwenderprogramm übernommen und zur Programmierung der Register (siehe ab Seite 5-15) verwendet werden.

WORT-5: Interrupt-Kanal zur Basiskarte

Hier kann der Anwender den Interrupt-Kanal der Basiskarte abspeichern, mit dem das Modul verbunden wird. Entsprechend kann das Interrupt-Select-Register (ISR) des Moduls gesetzt werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-5: Interrupt-Kanal (werks. Einst.)
																geändert am: von:
													0	0	0	Interrupt-Kanal 0 = kein Interrupt 1 = IRQ-A 2 = IRQ-B 3 = IRQ-E 4 = IRQ-C 5 = IRQ-D
0	0	0	0	0	0	0	0	0	0	0	0	0				Reserviert

WORT-6: PCLK-Select

WORT-6 speichert die Einstellung für die SCC-Taktfrequenz PCLK für alle 4 SCCs entsprechend dem PCLK-Select-Register (PSR) des Moduls.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-6: PCLK-Takt (werks. Einst.)
																geändert am: von:
														0	0	PCLK-Einstellung 0 = PCLK = Quarzosz. /5 1 = PCLK = Quarzosz. /4 2 = PCLK = Quarzosz. /3 3 = PCLK = Quarzosz. /2
0	0	0	0	0	0	0	0	0	0	0	0	0	0			Reserviert

WORT-7 bis WORT-14: Clock-Select-Konfiguration für Kanal A bis H

Hier kann der Anwender Konfigurationsdaten für die Clock-Select-Register (CSR) aller 8 Kanäle abspeichern. Die CSR/A bis /H Register des Moduls legen fest, welcher bzw. ob ein Basistakt oder ein anderer Takt an den RTxC-Pin des jeweiligen Kanals angelegt wird. Ebenfalls kann eingestellt werden, was am RTS-Pin des Kanals ausgegeben werden soll.

EEPROM-Wort	Clock-Select-Konfiguration für Kanal
7	A
8	B
9	C
10	D
11	E
12	F
13	G
14	H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-7 bis WORT-14: Clock-Select-Register Kanal A bis H (werks. Einst.)
																W-7 geändert am: von:
																W-8 geändert am: von:
																W-9 geändert am: von:
																W-10 geändert am: von:
																W-11 geändert am: von:
																W-12 geändert am: von:
																W-13 geändert am: von:
																W-14 geändert am: von:
													0	0	0	Clock-Select-Register (CSR/A bis /H) 0 = kein Takt (Kanal nicht verwendet) 1 = Basistakt-1 2 = Basistakt-2 3 = Basistakt-3 4 = Basistakt-4 5 = Quarzoszillator /4 6 = Reserviert 7 = CLK_{in} von CTS
												0				RTS-Ausgang 0 = RTS des SCC 1 = TRxC des SCC (als CLK_{out})
0	0	0	0	0	0	0	0	0	0	0	0					Reserviert

WORT-15 bis WORT-18: Einstellung der Basistakte 1 bis 4

Hier kann der Anwender die Einstellung für die Basistaktgeneratoren (BTR/1 bis /4) abspeichern (siehe Seite 5-17).

EEPROM-Wort	Basistaktgenerator
15	1
16	2
17	3
18	4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-15 bis WORT-18: Basistakt 1 bis 4 (werks. Einst.)
																W-15 geändert am: von:
																W-16 geändert am: von:
																W-17 geändert am: von:
																W-18 geändert am: von:
								0	0	0	0	0	0	0	0	Basistaktregister 1-4 (BTR/1 bis /4)
0	0	0	0	0	0	0	0									Reserviert

Programmierung

Auf dem Modul müssen folgende Funktionsgruppen programmiert werden:

Im Gate-Array des Moduls:

- Wahl des PCLK-Taktes für alle 4 SCCs gemeinsam
  (PCLK-Select-Register PSR, 2 Bit)
- Einstellung der vier Basistaktgeneratoren
  (Basistaktgenerator-Register BTR/1-4, je 8 Bit)
- Zuordnung der Basistakte
  (Wahl des Eingangssignals für den RTxC-Pin eines Kanals)
  (Clock-Select-Register CSR/A bis /H, je 4 Bit, Channel-Select-Register CHR, 3 Bit)
- Anwahl einer Interrupt-Leitung (vom Modul zur Basiskarte)
  (Interrupt-Select-Register ISR, 3 Bit)

In den 4 SCCs:

- Interrupt-Vektoren
- Baudratengeneratoren
- Übertragungsparameter (Parity, Anzahl Datenbits, etc.)
- Diverses

Einstellung des PCLK-Taktes

Je nach Quarzfrequenz und SCC-Typ des Moduls ist es notwendig, den Eingangstakt der SCCs (PCLK) einzustellen. Bit 0 und 1 im PCLK-Select-Register (PSR) legen den Teilerfaktor für die Quarzoszillator-Frequenz fest:

Bit 1	**Bit 0**	**PCLK**
0	0	PCLK = Quarzoszillator · 1/5
0	1	PCLK = Quarzoszillator · 1/4
1	0	PCLK = Quarzoszillator · 1/3
1	1	PCLK = Quarzoszillator · 1/2

Bit-2 bis Bit-7 sind reserviert und sollten auf 0 gesetzt werden.

Einstellung der Basistakte für die Baudratengeneratoren

Das Modul enthält vier programmierbare Basistaktgeneratoren und 8 Baudratengeneratoren. Jeder der acht Baudratengeneratoren in den SCCs kann mit einem der vier Basistakte verbunden werden (siehe Seite 5-19). Zur Einstellung der Basistaktgeneratoren dienen die 4 Basistaktgenerator-Register (BTR/1 bis /4).

Viele Standard-Baudraten können durch folgende Einstellungen in den Basistaktregistern erzeugt werden:

Basistakt = 7,3846 MHz, Einstellung im Basistaktregister BTR: 1000 1100 (8ch)

Beispiele für mögliche Baudraten (bei einer Abweichung von 0,16%):

50 Baud	60 Baud	110 Baud	150 Baud	220 Baud
300 Baud	600 Baud	1,2 kBaud	2,4 kBaud	4,8 kBaud
9,6 kBaud	19,2 kBaud	38,4 kBaud	57,6 kBaud	76,8 kBaud
115,2 kBaud	230,4 kBaud	460,8 kBaud		

Basistakt = 7,68 MHz, Einstellung im Basistaktregister BTR: 1001 1000 (98h)

Beispiele für mögliche Baudraten:

50 Baud	60 Baud	110 Baud	150 Baud	220 Baud
300 Baud	600 Baud	1,2 kBaud	2,4 kBaud	4,8 kBaud
9,6 kBaud	48 kBaud	480 kBaud		

Basistakt = 8 MHz, Einstellung im Basistaktregister BTR: 1000 0000 (80h)

Beispiele für mögliche Baudraten:

50 Baud	60 Baud	110 Baud	150 Baud	220 Baud
300 Baud	600 Baud	1,2 kBaud	2,4 kBaud	4,8 kBaud
9,6 kBaud	19,2 kBaud	500 kBaud		

Darüber hinaus können weitere Baudraten erzeugt werden, wenn man einen der Basistakte ändert oder die Baudratengeneratoren entsprechend programmiert.

Berechnung der Basistaktfrequenz:

$$Basistakt = Q \cdot \frac{X \cdot (Y+1) - Y}{X \cdot (Y+1) + 1} \cdot \frac{1}{5 - Z}$$

mit

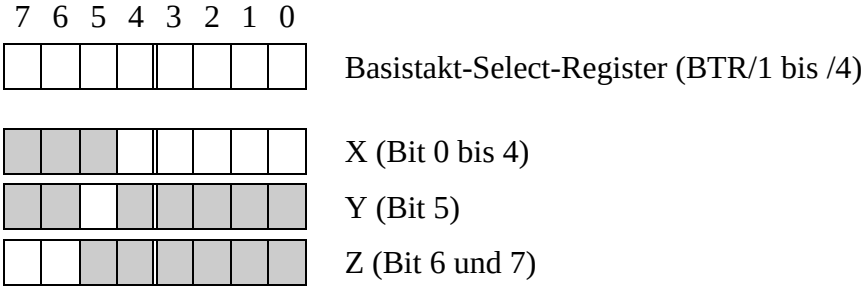
$Q = Frequenz\ des\ Quarzoszillators$

$X = 1\ bis\ 31$

$Y = 0\ oder\ 1$

$Z = 0\ bis\ 3$

Einstellung in den Basistakt-Select-Register BTR/1 bis /4:



Beispiel: **BTR/1 = 98h, Quarzoszillator = 24 MHz**

Frequenz von Basistakt 1 = 24 MHz · 24/25 · 1/3 = 7,68 MHz

Basistakt auswählen und RTS/CTS-Mode einstellen

Jeder der acht Kanäle kann mit einem der vier programmierbaren Basistakte versorgt werden. Dabei wird der Ausgang eines Basistaktgenerators als Takteingang an den jeweiligen RTxC-Pin des SCC geschaltet.

Mit dem Clock-Select-Register (CSR/A bis /H, 4 Bit) des jeweiligen Kanals wird einer der 4 Basistakte ausgewählt. Es ist auch möglich, das CTS-Signal der Schnittstelle dem jeweiligen Baudratengenerator zuzuführen und somit den Takt von außen zuzuführen (als CLK_{in}).

Zusätzlich kann je Kanal eingestellt werden, welches Signal am RTS-Ausgang anliegen soll. Der RTS-Pin eines Kanals kann entweder mit dem RTS-Ausgang des SCC oder mit dem TRxC-Ausgang des gleichen Kanals verbunden werden (als CLK_{out}).

Clock-Select-Register CSR/A bis CSR/H (für die Kanäle A bis H) [1]:

Bit 2	Bit 1	Bit 0	Eingangstakt des Baudratengenerators
0	0	0	Kanal wird nicht verwendet
0	0	1	Basistakt 1
0	1	0	Basistakt 2
0	1	1	Basistakt 3
1	0	0	Basistakt 4
1	0	1	Quarzoszillator/4
1	1	0	Reserviert
1	1	1	CTS (als CLK_{in})

Bit 3 eines Clock-Select-Registers legt fest, was auf der RTS-Leitung des Kanals ausgegeben wird. Ist Bit 3 = 0, so wird der Zustand der RTS-Leitung des SCC des jeweiligen Kanals geliefert. Ist Bit 3 = 1, wird statt dessen das Signal am TRxC-Pin durchgeschaltet.

Bit 3	RTS-Ausgang verbunden mit
0	RTS-Pin des SCC des zugehörigen Kanals
1	TRxC-Pin des SCC des zugehörigen Kanals (als CLK_{out})

Die Bits 4 bis 7 sind reserviert und sollten auf 0 gesetzt werden.

[1] Um auf eines der 8 Clock-Select-Register CSR/A bis /H zugreifen zu können, muß vorher der Kanal (A = 0 bis H = 7) in das Channel-Select-Register CHR geschrieben werden.

Signalfluß zwischen Gate-Array und SCC

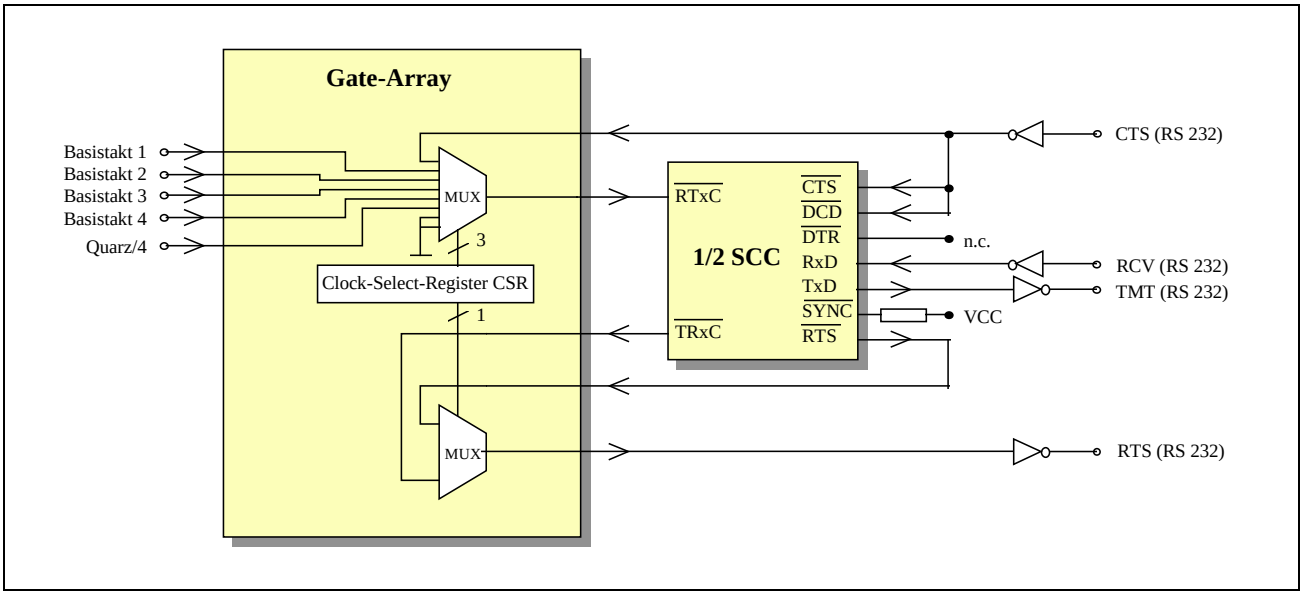


***Abb. 5-1**: Signalfluß zwischen Gate-Array und SCC (für einen Kanal)*

Anwahl einer Interrupt-Leitung zur Basiskarte

Das Modul ist interruptfähig, d.h. es kann bei bestimmten Ereignissen einen Interrupt auf der Basiskarte auslösen. Die Interrupt-Leitung des Moduls kann per Software mit einem der Interrupt-Eingänge der Basiskarte verbunden werden.

Während der Einstellung des Interrupt-Kanals darf das Modul keinen Interrupt anfordern bzw. auf der Basiskarte müssen die Interrupts (vorübergehend) maskiert werden.

Die Anwahl eines Interrupts geschieht durch Setzen des Interrupt-Select-Register ISR des Moduls (nur Bit 0 bis 2 werden benutzt, die restlichen Bits sind reserviert und sollten auf 0 gesetzt werden):

Bit 2	Bit 1	Bit 0	MODULAR-4/486
0	0	0	keine
0	0	1	IRQ-A
0	1	0	IRQ-B
0	1	1	IRQ-E
1	0	0	IRQ-C
1	0	1	IRQ-D

Programmierung der SCC-Bausteine

Die Programmierung der SCC-Bausteine erfolgt durch Schreiben bzw. Lesen der jeweiligen SCC-Register. Der Zugriff auf diese Register geschieht in zwei Schritten:

1. Ein Schreibzugriff auf das jeweilige SCC-Write-Register 0 (WR0) setzt den Register-Pointer dieses SCC-Kanals (Anwahl des zu lesenden bzw. zu schreibenden Registers).
2. Mit dem zweiten Zugriff auf die selbe Adresse wird dann der Inhalt eines Registers des angewählten Kanals gesetzt bzw. ausgelesen.

Zugriffe auf das Empfangsregister (SCC-RR8) und das Senderegister (SCC-WR8) eines Kanals benötigen nur einen Zugriff auf die jeweilige Adresse.

Programmierung der SCCs (allgemein)

Da dieser Baustein sehr komplex ist, wird hierzu auf die Literatur von Zilog (zum Baustein Z8530, Z85C30, und Z85230) bzw. Intel (82530) verwiesen. Dabei sind die Bausteine Z8530, Z85C30 und 82530 untereinander kompatibel, der Baustein Z85230 stellt eine verbesserte und erweiterte Version dar. Ein ausführliches Programmierhandbuch (in engl. Sprache) zu diesen Bausteinen ist bei SORCUS erhältlich (Best.-Nr.: MA-1529).

Bedingt durch die Konstruktion des Moduls M-COM-8 sind bei der Programmierung der SCCs einige Dinge zu berücksichtigen:

- Die Pins "DCD" der SCC Bausteine sind mit den CTS-Leitungen des jeweils selben Kanals verbunden.
- Die Pins "SYNC/A" und "SYNC/B" aller 8 Kanäle müssen als Eingänge konfiguriert werden. Sie haben keine Funktion und liegen über Pull-Up-Widerstände auf +5V Potential.

- Die "RTxC"-Pins der SCCs müssen als Eingänge konfiguriert werden. Sie werden nicht für einen Quarzoszillator eingesetzt (SCC-WR11, Bit 7 = 0 setzen, siehe SCC-Beschreibung). Die an diesen Pins liegenden Signale können innerhalb des SCCs für denselben Kanal für folgende Zwecke verwendet werden (SCC-WR11, Bit 5 und 6):

Signal	In-/Output	Verwendbar als
RTxC	Input	Receive Clock (max. 1/4 PCLK)
		Transmit Clock (max. 1/4 PCLK)
		Eingang der DPLL
		Eingang von Baudratengenerator

- Die Pins "TRxC/A" bis "TRxC/H" der SCCs müssen als Ausgänge konfiguriert werden (SCC-WR11, Bit 3 und 4).

Signal	In-/Output	Verwendbar als
TRxC	Output	Ausgang des Transmit Clock
		Ausgang des Baudratengenerators
		Ausgang der Receive DPLL
		Signal vom Pin "RTxC"

- Standardmäßig verfügt das Modul über einen Quarzoszillator von 24 MHz. An den "PCLK"-Pins der 4 SCCs liegt dieser Takt, geteilt durch einen einstellbaren Faktor (siehe Seite 5-17) an. Dieses PCLK-Signal kann z. B. auch als Eingangstakt für die Baudratengeneratoren der SCCs eingesetzt werden.

Programmierung der Interrupt-Vektoren der vier SCC-Bausteine

Die vier SCCs des Moduls sind für die Interrupt-Abwicklung in Form einer Daisy-Chain miteinander verbunden. Wenn ein SCC einen Interrupt anfordert, so signalisiert er dies dem Gate-Array über die gemeinsame /INT-Leitung der SCCs. Das Gate-Array leitet die Anforderung über die ausgewählte Interrupt-Leitung an die Basiskarte.

Das bedeutet, daß im Falle mehrerer Interrupt-Anforderungen zunächst der SCC mit der höchsten Priorität seinen Interrupt-Vektor an die Basiskarte ausgibt. Sobald dessen Interrupt-Anforderung bedient wurde, kommen die nachfolgenden Interrupts (mit niedrigerer Priorität) an die Reihe.

Eine Interrupt-Anforderung wird per Software (Interrupt-Service-Routine) durch einen Interrupt-Acknowledge-Zyklus bedient. Sobald die Interrupt-Acknowledge-Settle-Time abgelaufen ist, kann durch einen Lesezugriff auf einen beliebigen SCC der Interrupt-Vektor eingelesen werden.

Um die SCCs voneinander (und damit auch die Interrupt-Quellen) unterscheiden zu können, müssen vom Anwender zuvor die Interrupt-Vektor-Register der 4 SCCs unterschiedlich gesetzt worden sein (SCC-Write-Register WR2). Zusätzlich müssen Bit 2 (DLC) und Bit 4 (Status High/Low) in WR9 der SCCs gleich 0 gesetzt werden.

SCC Nummer	IC	Kanäle	Vektor (Wert in SCC-WR2)
0	4	A und B	0000 0000
1	5	C und D	0100 0000
2	6	E und F	1000 0000
3	7	G und H	1100 0000

Der Ablauf einer Interrupt-Anforderung und deren Behandlung sieht folgendermaßen aus:

1. Ein SCC fordert einen Interrupt an, das Gate-Array leitet die Interrupt-Anforderung an die Basiskarte weiter.
2. Die Interrupt-Service-Routine der CPU wird aufgerufen.
3. Durch einen I/O-Schreibzugriff auf einen SCC werden die INTAK-Pins der 4 SCCs des Moduls auf Low-Potential gelegt, damit beginnt der Interrupt-Acknowledge-Zyklus.
4. Der den Interrupt auslösende SCC mit der höchsten Priorität legt, nachdem die SCC-Daisy-Chain-Settle-Time (max. 1,6 µs) abgelaufen ist, den Interrupt-Vektor an das Gate-Array. Dort kann es durch einen I/O-Lesezugriff auf einen der 4 SCCs gelesen werden.
5. Durch das Lesen des Interrupt-Vektors gehen die INTAK-Pins wieder auf High-Potential. Die Interrupt-Service-Routine kann nun das Interrupt-Pending Register des auslösenden SCCs lesen (SCC-RR3).
6. Die Interrupt-Service-Routine muß die Interrupt-Quelle im SCC löschen.
7. Am Ende der Interrupt-Service-Routine muß noch ein Reset-IUS-Kommando an den SCC gesendet werden, um die Daisy-Chain freizugeben und Interrupts mit niedriger Priorität wieder zuzulassen.

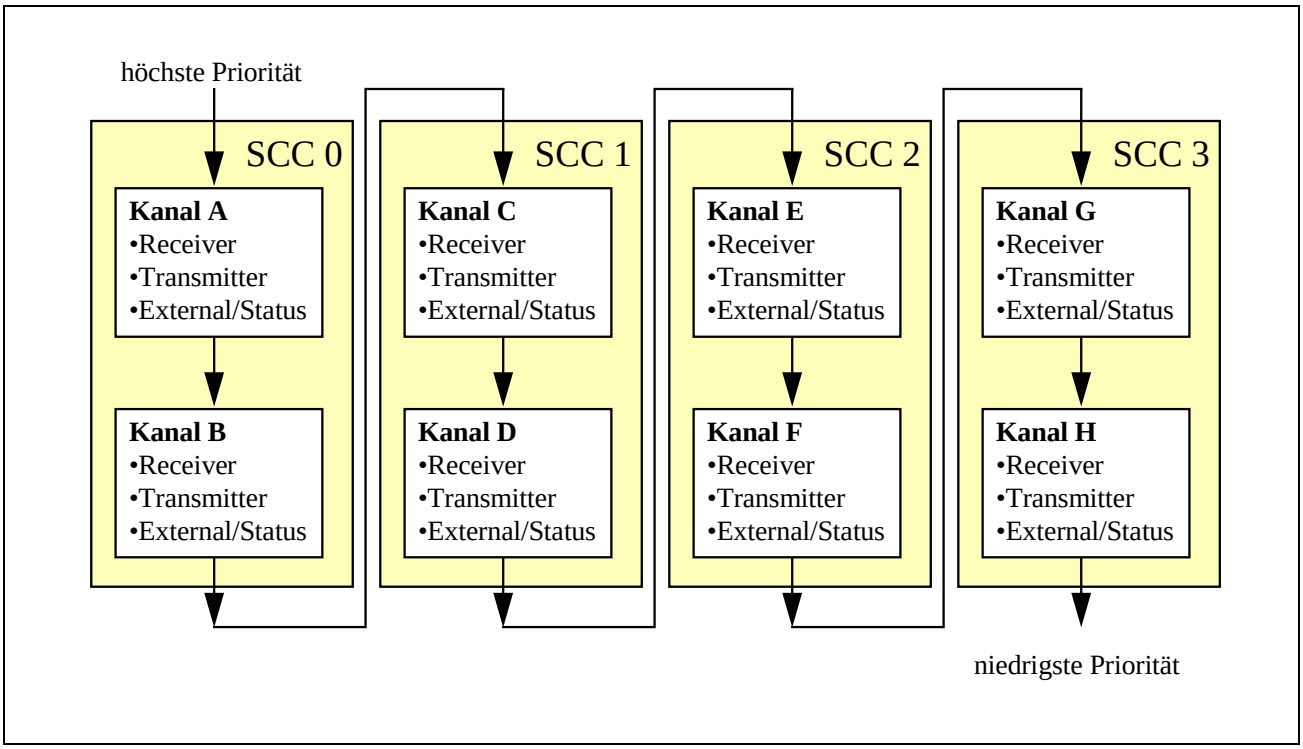


***Abb. 5-2**: Interrupt-Priorität durch Daisy-Chaining der SCCs*

Steckerbelegung St1

Pin	Signal	Kanal	Bedeutung
1	RxD/A	A	Receive Data
2	RTS/A	A	Request To Send
3	TxD/A	A	Transmit Data
4	CTS/A	A	Clear To Send
5	GND		Ground
6	RxD/B	B	Receive Datal
7	RTS/B	B	Request To Send
8	TxD/B	B	Transmit Data
9	CTS/B	B	Clear To Send
10	GND		Ground
11	RxD/C	C	Receive Data
12	RTS/C	C	Request To Send
13	TxD/C	C	Transmit Data
14	CTS/C	C	Clear To Send
15	GND		Ground
16	RxD/D	D	Receive Data
17	RTS/D	D	Request To Send
18	TxD/D	D	Transmit Data
19	CTS/D	D	Clear To Send
20	GND		Ground
21	RxD/E	E	Receive Data
22	RTS/E	E	Request To Send
23	TxD/E	E	Transmit Data
24	CTS/E	E	Clear To Send
25	GND		Ground
26	RxD/F	F	Receive Data
27	RTS/F	F	Request To Send
28	TxD/F	F	Transmit Data
29	CTS/F	F	Clear To Send
30	GND		Ground

Pin	Signal	Kanal	Bedeutung
31	RxD/G	G	Receive Data
32	RTS/G	G	Request To Send
33	TxD/G	G	Transmit Data
34	CTS/G	G	Clear To Send
35	GND		Ground
36	RxD/H	H	Receive Data
37	RTS/H	H	Request To Send
38	TxD/H	H	Transmit Data
39	CTS/H	H	Clear To Send
40	GND		Ground

Vorschlag für ein Anschlußkabel für einen Kanal an einen 9-poligen D-Submin.-Stecker mit IBM-AT-typischer Belegung:

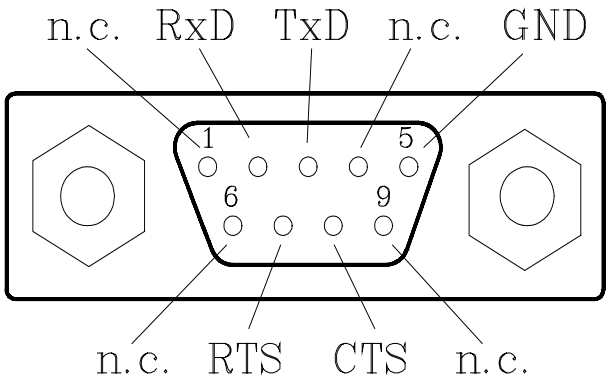


***Abb. 5-3**: AT-typische Steckerbelegung für M-COM-8*

Hochsprachenbibliothek

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M048_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Prozedur **m048_bib_startup** einmal aufgerufen werden.

m048_bib_startup — Initialisiere Modulbibliothek

Pascal PROCEDURE m048_bib_startup (micro_slot: byte);

C void EXPORT m048_bib_startup (byte micro_slot);

Funktion Diese Prozedur initialisiert die Modulbibliothek M048_LIB. Es werden u.a. die Initialisierungsdaten aus den EEPROMs aller M-COM-8 Module übernommen, die sich auf der Basiskarte befinden. Die Register des Gate-Arrays werden gemäß den EEPROM-Inhalten gesetzt.

m048_scc_reset — Führe SCC-Hardware-Reset durch

Pascal PROCEDURE m048_scc_reset (micro_slot, scc_num: byte);

C void EXPORT m048_scc_reset (byte micro_slot, byte scc_num);

Funktion Diese Prozedur führt einen Hardware-Reset eines SCC durch.

Parameter *scc_num*: Nummer des SCC (0 bis 3)

m048_scc_init — Initialisiere SCC-Kanal

Pascal: FUNCTION m048_scc_init (micro_slot, channel, mode, intmode, int_x_mode: byte; timeconst: word; use_brg, clkmode, databits, stopbits, parity_type: byte): byte;

C: byte EXPORT m048_scc_init (byte micro_slot, byte channel, byte mode, byte intmode, byte int_x_mode, ushort timeconst, byte use_brg, byte clkmode, byte databits, byte stopbits, byte parity_type);

Funktion: Diese Funktion initialisiert einen Kanal des Moduls.

Parameter: *channel*: Nummer des Kanals (0=A bis 7=H)

mode: Einstellung der Taktquellen für Receiver und Transmitter (entspricht WR11 eines SCCs, siehe SCC-Handbuch)

7	6	5	4	3	2	1	0	
								Clock-Source Select
						1	1	TRxC Output = DPLL Output
						1	0	TRxC Output = BRG [1] Output
						0	1	TRxC Output = Transmit Clock
						0	0	nicht erlaubt
					1			TRxC-Pin ist Output
					0			TRxC-Pin ist Input
			1	1				Transmit Clock = DPLL Output
			1	0				Transmit Clock = BRG [1] Output
			0	1				Transmit Clock = TRxC Pin
			0	0				Transmit Clock = RTxC Pin
	1	1						Receive Clock = DPLL Output
	1	0						Receive Clock = BRG [1] Output
	0	1						Receive Clock = TRxC Pin
	0	0						Receive Clock = RTxC Pin
0								TTL Signal an RTxC (muß = 0 sein)

Normalerweise werden der Receive- und der Transmit-Clock mit dem Baudratengenerator erzeugt (mode = 56h).

[1] BRG: Baudratengenerator

Es ist aber auch möglich den Takt über RTxC oder TRxC einzuspeisen.

int_mode: Interrupt-Betriebsart festlegen (entspricht WR 1 eines SCCs)

7	6	5	4	3	2	1	0	
								Interrupt Enable
0	0	0						Reserviert
			1	1				Rx Interrupt On Special Condition
			1	0				Interrupt On All Rx Character or Special Condition
			0	1				Rx Interrupt On First Character or Special Condition
			0	0				Rx Interrupt Disable
					1			Parity is Special Condition
						1		Tx Interrupt Enable
							1	Ext. Interrupt Enable

int_x_mode: Erweiterte Interrupt-Betriebsart festlegen (entspricht WR 15 eines SCCs)

7	6	5	4	3	2	1	0	
								Ext./Status Interrupt Select
1								Break/Abort Interrupt Enable
	1							Tx Underrun/EOM Interrupt Enable
		1						CTS Interrupt Enable
			1					Sync/Hunt Interrupt Enable
				1				DCD Interrupt Enable (= CTS)
					1			SDLC FIFO Enable
					0			Reserviert bei NMOS SCC
						1		Zero Count Interrupt Enable
							1	WR7' SDLC Feature Enable
							0	Reserviert bei CMOS/NMOS SCC

Soll z.B. ein Interrupt durch einen Signalwechsel der CTS Leitung ausgelöst werden, so müssen Bit-0 von *int_mode* und Bit 5 von *int_x_mode* gesetzt sein.

timeconst: Zeitkonstante für Baudratengenerator (0 bis ffffh)

clkmode: Clock-Mode (1, 16, 32, 64) für Receiver und Transmitter

use_brg: Baudratengenerator verwenden (1) oder nicht (0)

databits: Anzahl Datenbits (5 bis 8)

stopbits: Anzahl Stopbits (0 = 0, 1 = 1, 2 = 1,5 3 = 2 Stopbits)

parity_type: Parität (0 = keine Parität, 1 = odd, 2 = even)

m048_transmit_scc_character — Sende Zeichen

Pascal: FUNCTION m048_transmit_scc_character (micro_slot, channel, data: byte): byte;

C: byte EXPORT m048_transmit_scc_character (byte micro_slot, byte channel, byte data);

Funktion: Diese Funktion sendet ein Zeichen. Vor dem Senden wird der Zustand des Sendepuffers abgefragt. Ist der Sendepuffer voll (TBE = 0), so wird 1 zurückgeliefert. Wurde das Zeichen erfolgreich in den Sendepuffer geschrieben, liefert die Funktion 0 zurück.

Parameter:
- *channel*: Nummer des Kanals (0=A bis 7=H)
- *data*: Zu sendendes Datenbyte

m048_receive_scc_character — Empfange Zeichen

Pascal: FUNCTION m048_receive_scc_character (micro_slot, channel: byte; var data: byte): byte;

C: byte EXPORT m048_receive_scc_character (byte micro_slot, byte channel, byte *data);

Funktion: Diese Funktion liest den Empfangspuffer eines Kanals. Vor dem Empfangen wird der Zustand des Empfangspuffers abgefragt. Ist dieser leer (RBF = 0), so wird 1 zurückgeliefert. Wurde ein Zeichen empfangen, liefert die Funktion 0 zurück.

Parameter:
- *channel*: Nummer des Kanals (0=A bis 7=H)
- *data*: Empfangenes Datenbyte

m048_scc_rbf Frage Empfangspuffer-Status (RBF) ab

Pascal FUNCTION m048_scc_rbf (micro_slot, channel: byte): byte;

C byte EXPORT m048_scc_rbf (byte micro_slot, byte channel);

Funktion Diese Funktion prüft den Status des Empfangspuffers eines Kanals. Der Zustand des RBF-Bits wird zurückgegeben (RBF = 1: receive buffer full, RBF = 0: receive buffer empty).

Parameter *channel*: Nummer des Kanals (0=A bis 7=H)

m048_scc_tbe Frage Sendepuffer-Status (TBE) ab

Pascal FUNCTION m048_scc_tbe (micro_slot, channel: byte): byte;

C byte EXPORT m048_scc_tbe (byte micro_slot, byte channel);

Funktion Diese Funktion prüft den Status des Sendepuffers eines Kanals. Der Zustand des TBE-Bits wird zurückgegeben (TBE = 1: transmit buffer empty, TBE = 0: transmit buffer full).

Parameter *channel*: Nummer des Kanals (0=A bis 7=H)

m048_scc_cts Frage Clear To Send (CTS) ab

Pascal FUNCTION m048_scc_cts (micro_slot, channel: byte): byte;

C byte EXPORT m048_scc_cts (byte micro_slot, byte channel);

Funktion Diese Funktion liefert den aktuellen Zustand der CTS-Leitung eines Kanals.

Parameter *channel*: Nummer des Kanals (0=A bis 7=H)

m048_set_rts — Setze Request To Send (RTS)

Pascal PROCEDURE m048_set_rts (micro_slot, channel, rts: byte);

C void EXPORT m048_set_rts (byte micro_slot, byte channel, byte rts);

Funktion Diese Prozedur setzt die RTS-Leitung eines Kanals auf den angegebenen Wert.

Parameter *channel*: Nummer des Kanals (0=A bis 7=H)

rts: Zu setzender Wert (0 oder 1)

m048_scc_intak — Lies Interrupt-Vektor

Pascal PROCEDURE m048_scc_intak (micro_slot: byte; var int_vector, rr0, rr1, rr3: byte);

C void EXPORT m048_scc_intak (byte micro_slot, byte *int_vector, byte *rr0, byte *rr1, byte *rr3);

Funktion Diese Funktion kann dazu verwendet werden, nach einer SCC-Interrupt-Anforderung, den Interrupt-Acknowledge-Zyklus einzuleiten und den Interrupt-Vektor zu lesen.

Parameter *vector*: Interrupt-Vektor

Die Bits 7 und 6 des Interrupt-Vektors geben die Nummer des SCCs an, der den Interrupt ausgelöst hat. Bit 3 bis 1 (V3, V2 und V1) enthalten weitere Informationen über den aufgetretenen Interrupt. Die restlichen Bits sind reserviert.

rr0: Aktueller Zustand von SCC-Read-Register[1] 0 (Transmit/Receive und External Status)

rr1: Aktueller Zustand von SCC-Read-Register 1 (Special Receive Status)

rr3: Aktueller Zustand von SCC-Read-Register 3 (Interrupt Pending)

---

[1] Bedeutung der Read-Register siehe SCC-Handbuch.

m048_baudrate_calc Ermittle Einstellungen für Baudrate

Pascal
PROCEDURE m048_baudrate_calc (baudrate, quartz: longint;
var bauderror: longint; var basereg, clk_mode: byte;
var timeconst: word, var use_brg: byte);

C
void EXPORT m048_baudrate_calc (long baudrate, long quartz,
long *bauderror, byte *basereg, byte *clk_mode, ushort *timeconst,
byte *use_brg);

Funktion
Diese Prozedur ermittelt die notwendigen Einstellungen für eine gewünschte Baudrate bei einer bestimmten Frequenz des Quarzoszillators. Als Ergebnisse liefert die Prozedur die Einstellungen für das Basistaktregister des Gate-Arrays sowie die SCC-Parameter für den Baudratengenerator.

Parameter

baudrate: Gewünschte Baudrate in bd

quartz: Taktfrequenz des Quarzoszillators in Hz (Standard: 24 MHz)

bauderror: Absoluter Fehler des Basistaktes bzw. der Baudrate der sich bei dieser Einstellung ergibt

Der relative Fehler berechnet sich wie folgt:

$$\frac{bauderror - \text{Basistakt}[basereg]}{\text{Basistakt}[basereg]}$$

basereg: Einstellungen für das Basistaktregister des Gate-Arrays

timeconst: Einstellung für die Zeitkonstante des Baudratengenerators

clk_mode: Einstellung für Clock-Mode des Receivers/Transmitters

use_brg: Verwendung des Baudratengenerators (0=nein, 1=ja)

m048_set_channel — Konfiguriere Kanal

Pascal

FUNCTION m048_set_channel (micro_slot, channel: byte; baudrate: word; flag, scc_mode, scc_intmode, scc_int_x_mode, databits, stopbits, parity_type: byte; var bauderror: longint; var used_clkmode: byte): byte);

C

byte EXPORT m048_set_channel (byte micro_slot, byte channel, long baudrate, byte flag, byte mode, byte intmode, byte int_x_mode, byte databits, byte stopbits, byte parity_type, long *bauderror, byte *used_clkmode);

Funktion

Diese Funktion stellt eine Kombination aus **m048_baudrate_calc** und **m048_init_scc** dar (siehe auch Parameter dieser Funktionen).

Parameter

flag: Strategie zur Einstellung der gewünschte Baudrate

flag = 1-4: Es wird das Basistakt-Register (1, 2, 3 oder 4) zur Erzeugung der Baudrate verwendet.

flag = 0,5 und 6:
Es wird versucht, die Baudrate mit einem vorhandenen Basistakt zu erzeugen. Ist dies nicht möglich, wird ein freier Basistakt verwendet.

flag = 0: Die Funktion liefert einen Fehlercode 1 zurück, falls der Kanal bereits benutzt wird (Clock-Select-Register des Kanals ungleich 0).

flag = 5: Die Funktion liefert einen Fehlercode 2 zurück, falls kein freier Basistakt verfügbar ist.

flag = 6: Falls alle Basistakte bereits vergeben sind, wird zur Erzeugung der Baudrate der günstigste der vorhandenen Basistakte ausgewählt.

Programmierung mit I/O-Zugriffen

Dieses Kapitel ist für jene Anwender gedacht, die eigene Anwendungsprogramme für die Basiskarte MODULAR-4/486 schreiben wollen.

Lokale I/O-Adressen

Alle Adressen sind in hexadezimaler Schreibweise angegeben.

Adresse	Zugr.	Funktion
		SCC 0 (Kanal A und B)
MBA+00h	RW8 [1]	Kanal B, Control [2]
MBA+01h	RW8	Kanal B, Data [3]
MBA+02h	RW8	Kanal A, Control
MBA+03h	RW8	Kanal A, Data
		SCC 1 (Kanal C und D)
MBA+04h	RW8	Kanal D, Control
MBA+05h	RW8	Kanal D, Data
MBA+06h	RW8	Kanal C, Control
MBA+07h	RW8	Kanal C, Data
		SCC 2 (Kanal E und F)
MBA+08h	RW8	Kanal F, Control
MBA+09h	RW8	Kanal F, Data
MBA+0ah	RW8	Kanal E, Control
MBA+0bh	RW8	Kanal E, Data
		SCC 3 (Kanal G und H)
MBA+0ch	RW8	Kanal H, Control
MBA+0dh	RW8	Kanal H, Data
MBA+0eh	RW8	Kanal G, Control
MBA+0fh	RW8	Kanal G, Data

[1] RW8 = 8-Bit Schreibzugriffe und 8-Bit Lesezugriffe erlaubt

[2] Control: SCC-Register werden durch zwei aufeinanderfolgende Zugriffe gelesen bzw. gesetzt

[3] Data: Direkter Zugriff auf das Sende- oder das Empfangsregister (WR 8 bzw. RR 8)

Adresse	Zugr.	Funktion
MBA+10h	RW8	Kanal-Select-Register (CHR) setzen/lesen (nur die unteren 3 Bits werden verwendet, Werte 0 bis 7)
MBA+11h	RW8	Clock-Select-Register [1] (CSR, siehe Seite 5-19.) setzen/lesen
MBA+12h	RW8	Basistaktregister 1 (BTR/1) setzen/lesen (siehe Seite 5-17)
MBA+13h	RW8	Basistaktregister 2 (BTR/2) setzen/lesen
MBA+14h	RW8	Basistaktregister 3 (BTR/3) setzen/lesen
MBA+15h	RW8	Basistaktregister 4 (BTR/4) setzen/lesen
MBA+16h	RW8	Interrupt-Select-Register (ISR) setzen/lesen Bit 2, 1, und 0 bestimmen die Interrupt-Leitung: 0 0 0 keine Interrupt-Leitung angewählt 0 0 1 IRQ-A bzw. INT-1 0 1 0 IRQ-B bzw. INT-3 0 1 1 IRQ-E bzw. INT-4 1 0 0 IRQ-C bzw. INT-5 1 0 1 IRQ-D bzw. INT-7
MBA+17h	RW8	PCLK-Select-Register (PSR) setzen/lesen Bit 1 und 0 legen den PCLK-Takt für alle 4 SCCs fest: 0 0 PCLK = Quarzoszillator / 5 0 1 PCLK = Quarzoszillator / 4 1 0 PCLK = Quarzoszillator / 3 1 1 PCLK = Quarzoszillator / 2
MBA+18h	W8x [2]	Interrupt-Acknowledge-Zyklus starten (siehe Seite 5-22.)
MBA+00h	R8 [3]	Interrupt-Vektor lesen
MBA+18h	R8	Gate-Array-Version lesen Bit 0 bis 3 = Revisionsnummer Bit 4 bis 7 = Versionsnummer z.B.: 0010 0101 = Version 2, Revision 5

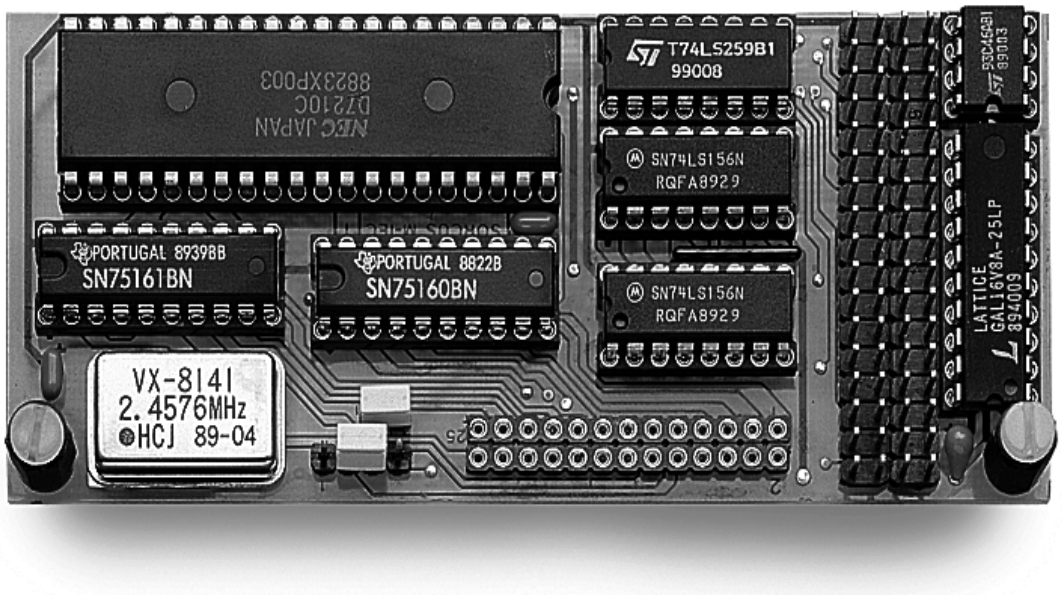
[1] Um das Clock-Select-Register lesen bzw. schreiben zu können, muß vorher über das Kanal-Select-Register CHR ein Kanal (0 bis 7, 0 = A, 7 = H) angewählt werden.

[2] W8x = 8-Bit Schreibzugriffe erlaubt, geschriebene Daten werden ignoriert

[3] R8 = 8-Bit Lesezugriffe erlaubt

6. M-IEC-1

IEC-Bus-System-Controller, Controller, Talker und Listener



Funktionsbeschreibung

Allgemeine Informationen zum IEC-Bus[1]

Der IEC-Bus besteht aus 16 Leitungen (8 Datenleitungen, 5 Steuerleitungen und 3 Handshake-Leitungen). Die Übertragung der Daten geschieht normalerweise im 7-Bit-ASCII-Code. Die drei Handshake-Leitungen ermöglichen die Anpassung der Übertragungsrate an das langsamste Gerät am Bus.

Der IEC-Bus bietet zwei grundlegende Möglichkeiten, Geräte miteinander zu verbinden:

1. Die einfachste Buskonfiguration beschränkt sich auf zwei Geräte, von denen das eine nur Daten sendet (Talk Only) und das andere nur Daten empfängt (Listen Only).
2. An den IEC-Bus sind zwei oder mehr Geräte angeschlossen, die sowohl senden als auch empfangen können sollen. Bei dieser Buskonfiguration muß mindestens ein Gerät angeschlossen sein, welches die Funktionen des Controllers bereitstellen kann. Sind mehrere Controller am IEC-Bus angeschlossen, so muß eines dieser Geräte als System-Controller deklariert werden, damit beim Einschalten der Geräte oder bei Störungen auf dem IEC-Bus kein "Chaos" entsteht. Es darf in **einer** Buskonfiguration nur **ein** Gerät als System-Controller deklariert sein.

Der jeweils aktive Controller steuert die anderen Geräte über die fünf Steuerleitungen IFC, REN, SRQ, ATN und EOI:

IFC - Interface Clear

Diese Leitung darf nur vom System-Controller aktiviert werden. Sie versetzt alle angeschlossenen Geräte in ihren Grundzustand, solange sie aktiv ist. Der IEC-Standard fordert eine minimale Aktivierungszeit von 100 µs.

[1] Verwiesen sei hier außerdem auf das Buch "IEC-Bus" von Anton Piotrowski (Franzis Verlag, ISBN 3-7723-6953-7), in dem auch alle nachfolgend verwendeten Abkürzungen und Begriffe beschrieben sind.

REN - Remote Enable

Diese Leitung teilt allen angeschlossenen Geräten mit, daß sie von nun an Kommandos über den IEC-Bus erhalten und nicht mehr von den geräteinternen Bedienungselementen. Auch diese Leitung wird nur vom System-Controller aktiviert. Diese Leitung wird normalerweise einmal vom System-Controller aktiviert und bleibt dann während des gesamten Betriebs aktiv.

SRQ - Service Request

Diese Leitung kann von jedem Gerät aktiviert werden, um dem Controller anzuzeigen, daß es Bedienung verlangt. Der Controller führt daraufhin ein paralleles und/oder serielles Pollen durch, um festzustellen, welches Gerät Bedienung anfordert.

ATN - Attention

EOI - End Or Identify

Diese beiden Leitungen bestimmen, welche Art von Information die Datenleitungen enthalten:

ATN	EOI	Funktion der Datenleitungen
0	0	Normale Übertragung von Daten
0	1	Übertragung des letzten Byte eines Datenblocks
1	0	Controller sendet ein Kommando
1	1	Paralleles Pollen

(1 = die entsprechende Leitung ist aktiv)

Die drei Handshake-Leitungen DAV, NRFD und NDAC haben folgende Funktionen:

DAV - Data Valid

Diese Leitung signalisiert den angeschlossenen Geräten, daß die Daten auf den DIO-Leitungen gültig sind.

NRFD - Not Ready For Data

Diese Leitung zeigt dem sendenden Gerät an, daß mindestens ein Gerät noch nicht bereit ist, Daten über den IEC-Bus zu empfangen.

NDAC - Not Data Accepted

Diese Leitung zeigt dem sendenden Gerät an, daß mindestens ein Gerät die gesendeten Daten noch nicht übernommen hat.

Die Kommandos des IEC-Bus

Der Controller besitzt die Möglichkeit, die Leitung ATN zu aktivieren und dadurch Kommandos über die Datenleitungen an die angeschlossenen Geräte zu senden. Bei diesen Kommandos werden nur die sieben niedrigwertigen Bit des Datenbus verwendet, der Zustand des achten Bit wird ignoriert. Die Kommandos des IEC-Bus werden in fünf Gruppen unterteilt:

ACG - Addressed Command Group	
Adressierte Kommandos	x 0 0 0 x x x x

Adressierte Kommandos sind nur für Geräte wirksam, an die zuvor ihre Listener-Adresse gesendet wurde.

- x 0 0 0 0 0 0 1 - GTL (Go To Local)

Durch dieses Kommando werden die als Listener adressierten Geräte aus dem Fernsteuerungszustand in den Eigensteuerungszustand überführt.

- x 0 0 0 0 1 0 0 - SDC (Selective Device Clear)

Dieses Kommando versetzt die als Listener adressierten Geräte in einen, von dem jeweiligen Gerät abhängigen, definierten Zustand (meistens den Einschaltzustand). Hierfür müssen die Geräte über die Interfacefunktion DC1 verfügen.

- x 0 0 0 0 1 0 1 - PPC (Parallel Poll Configure)

Hierdurch wird den als Listener adressierten Geräten mitgeteilt, daß das nächste Kommando das Sekundärkommando PPE oder PPD sein wird. Die Geräte müssen hierfür über die Interfacefunktion PP1 verfügen.

- x 0 0 0 1 0 0 0 - GET (Group Execute Trigger)

Dieses Kommando löst in den als Listener adressierten Geräten einen Trigger-Impuls aus. Voraussetzung hierfür ist allerdings, daß die Geräte über die Interfacefunktion DT1 verfügen.

- x 0 0 0 1 0 0 1 - TCT (Take Control)

  Mit diesem Kommando übergibt der aktive Controller die Kontrolle über den IEC-Bus an das adressierte Gerät, welches dafür mit einer der Interfacefunktionen C1 - C5 ausgestattet sein muß.

UCG - Universal Command Group	
Universelle Kommandos	x 0 0 1 x x x x

- x 0 0 1 0 0 0 1 - LLO (Local Lockout)

Durch dieses Kommando werden alle Geräte, die über die Interfacefunktion RL1 verfügen, in den Fernsteuerungszustand gebracht. Dieses Kommando hat eine ähnliche Funktion wie die nur vom System-Controller aktivierbare Leitung REN.

- x 0 0 1 0 1 0 0 - DCL (Device Clear)

Alle Geräte, die mit der Interfacefunktion DC1 oder DC2 ausgestattet sind, werden durch dieses Kommando in einen geräteabhängigen, definierten Zustand versetzt. Dieses Kommando hat eine ähnliche Funktion wie die nur vom System-Controller aktivierbare Leitung IFC.

- x 0 0 1 0 1 0 1 - PPU (Parallel Poll Unconfigure)

Dieses Kommando löscht bei allen Geräten die Konfiguration für das parallele Pollen.

- x 0 0 1 1 0 0 0 - SPE (Serial Poll Enable)

Den Geräten wird das serielle Pollen angekündigt. Das heißt, daß jedes nachfolgend als Talker adressierte Gerät keine normalen Daten, sondern das Statusbyte sendet, welches signalisiert, ob das entsprechende Gerät Bedienung wünscht oder nicht.

- x 0 0 1 1 0 0 1 - SPD (Serial Poll Disable)

  Hiermit wird das serielle Pollen beendet. Die Geräte senden wieder normale Daten, wenn sie als Talker adressiert werden.

LAG - Listener Address Group	
Listener-Adressen	x 0 1 a a a a a

- x 0 1 0 0 0 0 0 - LA0 (Listener Address 0)
  ⋮
  x 0 1 1 1 1 1 0 - LA30 (Listener Address 30)

Die Kommandos LA0 bis LA30 schalten das Gerät mit der entsprechenden Listener-Adresse aaaaa in den aktiven Listener-Zustand.

- x 0 1 1 1 1 1 1 - UNL (Unlisten)

Dieses Kommando schaltet alle aktiven Listener in den passiven Zustand.

TAG - Talker Address Group	
Talker-Adressen	x 1 0 a a a a a

- x 1 0 0 0 0 0 0 - TA0 (Talker Address 0)
  ⋮
  x 1 0 1 1 1 1 0 - TA30 (Talker Address 30)

Die Kommandos TA0 bis TA30 schalten das Gerät mit der zugehörigen Talker-Adresse als aktiven Talker ein.

- x 1 0 1 1 1 1 1 - UNT (Untalk)

  Dieses Kommando schaltet den gerade aktiven Talker in den passiven Zustand, so daß kein Gerät am IEC-Bus als aktiver Talker verbleibt und der Controller einen neuen Talker aktivieren kann.

SCG - Secondary Command Group	
Sekundäradressen bzw. Sekundärkommandos	x 1 1 x x x x x

Durch die IEC-Norm sind nur zwei spezielle Sekundärkommandos festgelegt. Für diese Sekundärkommandos muß das Primärkommando PPC vorausgegangen sein:

- x 1 1 0 S P P P - PPE (Parallel Poll Enable)

  Dieses Kommando gibt das adressierte Gerät für das parallele Pollen frei und legt gleichzeitig fest, auf welcher Leitung es sich mit welcher Polarität bei der Antwort auf das parallele Pollen melden soll. Die Polarität wird durch das mit S gekennzeichnete Bit definiert. Die durch PPP gekennzeichneten Bits legen die entsprechende Leitung nach folgendem Schema fest:

D3	D2	D1	verwendete Datenleitung
0	0	0	D1
0	0	1	D2
0	1	0	D3
0	1	1	D4
1	0	0	D5
1	0	1	D6
1	1	0	D7
1	1	1	D8

- x 1 1 1 x x x x - PPD (Parallel Poll Disable)

  Dieses Kommando sperrt das adressierte Gerät für das parallele Pollen. Mit diesem Befehl kann ein **einzelnes** Gerät aus den auf paralleles Pollen antwortenden herausgenommen werden (siehe auch Kommando PPU, welches bei **allen** Geräten die Konfiguration für das parallele Pollen löscht).

Funktionen des Moduls

Das Modul M-IEC-1 stellt eine komplette IEC-Bus-Schnittstelle zur Verfügung. Das Modul kann als System-Controller, Controller, Talker, Listener oder auch als Device konfiguriert werden.

Interface-Funktionen

Durch den IEC-625-Standard sind zehn Interface-Funktionen festgelegt. Durch diese Interface-Funktionen ist festgelegt, wie sich ein Gerät stand-alone oder in der Systemumgebung verhält. Es ist nicht notwendig, daß ein Gerät alle Funktionen beherrscht, und die meisten besitzen auch nur eine Untermenge von ihnen. Bei einigen Geräten sind die implementierten Funktionen neben dem Gerätestecker aufgedruckt.

Die häufig für die Funktionen verwendeten Abkürzungen sind in der folgenden Übersicht mit ihren Erklärungen zusammengestellt:

SH - Source Handshake

Diese Funktion besagt, daß ein Gerät Daten zu einem oder mehreren Listenern übertragen kann.

SH0 - kein Source Handshake
SH1 - Source Handshake

AH - Acceptor Handshake

Diese Funktion besagt, daß ein Gerät Daten von einem Talker empfangen kann.

AH0 - kein Acceptor Handshake
AH1 - Acceptor Handshake

T - Talker

TE - Extended Talker

Diese Funktion besagt, daß ein Gerät Daten und Statusinformationen senden kann, wenn es vom Controller als Talker adressiert wurde. Eine aus einem Byte bestehende Talker-Adresse adressiert einen Talker, eine aus zwei Byte bestehende einen Extended Talker. Funktionen von Extended Talkern sind in Klammern angegeben.

T0 - kein Talker oder Extended Talker
T1 - Talker, serielles Pollen, Talk Only Mode
T2 - Talker, serielles Pollen
T3 - Talker, Talk Only Mode
T4 - Talker
T5 - Talker, serielles Pollen, Talk Only Mode, Entadressieren bei MLA
(Entadressieren bei MSA und LPAS)
T6 - Talker, serielles Pollen, Entadressieren bei MLA
(Entadressieren bei MSA und LPAS)
T7 - Talker, Talk Only Mode, Entadressieren bei MLA
(Entadressieren bei MSA und LPAS)
T8 - Talker, Entadressieren bei MLA
(Entadressieren bei MSA und LPAS)

L - Listener

LE - Extended Listener

Diese Funktion besagt, daß ein Gerät Daten und Statusinformationen empfangen kann, wenn es vom Controller als Listener adressiert wurde. Eine aus einem Byte bestehende Listener-Adresse adressiert einen Listener, eine aus zwei Byte bestehende einen Extended Listener. Funktionen von Extended Listenern sind in Klammern angegeben.

L0 - kein Listener oder Extended Listener
L1 - Listener, Listen Only Mode
L2 - Listener
L3 - Listener, Listen Only Mode, Entadressieren bei MTA
(Entadressieren bei MSA und TPAS)
L4 - Listener, Entadressieren bei MTA
(Entadressieren bei MSA und TPAS)

SR - Service Request

Diese Funktion besagt, daß ein Gerät den Controller informieren kann, wenn es Bearbeitung wünscht. Der Service Request bleibt solange aktiv, bis das Gerät bearbeitet wurde.

SR0 - keine Service Request Funktion
SR1 - Service Request Funktion

RL - Remote/Local

Diese Funktion besagt, daß ein Gerät zwischen zwei Betriebsarten umgeschaltet werden kann: REMOTE-Betrieb (über den IEC-Bus) und LOCAL-Betrieb (über die Frontplatte).

RL0 - keine REMOTE/LOCAL-Programmierung
RL1 - komplette REMOTE/LOCAL-Programmierung
RL2 - kein Local Lockout

PP - Parallel Poll

Diese Funktion besagt, daß ein Gerät ein Bit zur Identifikation verwenden kann, wenn der Controller als Reaktion auf einen Service Request ein paralleles Pollen durchführt. Diese Funktion identifiziert ein Gerät nicht eindeutig, da mehrere Geräte dasselbe Bit verwenden können.

PP0 - keine Funktion für paralleles Pollen
PP1 - ferngesteuerte Konfigurierung für das parallele Pollen
PP2 - interne Konfigurierung für das parallele Pollen

DC - Device Clear

Diese Funktion besagt, daß ein Gerät in einen gerätespezifischen Grundzustand versetzt werden kann.

DC0 - kein Device Clear möglich
DC1 - alle Device Clear Möglichkeiten
DC2 - außer Selected Device Clear alle Device Clear Möglichkeiten

DT - Device Trigger

Diese Funktion besagt, das ein Gerät getriggert bzw. mit anderen Geräten synchronisiert werden kann.

DT0 - kein Device Trigger
DT1 - Device Trigger

C - Controller

Diese Funktion besagt, daß ein Gerät Daten- und Buskommandos senden und Geräte als Talker oder Listener adressieren kann. Weiterhin kann es serielles oder paralleles Pollen einleiten. An einem IEC-Bus können mehrere Controller angeschlossen sein, von denen aber nur einer zur Zeit aktiver Controller sein darf. Nur **ein** Controller am IEC-Bus darf System-Controller sein (mit der Fähigkeit IFC und REN zu aktivieren).

Controller werden folgendermaßen kategorisiert:

C0 - kein Controller
C1 - System Controller
C2 - IFC senden und Kontrolle übernehmen
C3 - REN senden
C4 - auf SRQ antworten
C5 - Interface-Befehle senden:
Kontrolle übernehmen
Kontrolle übergeben
Kontrolle an sich selbst übergeben
parallel Pollen
Kontrolle synchron übernehmen
C6 - verschiedene Controller-Funktionen, die sich aus Kombinationen der
⋮ obengenannten Funktionen ergeben
C28

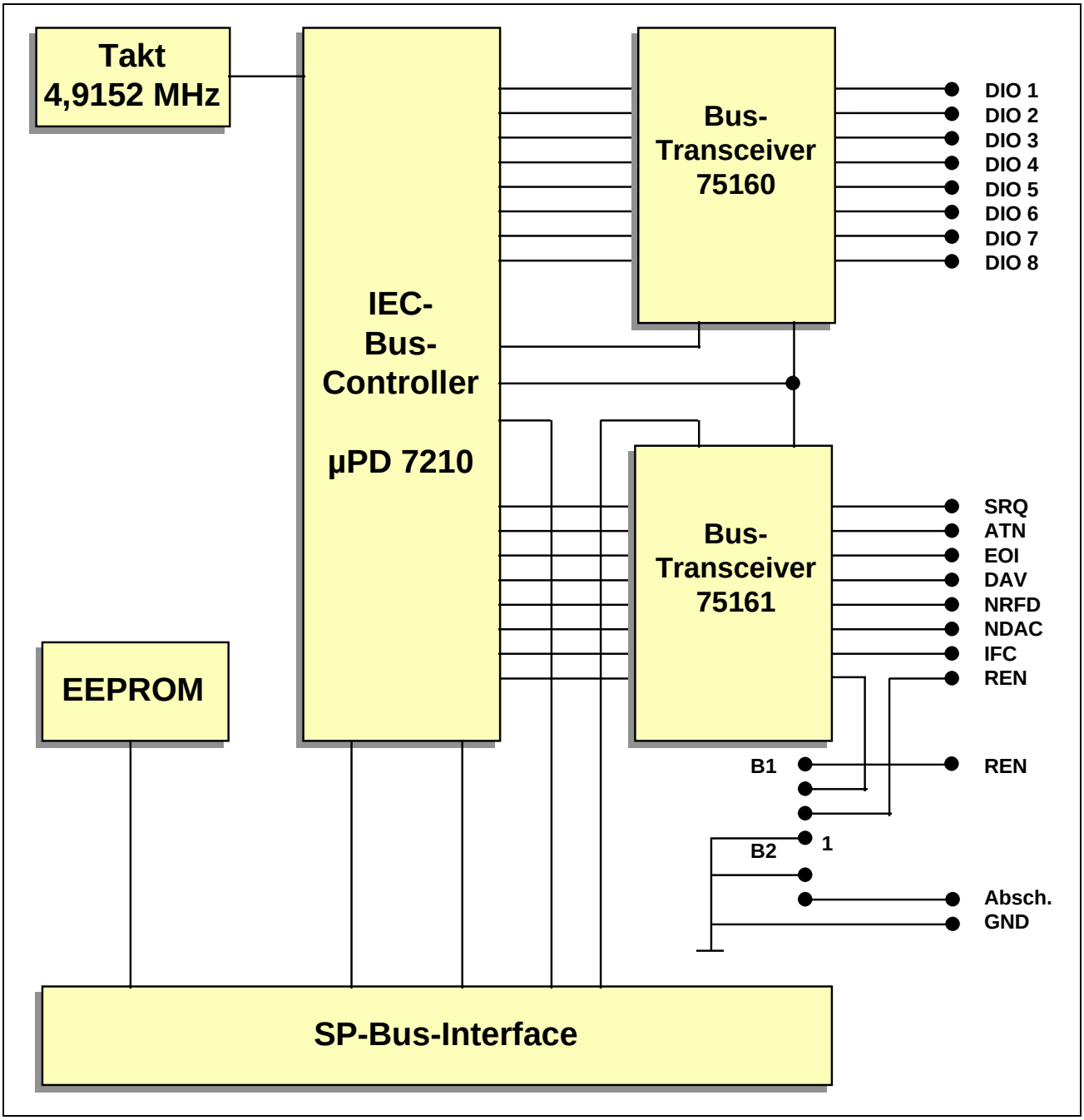
E - Art der im Gerät verwendeten Bustreiber

E1 - Open Collector Treiber
E2 - Tri-State Treiber
E1/2- Tri-State Treiber, die bei parallelem Pollen automatisch auf Open Collector Treiber umgeschaltet werden.

Das SPB-Modul M-IEC-1 stellt folgende Funktionen gemäß IEC-Norm zur Verfügung:

– SH1	– SR1	– DT1	– C3
– AH1	– RL1	– C0	– C4
– T5/TE5	– PP1/PP2	– C1	– C5
– L3/LE3	– DC1	– C2	– E1/2

Blockschaltbild



Technische Daten

Parameter	Wert	Einheit
Funktionen (gemäß IEC-Norm)	SH1, AH1, T5/TE5, L3/LE3, SR1, RL1, PP1/PP2, DC1, DT1, C0, C1, C2, C3, C4, C5, E1/2	-
Busanschluß	24-poliger Amphenol- oder 25-poliger D-Submin.-Stecker anschließbar	-
Funktion als System-Controller oder Device wählbar	ja	-
Maximale Datenübertragungsrate	ca. 300	KByte/s
Maximale Anzahl angeschlossener Geräte	15	-
Maximale Kabellänge des IEC-Bus	20	m
Maximale Kabellänge zwischen zwei Geräten	2	m
Betriebstemperatur, min./max.	0 / 70	°C
Abmessungen (L x B x H)	106 x 45 x 15	mm

Lieferumfang

- Modul M-IEC-1
- 26-poliger Pfostenstecker für Flachbandkabel
- Jumper (soweit erforderlich)
- Datenträger mit Programmbibliotheken (Pascal und C)

Konfiguration und Einbau

Auf dem Modul sind zwei Jumperfelder vorhanden, durch die u. a. die Belegung des Pfostensteckers an den gewünschten IEC-Bus-Stecker (Amphenol- oder D-Submin.-Stecker) angepaßt werden kann (siehe auch ab Seite 6-18).

Jumperfeld B1

Durch dieses Jumperfeld wird der verwendete Steckertyp eingestellt. Für einen 24-poligen Amphenol-Stecker müssen die Pins 2 und 3 verbunden werden, für einen 25-poligen D-Subminiatur-Stecker müssen die Pins 1 - 2 und 3 - 4 durch je einen Jumper verbunden sein.

Jumperfeld B2

Die Abschirmung des IEC-Bus-Kabels kann mit diesem Jumperfeld auf Masse gelegt werden.

24-poliger Amphenol-Stecker

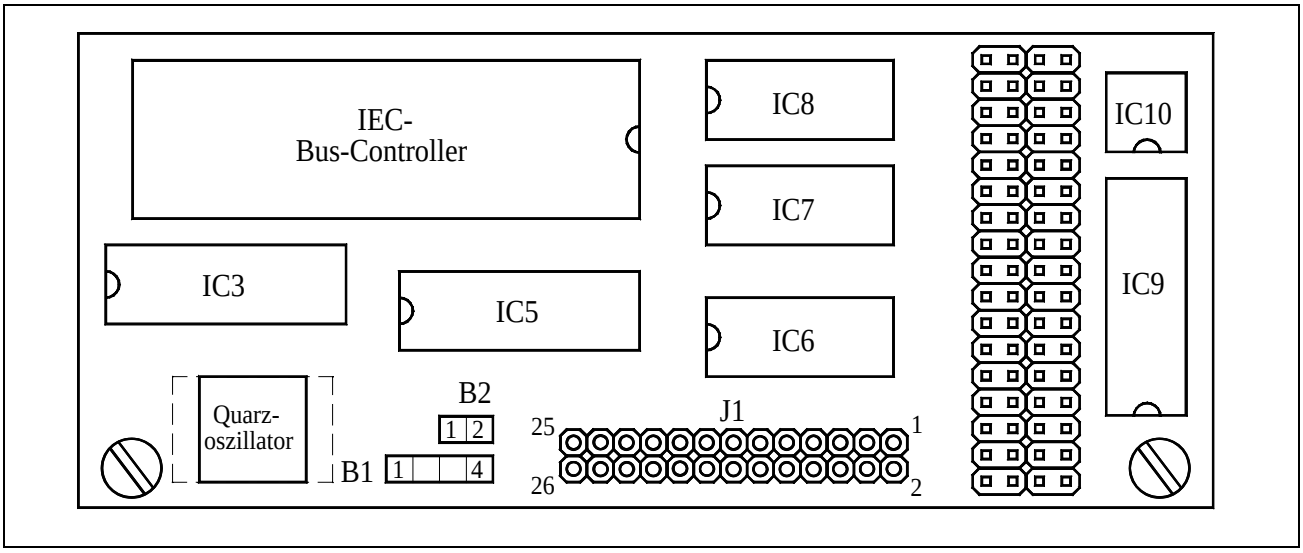
Ein 24-poliger Amphenol-Stecker kann direkt auf das Flachbandkabel aufgequetscht werden, wobei die Leitungen 25 und 26 des Kabels nicht mit eingequetscht werden und offen bleiben.

25-poliger D-Subminiatur-Stecker

Bei Verwendung eines 25-poligen D-Subminiatur-Steckers werden die Leitungen 25 und 26 zwischen die Leitungen 8 und 9 des Flachbandkabels gelegt und so an den D-Sub-Stecker gequetscht. Die Leitung 24 wird dabei nicht mit eingequetscht und bleibt offen. Die Leitungen des Flachbandkabels müssen also in folgender Reihenfolge am D-Subminiatur-Stecker ankommen:

1 bis 8, 25, 26, 9 bis 23

Lageplan



Die Größe des Quarzoszillators kann variieren.

EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0011	0001 0011	2313h	Modultyp M-IEC-1
1	0000 0000	0000 0001	0001h	Initialisierung
2	0000 0000	0000 1101	000dh	Betriebsarten
3	1110 0000	0000 0000	e000h	Device-Adressen
4	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	1	1	0	0	0	1	0	0	1	1	WORT-0: Kennung
								0	0	0	1	0	0	1	1	Modultyp: 19 = M-IEC-1
				0	0	1	1									Revision: 1 = A, 2 = B, 3 = C
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort kann eingestellt werden, ob das Modul nach dem Einschalten und bei einem Hardware-Reset entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															1	Init nach Hardware-Reset: 0 = nein, 1 = ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-2: Modes

Bit 0 und Bit 1: Diese Bits stellen den verwendeten Adreßmodus ein. Sie entsprechen den Bits ADM0 und ADM1 des Address-Mode-Registers des Controllers.

Bit 2: Dieses Bit legt fest, ob Datenübertragungen mit hoher oder geringer Geschwindigkeit geschehen.

Bit 3: Dieses Bit gibt an, ob das IEC-Bus-Interface der System-Controller ist oder nicht.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	WORT-2: Modes (werks. Einst.)
																geändert am: von:
														0	1	Address-Mode
													1			1 = High Speed, 0 = Low Speed
												1				1 = System-Controller, 0 = Device
0	0	0	0	0	0	0	0	0	0	0	0					Reserviert

WORT-3: Device-Adressen

Das obere und das untere Byte dieses Wortes enthalten jeweils eine der Device-Adressen, Informationen darüber, ob die entsprechende Adresse als Talker bzw. Listener-Adresse erkannt werden darf, und in welches der Adreßregister die Adresse geschrieben werden soll.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-3: Device-Adressen (werks. Einst.)
																geändert am: von:
											0	0	0	0	0	Adresse 0
										0						Listener-Adresse
									0							Talker-Adresse
								0								Select-Adreß-Register 0
			0	0	0	0	0									Adresse 1
		1														Listener-Adresse
	1															Talker-Adresse
1																Select-Adreß-Register 1

Steckerbelegung

Das Modul wird über einen 26-poligen (2 x 13) Steckverbinder und ein entsprechendes Flachbandkabel mit der Außenwelt verbunden.

Die Belegung des Steckers ist so gewählt, daß der 24-polige Amphenol-Stecker direkt auf das Flachbandkabel aufgequetscht werden kann und der 25-polige D-Subminiatur-Stecker durch das Umlegen von zwei Leitungen aufgequetscht werden kann (siehe dazu auch Beschreibung auf Seite 6-14).

Pfostenstecker

Pin	Signal
1	DIO1
2	DIO5
3	DIO2
4	DIO6
5	DIO3
6	DIO7
7	DIO4
8	DIO8
9	EOI
10	REN/GND
11	DAV
12	GND
13	NRFD
14	GND
15	NDAC
16	GND
17	IFC
18	GND
19	SRQ
20	GND
21	ATN
22	GND
23	Schirm
24	GND
25	***/REN
26	GND

D-Sub-Stecker

Pin	Signal
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	REN
6	EOI
7	DAV
8	NRFD
9	NDAC
10	IFC
11	SRQ
12	ATN
13	Schirm
14	DIO5
15	DIO6
16	DIO7
17	DIO8
18	GND
19	G EOI
20	G DAV
21	G NRFD
22	G NDAC
23	GND
24	G SRQ
25	G ATN
**********	

Amphenol-Stecker

Pin	Signal
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	EOI
6	DAV
7	NRFD
8	NDAC
9	IFC
10	SRQ
11	ATN
12	Schirm
13	DIO5
14	DIO6
15	DIO7
16	DIO8
17	REN
18	G DAV
19	G NRFD
20	G NDAC
21	G IFC
22	G SRQ
23	G ATN
24	GND
**********	
**********	

Bei den Angaben zum Pfostenstecker, Pin 10 und Pin 25, steht vor dem "/" die Belegung für den 24-poligen Amphenol-Stecker und hinter dem "/" die für den 25-poligen D-Subminiatur-Stecker (siehe auch Jumperfeld B1).

Hochsprachenbibliothek

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M019_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Prozedur **m019_bib_startup** einmal aufgerufen werden.

m019_bib_startup — Initialisiere Modulbibliothek

Pascal: PROCEDURE m019_bib_startup;

C: void EXPORT m019_bib_startup (void);

Funktion: Diese Prozedur initialisiert die Modulbibliothek. Es werden u. a. die Initialisierungsdaten aus den EEPROMs aller Module M-IEC-1 übernommen, die sich auf der Basiskarte befinden.

m019_set_conf_eeprom — Setze EEPROM-Konfiguration

Pascal: PROCEDURE m019_set_conf_eeprom (micro_slot: byte;
VAR err: byte);

C: void EXPORT m019_set_conf_eeprom (byte micro_slot, byte *err);

Funktion: Diese Prozedur setzt die Konfiguration so wie sie im EEPROM des Moduls angegeben ist. Die Default-Einstellungen nach dem Reset der Karte werden übernommen. Falls bereits mit der Prozedur **m019_set_conf_all** Werte festgelegt wurden, werden sie überschrieben.

Parameter: *err*:
- 1: Device ist kein System-Controller,
- 255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_set_conf_all Konfiguriere alle Funktionseinheiten

Pascal PROCEDURE m019_set_conf_all (micro_slot: byte; c: word; a: word; eoscode: byte; VAR err: byte);

C void EXPORT m019_set_conf_all (byte micro_slot, ushort c, ushort a, byte eoscode, byte *err);

Funktion Diese Prozedur konfiguriert alle Funktionseinheiten des Moduls. Die Parameter *c* und *a* enthalten die einzustellende Konfiguration. Die Codierung entspricht der der EEPROM-Wörter 2 und 3 des Moduls, wie ab Seite 6-16 dieser Beschreibung ausgeführt.

Parameter
c: Betriebsarten (siehe EEPROM-Wort 2)
a: Device-Adressen (siehe EEPROM-Wort 3)
eoscode: EOS-Zeichen
err: 1: Device ist kein System-Controller,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_get_conf_all Lies aktuelle Konfiguration

Pascal PROCEDURE m019_get_conf_all (micro_slot: byte; VAR confdata: word; VAR confaddr: word);

C void EXPORT m019_get_conf_all (byte micro_slot, ushort *confdata, ushort *confaddr);

Funktion Nach Aufruf dieser Prozedur enthalten die Variablen *confdata* und *confaddr* die in der Bibliothek gesetzten Werte der Konfiguration.

Parameter
confdata: Definition wie EEPROM-Wort 2
confaddr: Definition wie EEPROM-Wort 3

m019_write_aux Schreibe in Auxiliary-Mode-Register

Pascal PROCEDURE m019_write_aux (micro_slot: byte; par: byte);

C void EXPORT m019_write_aux (byte micro_slot, byte par);

Funktion Diese Prozedur schreibt das in *par* übergebene Byte in das Auxiliary-Mode-Register des Controllers auf dem Modul.

Parameter
par: Datenbyte

m019_put_command — Gib Kommando aus

Pascal
PROCEDURE m019_put_command (micro_slot: byte; wert: byte;
VAR err: byte);

C
void EXPORT m019_put_command (byte micro_slot, byte wert,
byte *err);

Funktion
Diese Prozedur gibt das in *wert* übergebene Command-Byte auf den Datenport des Controllers auf dem Modul aus. Danach wird abgewartet bis das Handshaking beendet ist.

Parameter
wert: Command-Byte
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_put_data — Gib Datenbyte aus

Pascal
PROCEDURE m019_put_data (micro_slot: byte; wert: byte;
VAR err: byte);

C
void EXPORT m019_put_data (byte micro_slot, byte wert, byte *err);

Funktion
Diese Prozedur gibt das in *wert* übergebene Datenbyte auf den Datenport des Controllers auf dem Modul aus. Danach wird abgewartet, bis das Handshaking beendet ist.

Parameter
wert: Gesendetes Datenbyte
err: 255: Bit 1 (DO) des Interrupt Status Registers 1 = 0,
sonst: OK

m019_get_data — Empfange Datenbyte

Pascal
PROCEDURE m019_get_data (micro_slot: byte; VAR wert: byte; VAR err: byte);

C
void EXPORT m019_get_data (byte micro_slot, byte *wert, byte *err);

Funktion
Diese Prozedur wartet bis ein Datenbyte vom Datenport des Controllers auf dem Modul verfügbar ist, liest es ein und gibt es in *wert* zurück.

Parameter
wert: Empfangenes Datenbyte
err: 255: Bit 1 (Do) des Interrupt Status Registers 0 = 0,
sonst: OK

m019_unlisten — Schalte Listener ab

Pascal
PROCEDURE m019_unlisten (micro_slot: byte; VAR err: byte);

C
void EXPORT m019_unlisten (byte micro_slot, byte *err);

Funktion
Mit dieser Prozedur wird allen am IEC-Bus angeschlossenen Geräten mitgeteilt, daß sie den Listen-Mode beenden sollen.

Parameter
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_address_listeners — Adressiere Listener

Pascal
PROCEDURE m019_address_listeners (micro_slot: byte;
anz: integer; VAR pararr; VAR err: byte);

C
void EXPORT m019_address_listeners (byte micro_slot, int anz,
byte *pararr, byte *err);

Funktion
Diese Prozedur schaltet alle bisherigen Listener ab und die ersten *anz* in *pararr* übergebenen Devices als Listener ein. Der Status wird in *err* zurückgegeben.

Parameter
anz: Anzahl einzuschaltender Listener
pararr: Adressen der Listener
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_address_talker — Adressiere Talker

Pascal
PROCEDURE m019_address_talker (micro_slot: byte; device1: byte; VAR err: byte);

C
void EXPORT m019_address_talker (byte micro_slot, byte device1, byte *err);

Funktion
Diese Prozedur schaltet den bisherigen Talker ab und das als *device1* übergebene Device als Talker ein.

Parameter
device1: Adresse des neuen Talker
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_interface_clear — Löse Interface-Clear aus

Pascal
PROCEDURE m019_interface_clear (micro_slot: byte;
VAR err: byte);

C
void EXPORT m019_interface_clear (byte micro_slot, byte *err);

Funktion
Mit dieser Prozedur werden alle am IEC-Bus angeschlossenen Geräte in den Einschaltzustand zurückversetzt.

Parameter
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

Hinweis
Diese Funktion kann nur vom System-Controller ausgelöst werden.

m019_remote_enable — Setze Remote-Enable-Leitung zurück

Pascal
PROCEDURE m019_remote_enable (micro_slot: byte; par: byte;
VAR err: byte);

C
void EXPORT m019_remote_enable (byte micro_slot, byte par,
byte *err);

Funktion
Diese Prozedur schaltet das REN-Signal des IEC-Bus um. Ist *par* = 1, dann wird REN aktiviert. Für *par* = 0 wird REN deaktiviert.

Parameter
par: 1: REN aktivieren
0: REN deaktivieren
err: 1: Device ist kein System-Controller,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

Hinweis
Diese Funktion steht nur dem System-Controller zur Verfügung.

m019_go_to_local — Gib Go-To-Local-Kommando aus

Pascal
PROCEDURE m019_go_to_local (micro_slot: byte; par: byte;
VAR err: byte);

C
void EXPORT m019_go_to_local (byte micro_slot, byte par,
byte *err);

Funktion
Diese Prozedur schaltet für *par* = 255 alle als Listener adressierten Devices in die LOCAL-Betriebsart. Alternativ kann auch eine bestimmte Device-Adresse in *par* übergeben werden. Dann werden alle bisherigen Listener deaktiviert und nur das angegebene Device wird in die LOCAL-Betriebsart geschaltet.

Parameter
par: Device-Adresse (255 = alle Listener)
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_write_eos — Lege EOS-Zeichen fest

Pascal
PROCEDURE m019_write_eos (micro_slot: byte; par: byte);

C
void EXPORT m019_write_eos (byte micro_slot, byte par);

Funktion
Diese Prozedur schreibt das in *par* übergebene Byte in das EOS-Register des Controllers auf dem Modul. Damit wird das Zeichen definiert, welches das Ende einer Datenübertragung über den IEC-Bus anzeigt (End Of String).

Parameter
par: EOS-Zeichen

m019_local_lockout — Gib Local-Lockout-Kommando aus

Pascal
PROCEDURE m019_local_lockout (micro_slot: byte;
VAR err: byte);

C
void EXPORT m019_local_lockout (byte micro_slot, byte *err);

Funktion
Diese Prozedur sendet das Universalkommando Local Lockout auf den IEC-Bus. Der Status wird in *err* zurückgeliefert.

Parameter
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_selected_device_clear Setze Gerät/e zurück

Pascal PROCEDURE m019_selected_device_clear (micro_slot: byte; par: byte; VAR err: byte);

C void EXPORT m019_selected_device_clear (byte micro_slot, byte par, byte *err);

Funktion Ist *par* = 255, so wird von dieser Prozedur das universelle Kommando Device Clear auf den IEC-Bus gesendet, was alle angeschlossenen Devices in ihren Grundzustand versetzt. Ist *par* = 254, so wird das adressierte Kommando Selected Device Clear auf den IEC-Bus gesendet, was alle als Listener adressierten Devices in ihren Grundzustand versetzt. Weiterhin besteht die Möglichkeit nur ein bestimmtes Device in *par* anzugeben, an welches dann Selected Device Clear übermittelt wird.

Parameter
par: Device-Adresse,
255: alle Devices,
254: alle Listener in den Grundzustand versetzen
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_parallel_poll_configure Konfiguriere paralleles Pollen

Pascal PROCEDURE m019_parallel_poll_configure (micro_slot: byte; par: byte; kon: byte; VAR err: byte);

C void EXPORT m019_parallel_poll_configure (byte micro_slot, byte par, byte kon, byte *err);

Funktion Diese Prozedur sendet an die als Listener adressierten Devices das Konfigurationsbyte *kon*. Ist allerdings der Parameter *par* = 255 dann wird das universelle Kommando Parallel Poll Unconfigure übermittelt.

Parameter
par: 255: an alle Listener Konfigurationsbyte senden,
sonst: an alle Listener PPU senden
kon: Konfigurationsbyte
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_group_execute_trigger — Löse Trigger aus

Pascal
PROCEDURE m019_group_execute_trigger (micro_slot: byte;
VAR err: byte);

C
void EXPORT m019_group_execute_trigger (byte micro_slot,
byte *err);

Funktion
Diese Prozedur sendet an alle als Listener adressierten Devices das adressierte Kommando Group Execute Trigger, was üblicherweise einen Triggerimpuls in den entsprechenden Geräten auslöst.

Parameter
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_take_control — Übergib Kontrolle

Pascal
PROCEDURE m019_take_control (micro_slot: byte; par: byte;
VAR err: byte);

C
void EXPORT m019_take_control (byte micro_slot, byte par,
byte *err);

Funktion
Diese Prozedur übergibt die Kontrolle über den IEC-Bus an das in *par* übergebene Device. Das Device muß, um die Kontrolle zu übernehmen, mit einer Controller-Funktion ausgestattet sein.

Parameter
par: Device-Adresse
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_execute_parallel_poll Führe paralleles Pollen durch

Pascal PROCEDURE m019_execute_parallel_poll (micro_slot: byte; VAR ppr: byte; VAR err: byte);

C void EXPORT m019_execute_parallel_poll (byte micro_slot, byte *ppr, byte *err);

Funktion Diese Prozedur führt ein paralleles Pollen durch. Die Antwort steht in *ppr*.

Parameter *ppr*: Resultat des parallelen Pollens
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_execute_serial_poll Führe serielles Pollen durch

Pascal PROCEDURE m019_execute_serial_poll (micro_slot: byte; adr: byte; VAR spr: byte; VAR err: byte);

C void EXPORT m019_execute_serial_poll (byte micro_slot, byte adr, byte *spr, byte *err);

Funktion Diese Prozedur pollt das in *adr* angegebene Device seriell. Die Antwort steht in *spr*.

Parameter *spr*: Ergebnis des seriellen Pollens
err: 8: OK,
255: Fehler bei der Kommunikation

m019_transfer_data — Löse Datentransfer aus

Pascal
PROCEDURE m019_transfer_data (micro_slot: byte; par: byte;
VAR err: byte);

C
void EXPORT m019_transfer_data (byte micro_slot, byte par,
byte *err);

Funktion
Diese Prozedur gibt den IEC-Bus für eine Datenübertragung vom in *par* übergebenen Talker-Device zu den adressierten Listener-Devices frei.

Parameter
par: Talker-Device-Adresse
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_send_data — Sende Daten an die Listener

Pascal
PROCEDURE m019_send_data (micro_slot: byte; anz: integer;
VAR pararr; VAR err: byte);

C
void EXPORT m019_send_data (byte micro_slot, int anz,
byte *pararr, byte *err);

Funktion
Diese Prozedur sendet die in *pararr* enthaltenen *anz* Byte als Daten an alle als Listener adressierten Devices.

Parameter
anz: Anzahl zu sendender Daten
pararr: Datenarray
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_receive_data — Empfange Daten vom Talker

Pascal PROCEDURE m019_receive_data (micro_slot: byte; VAR blk; VAR err: byte);

C void EXPORT m019_receive_data (byte micro_slot, byte *blk, byte *err);

Funktion Diese Prozedur empfängt einen Datenblock von dem als Talker adressierten Device. Die empfangenen Daten werden in *blk* zurückgeliefert.

Parameter *blk*: Datenarray
err: 8: OK,
255: Bit 3 (CO) des Interrupt Status Registers 2 = 0.

m019_wait_di — Erwarte Datenempfang

Pascal PROCEDURE m019_wait_di (micro_slot: byte; VAR err: byte);

C void EXPORT m019_wait_di (byte micro_slot, byte *err);

Funktion Diese Prozedur wartet, bis das DI-Bit im Interrupt-Status-Register 1 des Controllers auf dem Modul gesetzt ist.

Parameter *err*: 255: Timeout
sonst: OK

m019_wait_do — Warte, bis Datenausgabe möglich

Pascal PROCEDURE m019_wait_do (micro_slot: byte; VAR err: byte);

C void EXPORT m019_wait_do (byte micro_slot, byte *err);

Funktion Diese Prozedur wartet, bis das DO-Bit im Interrupt Status Register 1 des Controllers auf dem Modul gesetzt ist.

Parameter *err*: 255: Timeout,
sonst: OK

m019_wait_co Warte, bis Kommandoausgabe möglich

Pascal PROCEDURE m019_wait_co (micro_slot: byte; VAR err: byte);

C void EXPORT m019_wait_co (byte micro_slot, byte *err);

Funktion Diese Prozedur wartet, bis das CO-Bit im Interrupt Status Register 2 des Controllers auf dem Modul gesetzt ist.

Parameter *err*: 255: Timeout
sonst: OK

Programmierung mit I/O-Zugriffen

Lokale I/O-Adressen

Adresse	Zugr.	Funktion
MBA+00h	RW8	Data
MBA+01h MBA+02h	RW8 RW8	Interrupt Status/Mask 1 Interrupt Status/Mask 2
MBA+03h	RW8	Serial Poll Status/Mode
MBA+04h	RW8	Address Status/Mode
MBA+05h	R8	Command Pass Through
MBA+05h	W8	Auxiliary Mode
MBA+06h MBA+06h MBA+07h	R8 W8 R8	Address 0 Address 0/1 Address 1
MBA+07h	W8	End Of String

Das Modul kann auf zwei Arten Interrupts auf der Basiskarte auslösen. Für jede steht eine eigene Interrupt-Leitung vom Modul zur Basiskarte zur Verfügung, die per Software mit einer der Interrupt-Eingänge der Basiskarte verbunden werden kann. Die eine Interrupt-Leitung ("INT-1") wird durch einen Interrupt-Request des IEC-Bus Controllers aktiviert, die andere ("INT-2") wird durch einen DMA-Request des IEC-Bus Controllers aktiviert.

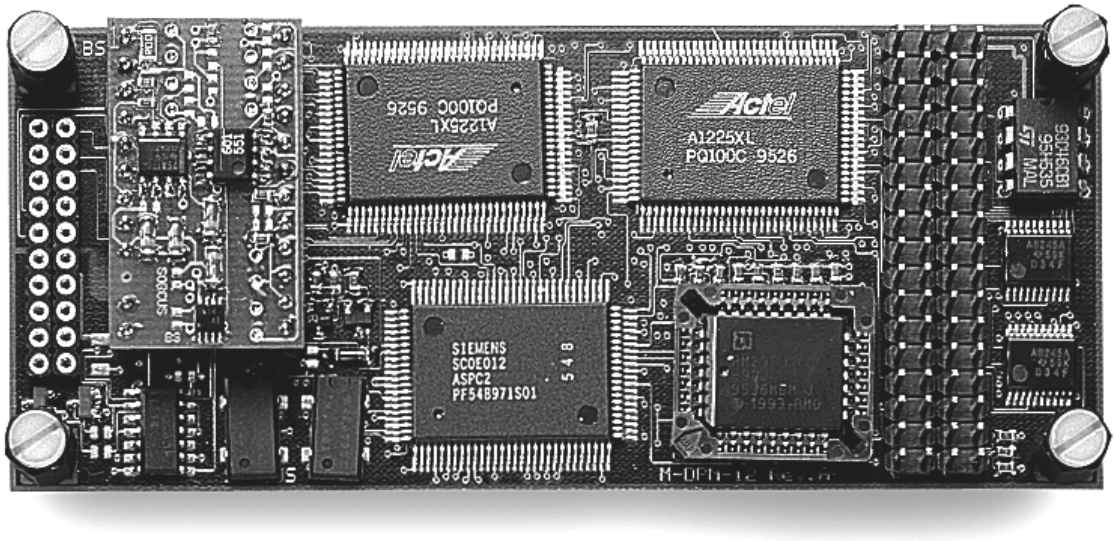
Um eine Leitung mit einem IRQ-x Eingang der Basiskarte zu verbinden, muß an drei 486 I/O-Adressen jeweils ein Dummy-Byte gesendet werden:

IRQ-x der Basiskarte	INT-1-Leitung	INT-2-Leitung
IRQ-A	MBA+11h, MBA+1bh, MBA+15h	MBA+13h, MBA+1dh, MBA+1fh
IRQ-B	MBA+11h, MBA+1ah, MBA+15h	MBA+13h, MBA+1ch, MBA+1fh
IRQ-E	MBA+10h, MBA+1ah, MBA+15h	MBA+12h, MBA+1ch, MBA+1fh
IRQ-C	MBA+10h, MBA+1bh, MBA+14h	MBA+12h, MBA+1dh, MBA+1eh
IRQ-D	MBA+10h, MBA+1bh, MBA+15h	MBA+12h, MBA+1dh, MBA+1fh

Notizen

7. M-DPM-12

PROFIBUS DP Master bis 12 MBaud



Programmierung 7-13

Hochsprachenbibliotheken 7-24

Inbetriebnahme 7-52

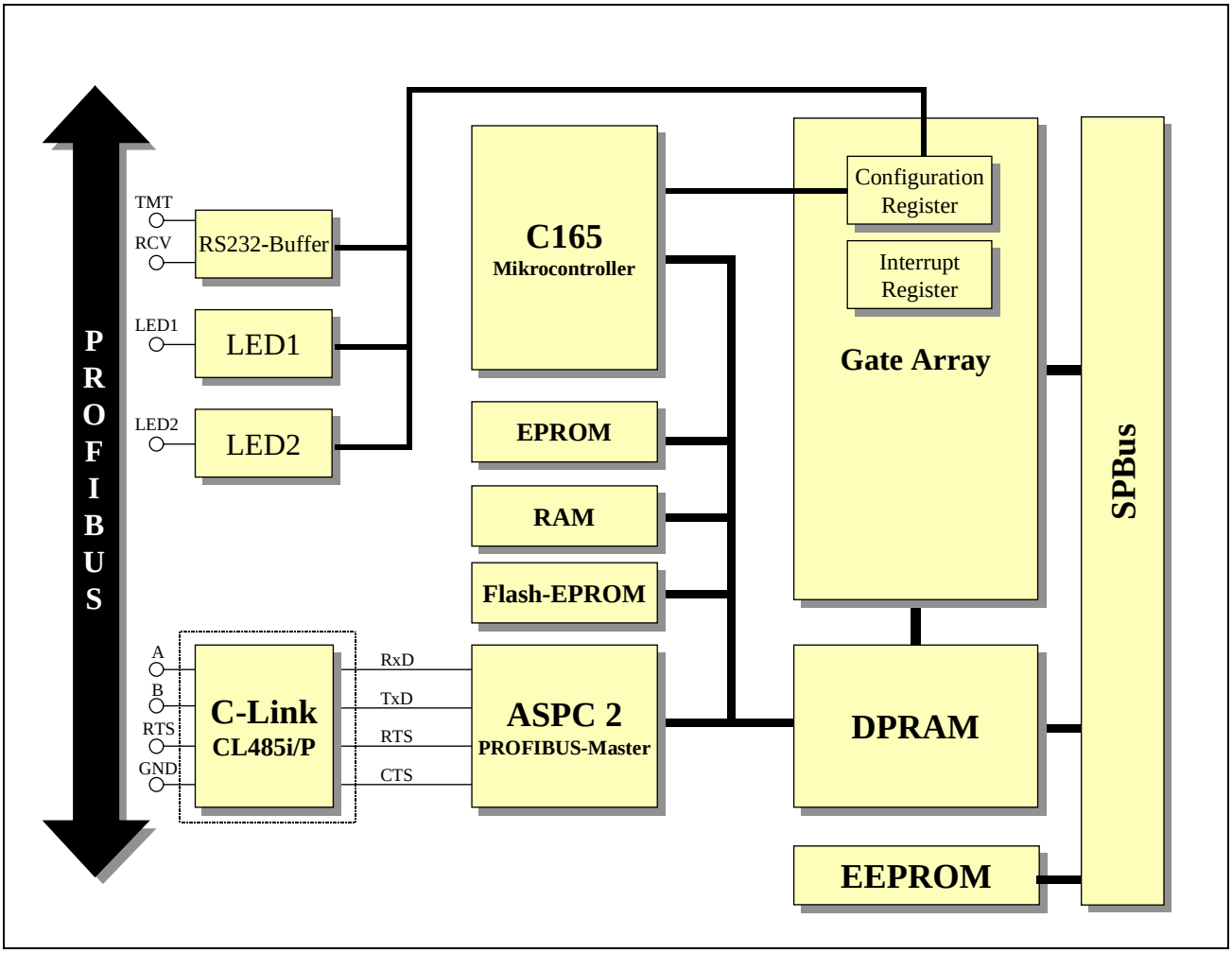
Programmierung mit I/O-Zugriffen 7-55

Funktionsbeschreibung

M-DPM-12 ist ein intelligentes PROFIBUS-DP Master-Modul für SORCUS MODULAR-4/486-Basiskarten. Es werden alle Baudraten inkl. 12 MBaud unterstützt. Als Schnittstelle zwischen der lokalen Intelligenz (C165 Mikrocontroller) und der MODULAR-4/486 dient ein Dual-Port-RAM (DPRAM) über das Befehle und Daten ausgetauscht werden. Die physikalische PROFIBUS-Schnittstelle wird per C-Link (Standard: RS-485, galvanisch getrennt) aufgesteckt.

Vor Inbetriebnahme des Moduls muß das C-Link in den dafür vorgesehenen Steckplatz eingesetzt werden, da zum Betrieb des Moduls das C-Link unbedingt erforderlich ist (auch wenn das Modul nicht an den PROFIBUS angeschlossen ist). **!**

Blockschaltbild



Technische Daten

Parameter		Wert	Einheit
PROFIBUS Controller		Siemens ASPC 2	-
Mikrocontroller 80C165		20	MHz
Dual Ported RAM		2 x 8	KByte
Statisches RAM		2 x 128	KByte
EPROM		512	KByte
Flash-Memory		512	KByte
Serielles EEPROM für Konfigurationsdaten		32	Wörter
Interruptfähig zur Basiskarte (Interrupt-Kanal per Software anwählbar)		ja	-
Versorgungsspannungen[1] (von der Basiskarte)		+5, ±12	V
Stromaufnahme (typ., extern nichts angeschlossen, LED 1 und 2 aus, mit C-Link CL485i/P):	+5 V	190	mA
	+12 V[1]	4	mA
	-12 V[1]	4	mA
Betriebstemperaturbereich		0 bis 60	°C
Abmessungen (L x B x H)		106 x 45 x 15	mm

Lieferumfang

Zum Lieferumfang des Moduls gehören:

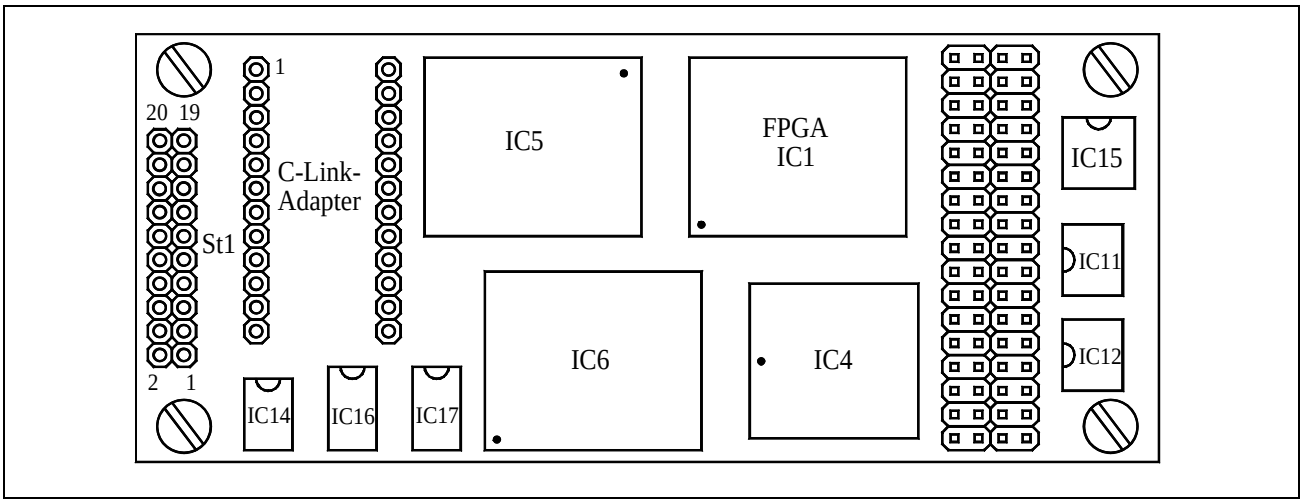
- Modul M-DPM-12
- 20-poliger Pfostenstecker für Flachbandkabel
- Datenträger mit Programmbibliotheken

[1] ±12 V werden nur für die RS232-Schnittstelle benutzt.

Konfiguration und Einbau

Vor dem Aufstecken des Moduls auf die Basiskarte muß der C-Link-Adapter aufgesteckt werden (Pin 1 ist auf dem Modul und auf dem C-Link gekennzeichnet). Das Modul und das C-Link enthalten keine Jumper, alle Einstellungen werden nach dem Einbau per Software vorgenommen.

Lageplan



EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0010	0010 1100	222ch	Modultyp M-DPM-12
1	0000 0000	0000 0000	0000h	Initialisierung
2	0000 0000	0000 0000	0000h	Interrupt-Kanal zur Basiskarte
3	0000 0000	0000 0000	0000h	Gate-Array Konfiguration
4	0000 0000	0000 0000	0000h	LED-Einstellung
5	1001 0101	1111 0000	95f0h	Gate-Array Konfiguration
6	0001 0101	1111 0000	15f0h	Gate-Array Konfiguration
7	0000 0000	0000 0000	0000h	Timeout-Counter
8	0000 0000	0000 0000	0000h	Gate Array Version/Revision (IC 2)
9	0000 0000	0000 0010	0002h	ASPC 2-Version (IC 6)
10	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

! *Die EEPROM-Inhalte der Wörter 2, 4 und 7 dienen zur Abspeicherung einer anwenderspezifischen Modulkonfiguration[1]. Die EEPROM-Inhalte werden nicht direkt (per Hardware) in die entsprechenden Registern des Moduls übernommen, sondern können vom Anwenderprogramm gelesen und zur Programmierung der Register verwendet werden.*

[1] Siehe Abschnitt Programmierung.

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	1	0	0	0	1	0	1	1	0	0	WORT-0: Kennung
								0	0	1	0	1	1	0	0	Modultyp (2ch = 44 = M-DPM-12)
				0	0	1	0									Revision: 1 = Rev. A, 2 = Rev. B
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

Hier hat nur Bit 0 z. Zt. eine Bedeutung. Wenn dieses Bit = 1 gesetzt ist, werden die Register des Moduls nach einem Hardware-Reset entsprechend den Daten in EEPROM konfiguriert und initialisiert.
Wenn Bit 0 = 0 gesetzt ist, wird das Modul nach einem Hardware-Reset (bzw. nach dem Einschalten des Systems) nicht automatisch konfiguriert und initialisiert.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															0	Init nach Hard-Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-2: Interrupt-Kanal zur Basiskarte

Hier kann der Anwender den Interrupt-Kanal der Basiskarte abspeichern, mit dem das Modul verbunden wird.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-2: Interrupt-Kanal (werks. Einst.)
																geändert am: von:
													0	0	0	Interrupt-Kanal 0 = kein Interrupt 1 = IRQ-A 2 = IRQ-B 3 = IRQ-C 4 = IRQ-D 5 = IRQ-E 6 = IRQ-F 7 = NMI
												0				Reserviert
											0					XINTH-Interrupt Enable 0 = Interrupt disabled 1 = Interrupt enabled
										0						Konsistenzkonflikt-Erkennnung 0 = per Timeout-Zähler 1 = per Interrupt
									0							Firmware-Timer Interrupt 0 = Interrupt disabled 1 = Interrupt enabled
0	0	0	0	0	0	0	0	0								Reserviert

WORT-3: Gate-Array Konfiguration (darf nicht geändert werden!)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-5: Gate-Array Konfiguration
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Gate-Array Konfiguration

Hier ist **werkseitig** die Gate-Array Konfiguration abgespeichert.

WORT-4: LED 1 und 2

Hier kann der Sollzustand der Leuchtdioden LED 1 und 2 nach Reset abgespeichert werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-4: LED 1 und 2 (werks. Einst.)
																geändert am: von:
															0	LED 1 0 : aus, 1 : ein
														0		LED 2 0 : aus, 1 : ein
0	0	0	0	0	0	0	0	0	0	0	0	0	0			Reserviert

WORT-5: Gate-Array Konfiguration (darf nicht geändert werden!)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	WORT-5: Gate-Array Konfiguration
1	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	Gate-Array Konfiguration

Hier ist **werkseitig** die Gate-Array Konfiguration abgespeichert.

WORT-6: Gate-Array Konfiguration (darf nicht geändert werden!)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	WORT-6: Gate-Array Konfiguration
0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	Gate-Array Konfiguration

Hier ist **werkseitig** die Gate-Array Konfiguration abgespeichert.

WORT-7: Initialisierungswert für den Timeout-Counter

Hier kann der Anwender den Timeout-Counter-Wert (in Abhängigkeit von der PROFIBUS-Baudrate) für die Konsistenzsteuerung abspeichern.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-7: Timeout-Counter (werks. Einst.)
																geändert am: von:
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16 Bit Timer-Wert

WORT-8: Gate-Array Version (IC 2, nicht vom Anwender zu ändern)

Hier ist **werkseitig** die Version des Gate-Arrays (IC 2) abgespeichert.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	WORT-8: CPLD Version
												0	0	0	0	Revision
								0	0	0	1					Version
0	0	0	0	0	0	0	0									Reserviert

WORT-9: ASPC 2 Version (IC 6, nicht vom Anwender zu ändern)

Hier ist **werkseitig** die Version des ASPC 2 ASICs (IC 6) abgespeichert.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	WORT-9: ASPC 2 Version
												0	0	1	0	Step (0 = A, 1 = B, 2 = C, ...)
								0	0	0	0					Revision
0	0	0	0	0	0	0	0									Reserviert

Steckerbelegung

Pin[1] St1	Signal	Bedeutung	Pin St1	Pin D-Sub. 9-pol.	Signal	Bedeutung
1	GND	Ground (PC)	11	1	DPPE	DPPE (optional)
2	+5 V	+5 Volt (PC), optional	12	6	DP5V	+ 5 Volt, isoliert
3	RCV	C165 RS232	13	2	-	n.c.
4	LED1	Leuchtdiode 1	14	7	-	n.c.
5	TMT	C165 RS232	15	3	DPB	DPB, isoliert
6	LED2	Leuchtdiode 2	16	8	DPA	DPA, isoliert
7	-	n.c.	17	4	DPRTS	Request To Send, isol.
8	-	n.c.	18	9	-	n.c.
9	-	n.c.	19	5	DPGND	Ground, isoliert
10	-	n.c.	20	-	-	n.c.

***Tabelle 7-1**: Pinbelegung des Pfostensteckers St1 (Rev. B)*

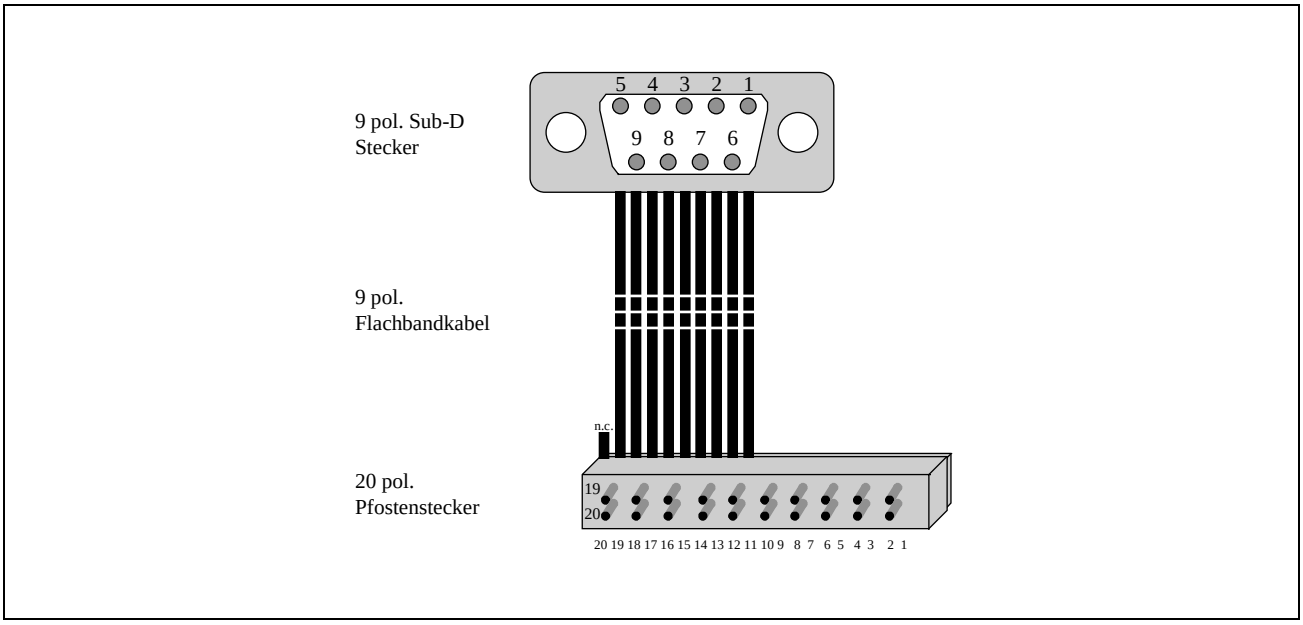


***Abb. 7-1**: Kabel für M-DPM-12 an Siemens ET200 (z.B. SORCUS K2-2720)*

[1] Die Pins 3 und 5 des Steckers dürfen nicht beschaltet werden!

Programmierung

Dieses Kapitel ist nur für jene Anwender gedacht, die direkt auf die Hardware des Moduls zugreifen möchten (Programmierung mit I/O-Adressen).

Auf dem Modul müssen folgende Funktionsgruppen programmiert werden:

- Anwahl einer Interrupt-Leitung vom Modul zur Basiskarte, Einstellung der Art der Konflikterkennung / Interrupt-Auswahl (siehe Interrupt-Select-Register ISR, 8 Bit)
- Timeout-Zähler für die Konfliktbehandlung (siehe Timeout-Register TOR, 16 Bit)
- LED 1 und 2 setzen (siehe LED-Register LER, 2 Bit)
- Das DPRAM als Schnittstelle zum Modul

Nach einem Hardware-Reset des Moduls (z.B. Power-On oder durch das Betriebssystem OsX) sind alle beschreibbaren Register und Pointer des Moduls = 0. Der Timeout-Counter wird auf ffffh gesetzt. **!**

Initialisierung

Modulreset durchführen

Dies kann auf zwei Arten geschehen. Ein Schreibzugriff auf I/O-Adresse MBA + 07h (MBA = Modul-Basis-Adresse) führt einen Reset des Gate-Arrays und einen Reset des PROFIBUS-Masters durch. Die PROFIBUS-Schnittstelle kann auch separat zurückgesetzt werden (XRESET). Die XRESET-Leitung wird über I/O-Adresse MBA + 05h eingestellt. Um einen Reset auszulösen, müssen nacheinander die Datenbytes 8fh, 0fh und 8fh auf die I/O-Adresse MBA + 05h geschrieben werden. Nach einem Reset des Masters geht dieser in den STOP-Zustand. Der Reset des Masters dauert ca. 1 Sekunde. Während dieser Zeit darf nicht auf den Master zugegriffen werden!

Bei einem Reset des Gate-Array werden alle internen Register = 0 gesetzt. Der Timeout-Counter wird auf ffffh gesetzt.

Download eines Binärdatensatzes

Die Parametrierung des PROFIBUS erfolgt über einen Binärdatensatz, der mit Hilfe der Siemens Software COM PROFIBUS erstellt werden kann. Der Binärdatensatz muß dann einmalig mit Hilfe der Bibliothek (M044_LIB) auf das Modul übertragen werden. Anschließend ist ein Reset (s.o.) erforderlich, um die neue Parametrierung zu aktivieren.[1]

LED 1 und 2

Die beiden LEDs auf dem Modul können gemeinsam oder einzeln programmiert werden. Ein Schreibzugriff auf das LED-Register LER setzt den Zustand der LEDs (0 = aus, 1 = ein).

Dabei werden nur die Bits 0 bis 3 verwendet, die Bits 4 bis 7 sind ungültig und sollten zu 0 gesetzt werden.

Bit 0	**LED 1 Anwahl**
0	LED 1 unverändert
1	LED 1 wird entsprechend Bit 2 gesetzt

Bit 1	**LED 2 Anwahl**
0	LED 2 unverändert
1	LED 2 wird entsprechend Bit 3 gesetzt

Bit 2	**LED 1 Ausgabewert**
0	LED 1 aus
1	LED 1 ein

Bit 3	**LED 2 Ausgabewert**
0	LED 2 aus
1	LED 2 ein

Der Status der Leuchtdioden bzw. das LED-Register (LER) kann auch von der Basiskarte zurückgelesen werden (insgesamt 2 Bit, die Bits 2 bis 7 sind ungültig).

---

[1] Siehe Abschnitt 'Inbetriebnahme'.

Anwahl einer Interrupt-Leitung vom Modul zur Basiskarte

Das Modul ist interruptfähig, d.h. es kann bei bestimmten Ereignissen einen Interrupt zur Basiskarte auslösen (pos. Flanke). Die Interrupt-Leitung des Moduls kann per Software mit einem der Interrupt-Eingänge der Basiskarte verbunden werden. Die Anwahl eines Interrupts geschieht durch Setzen des Interrupt-Select-Registers ISR des Moduls Bit 0 bis 2.

Es gibt drei unabhängige Interrupt-Quellen:

1. Durch eine Anforderung des Masters (ASPC 2/C165) über die XINTH-Leitung: Dieser Interrupt kann z.B. dazu verwendet werden, bei einem Systemfehler die Fehlerbehandlung aufzurufen. Aktiviert wird diese Interrupt-Quelle per Software-Reset (siehe Abschnitt 'Hochsprachenbibliotheken').

2. Die Konsistenzsteuerung kann bei einem Konsistenzkonflikt (siehe Abschnitt 'Hochsprachenbibliotheken') einen Interrupt auslösen.

3. Zusätzlich kann der Firmware-Timer über die XTESTO-Leitung ebenfalls zur Interrupt-Auslösung verwendet werden.

Die Bits 4 bis 6 des ISR dienen zur Auswahl der Interrupt-Quelle. Bit 4 selektiert Interrupts vom DPRAM per XINTH. In Bit 5 kann angegeben werden, ob das Konsistenz-Status-Bit einen Interrupt auslösen kann oder ob der interne Timeout-Zähler verwendet wird (kein Interrupt). In Bit 6 kann eingestellt werden, ob der Firmware-Timer Interrupts auslösen kann. Ein gesetztes Bit (=1) aktiviert die jeweilige Interrupt-Quelle.

Während der Einstellung des Interrupt-Kanals darf das Modul keinen Interrupt anfordern bzw. auf der Basiskarte müssen die Interrupts (vorübergehend) maskiert werden. Bit 3 und Bit 7 des ISR sind reserviert und sollten auf 0 gesetzt werden.

Anwahl des Interrupts zur Basiskarte:

Bit 2	Bit 1	Bit 0	Interrupt-Leitung der MODULAR-4/486
0	0	0	keine
0	0	1	IRQ-A
0	1	0	IRQ-B
0	1	1	IRQ-C
1	0	0	IRQ-D
1	0	1	IRQ-E
1	0	1	IRQ-F
1	1	1	NMI

Mögliche Interrupt-Quelle:

Bit 4	Master-Interrupt Enable (XINTH)[1]
0	kein Interrupt
1	Interrupt bei XINTH = 0

Bit 5	Konsistenzkonflikt-Steuerung/-Erkennung
0	Konsistenzanforderung durch ASPC 2 löst Konsistenz-Timer aus
1	Interrupt bei Konsistenzanforderung durch ASPC 2

Bit 6	Firmware-Timer Interrupt Enable
0	kein Interrupt
1	periodischer Interrupt per Firmware-Timer

Das Interrupt-Select-Register (ISR) kann auch von der Basiskarte gelesen werden. Dabei sind die Bits 3 und 7 reserviert bzw. ungültig.

[1] Siehe auch Abschnitt Programmierung.

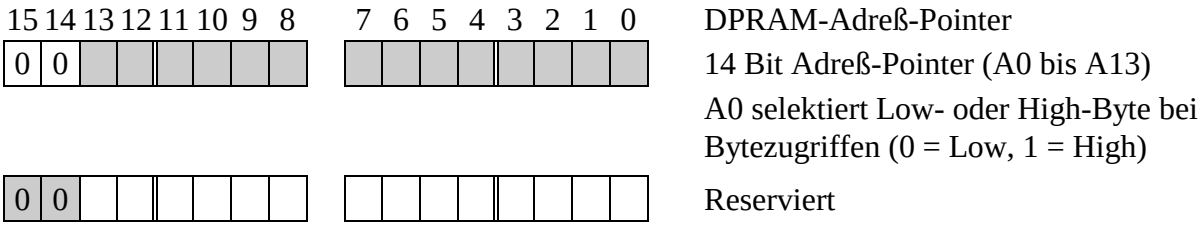
Löschen einer Interrupt-Anforderung:

Eine Interrupt-Anforderung kann vom Anwender im Interrupt-Status-Register (IST) gelesen werden. Die Master-Interrupt-Anforderung bleibt solange aufrecht erhalten (XINTH = 0), bis der Anwender einen Interrupt-Acknowledge ausführt (durch einen Lesezugriff auf die DPRAM-Adresse 1ffeh). Das Konsistenz-Status-Bit wird durch Setzen oder Löschen einer Konsistenzanforderung gelöscht. Ein Firmware-Timer Interrupt muß durch einen Schreibzugriff auf das Interrupt-Select-Register (ISR) gelöscht werden.

Zugriffe auf das Dual-Port-RAM (DPRAM)

Zugriffe auf das DPRAM auf dem Modul erfolgen über einen programmierbaren Adreß-Pointer (14 Bit). Um das Dual-Ported-RAM sowohl byte- als auch wortweise zugreifen zu können, selektiert Bit 0 des Adreß-Pointers bei Bytezugriffen, ob das Low- oder das High-Byte verwendet wird. Unabhängig davon wird bei einem DPRAM-Zugriff über die I/O-Adresse festgelegt, ob wort- oder byteweise gelesen bzw. geschrieben wird. Bei Wortzugriffen wird unabhängig von Bit 0 auf das gerade selektierte Wort zugegriffen.

Da die I/O-Zugriffe der Basiskarte auf das DPRAM immer wortweise erfolgen, muß die Anwendersoftware die Daten bei High-Byte-Zugriffen bei Bedarf entsprechend swappen.



Bei Zugriffen auf das DPRAM kann über die I/O-Adresse ausgewählt werden, ob der Adreß-Pointer nach Beendigung des Zugriffs automatisch inkrementiert werden soll. Bei Wortzugriffen wird der Adreß-Pointer dann um 2 inkrementiert, bei Bytezugriffen um 1. Der gültige Adreßbereich liegt von 0000h bis 3fffh.

Adressierung des Dual-Ported RAM:

A0 = 0, Low-Byte	A0 = 1, High-Byte
3ffeh	3fffh
.	.
.	.
0000	0001

Um auf das DPRAM zugreifen zu können, sind folgende Zugriffe notwendig:

Zuerst muß der DPRAM-Adreß-Pointer, per I/O-Schreibzugriff (MBA + 00h) auf die zu lesende bzw. schreibende Adresse, gesetzt werden. Nachfolgende Lese- oder Schreibzugriffe auf das DPRAM greifen dann über diesen Adreß-Pointer zu: 8- oder 16-Bit Lese bzw. Schreibzugriff durchführen, bei Bedarf mit Autoinkrement des Adreß-Pointers um 1 bzw. 2.

Konsistente Zugriffe

Konsistente Zugriffe auf das DPRAM werden durch Setzen der Read- bzw. Write-Konsistenzleitung eingeleitet (Konsistenzanforderung). Danach müssen sofort die Zugriffe auf das DPRAM erfolgen. Sind alle Zugriffe ausgeführt worden, muß die Konsistenzanforderung zurückgenommen werden. Das geschieht durch einen Lesezugriff auf das Modul-Statusregister. Wenn sowohl die Basiskarte als auch das Modul gleichzeitig konsistent auf das DPRAM zugreifen wollen, wird eine Konfliktsteuerung aktiviert. Dabei hat der ASPC 2 Vorrang vor dem Host, d.h. nach einer bestimmten Zeit (Timeout) muß der Host seine Konsistenzanforderung zurücknehmen.

Die Konfliktsteuerung des Moduls regelt den Ablauf der Zugriffe so, daß der Host (also die Basiskarte) entweder nach Ablauf eines internen Timeout-Zählers keine gültigen Zugriffe durchführen kann oder ein Interrupt ausgelöst wird, sobald der ASPC 2 konsistent zugreifen will. In diesem Fall hat der Host noch eine bestimmte Zeit (s.u., abhängig von der PROFIBUS-Baudrate) zur Verfügung, um seine Zugriffe zu beenden.

Konfliktsteuerung

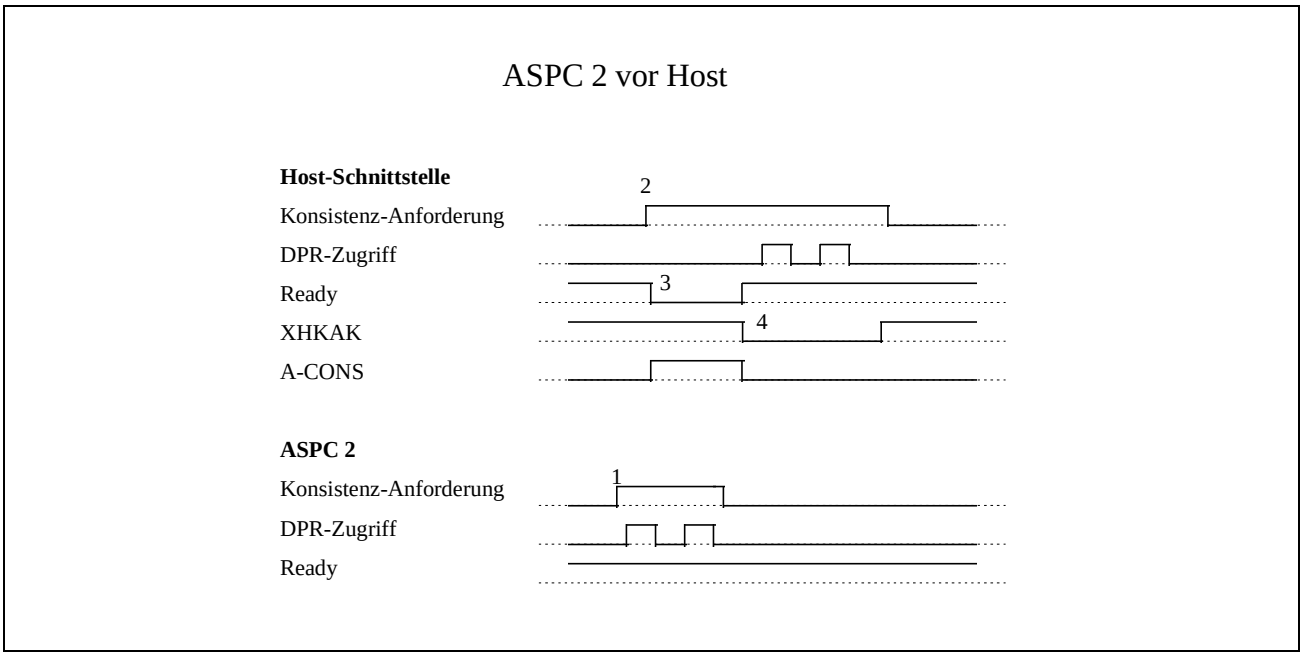
Vorab ein kurze Erklärung zu den verwendeten Steuerleitungen:

XRHCONS	Konsistenz-Leseanforderung (low-aktiv) von der Basiskarte
XWHCONS	Konsistenz-Schreibanforderung (low-aktiv) von der Basiskarte
XHKAK	Konsistenz-Acknowledge (Bestätigung, low-aktiv) vom ASPC 2, konsistente Zugriffe von der Basiskarte freigegeben
A-CONS	Konsistenzanforderung (high-aktiv) vom ASPC 2, die Basiskarte kann nur dann zugreifen, wenn A-CONS = 0 und XHKAK = 0
RDYH	Ready-Leitung zur Basiskarte (high-aktiv) Zugriffe von der Basiskarte können im Konfliktfall verzögert werden
RDYP	Ready-Leitung zum ASPC 2 (high-aktiv) Zugriffe vom ASPC 2 können im Konfliktfall verzögert werden

Basiskarte greift nach ASPC 2 zu

Wenn der ASPC 2 bereits konsistent zugreift, und die Basiskarte ebenfalls zugreifen will, wird der Basiskartenzugriff automatisch gebremst (*RDYH-Entzug*). Sobald der ASPC 2 seine Zugriffe beendet hat, wird der Basiskartenzugriff freigegeben und die Zugriffe können erfolgen.

Der Anwender kann auch, nachdem er die Konsistenzanforderung gesetzt hat, die A-CONS-Leitung abfragen. Sobald A-CONS = 0 ist (d.h. der ASPC 2 greift nicht mehr konsistent zu), können die Zugriffe auf das DPRAM erfolgen.



Ablauf:

1. Der ASPC 2 fordert konsistente Zugriffe an.
2. Der Host fordert konsistente Zugriffe an (etwas später).
3. Der Host greift ohne auf XHKAK zu warten zu, und wird über Ready-Entzug gebremst.
4. Der ASPC 2 beendet seinen Zugriff, damit kann der Host beginnen.

ASPC 2 greift nach Basiskarte zu

Solange gültige Basiskartenzugriffe erfolgen, wird die *XHKAK*-Leitung (Konsistenz-Acknowledge) aktiviert.

Wenn während eines konsistenten Zugriffs (Konsistenzanforderung der MODULAR-4/486 Karte aktiv) der ASPC 2 ebenfalls konsistent zugreifen möchte, so wird dieser automatisch gebremst (*RDYP-Entzug*). Allerdings muß die Konsistenzanforderung der Basiskarte spätestens 80µs (bei 12 MBaud) nach der Anforderung des ASPC 2 beendet sein, da dieser sonst nicht die Zykluszeit einhalten kann.

Konfliktsteuerung mit Timeout-Zähler

Das Gate-Array startet einen Timer, sobald der ASPC 2 konsistent zugreifen möchte (*ACONS* wird aktiviert und *XHKAK* ist aktiv). Überschreitet der Timer die eingestellte Timeout-Zeit, wird der Zugriff der Basiskarte abgebrochen (XCSHOST = 1, Konsistenzanforderung beendet). Damit kann der ASPC 2 seine Zugriffe beginnen.

Der Anwender muß nach seinem letzten konsistenten Zugriff die Konsistenzanforderung beenden. Das geschieht durch einen Lesezugriff, der zusätzlich den Status der Leitungen TIMEOUT-STATUS, R-CONS, W-CONS, A-CONS, XHKAK, XCSDPR2 und das KONSISTENZ-STATUS-BIT zurückliefert. Ist TIMEOUT-STATUS = 1, so ist der Timeout-Zähler abgelaufen. Somit war der Zugriff bzw. die Daten durch den Host ungültig. Der gesamte Zugriff muß dann wiederholt werden. Zuvor muß zusätzlich der Status gelöscht werden.

$$t_{TIMEOUT} = \frac{8 \cdot AnzByte_{FIFO}}{Baudrate_{PROFIBUS}}$$

Der Timeout-Zähler läßt sich von 0 bis 0,65 s einstellen. Dazu dient das Timeout-Counter Register TOR (16 Bit). Die Timeout-Zeit ist abhängig von der verwendeten PROFIBUS-Baudrate und der FIFO-Größe des ASPC 2.

Bei einem ASPC 2 Step C (128 Byte FIFO) ergeben sich folgende max. Timeout-Zeiten:

Mögliche Timeout-Werte sind z.B.: 84 µs bei 12 MBaud

68 ms bei 1,5 Mbaud

bis

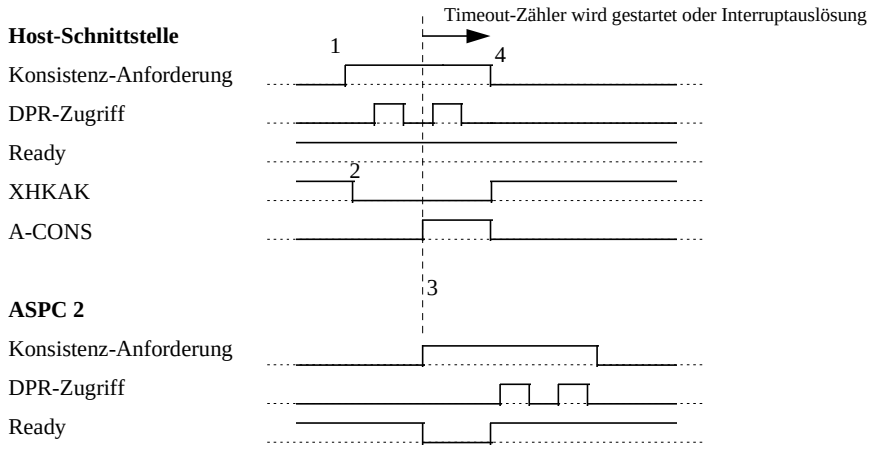
106 ms bei 9600 Baud

Bei ASPC 2 Step B (64 Byte FIFO) halbieren sich die Timeout-Zeiten!

Durch einen Schreibzugriff auf I/O-Adresse MBA + 08h kann der 16 Bit Timer-Wert (TOR) gesetzt werden.

Konfliktsteuerung mit Interrupt

Wenn während eines konsistenten Zugriffs durch den Host eine Konsistenzanforderung durch den ASPC 2 erfolgt (A-CONS = 1, s.o.), kann ein Interrupt zur Basiskarte ausgelöst (KONSISTENZ-STATUS-BIT = 1) werden. Der Host muß dann dafür sorgen, daß die Zugriffe rechtzeitig beendet werden. Der Timeout-Zähler wird dabei nicht benutzt. Wird die Konsistenzanforderung nicht rechtzeitig beendet bzw. wenn der Zugriff die Timeout-Zeit überschreitet, so kann das zu einem Fehlerzustand des PROFIBUS führen.



Ablauf:

1. Der Host fordert konsistente Zugriffe an.
2. XHKAK gibt die Zugriffe frei, da ASPC 2 keine Konsistenz anfordert.
3. ASPC 2 fordert Konsistenz, wird aber durch Ready-Entzug gebremst. Der Timeout-Zähler wird gestartet oder ein Interrupt zur Basiskarte ausgelöst. Die Host-Konsistenz muß vor Ablauf des Timers bzw. der Timeout-Zeit beendet sein.
4. Host-Konsistenz beendet, ASPC 2 beginnt zuzugreifen.

Allgemeiner Ablauf

Sollen konsistente Zugriffe auf das DPRAM durchgeführt werden, sieht der Ablauf folgendermaßen aus:

1. **Setzen der Lese- oder Schreib-Konsistenzanforderung** (MBA + 18h, Data = 01h für Lesen, Data = 02h für Schreiben).
2. **Modulstatus lesen** (MBA + 19h) bis XHKAK = 0 (Bit 5, Konsistenz-Acknowledge).
3. **Setzen des DPRAM-Pointers** auf die zu lesende bzw. schreibende Adresse.
4. **8- oder 16-Bit Lese bzw. Schreibzugriff durchführen**, bei Bedarf mit Autoinkrement des Adreß-Pointers.
5. **Wiederholen der Punkte 3 (optional) und 4** bis alle Daten übertragen sind.
6. **Löschen der Konsistenzanforderung** durch Lesen von MBA + 18h. Dabei wird automatisch der Modulstatus ausgegeben. Ist KONSISTENZ-STATUS = 1, ist während des konsistenten Host-Zugriffs eine konsistente Anforderung des ASPC 2 aufgetreten. Nur wenn auch der TIMEOUT-STATUS = 1 ist, kam es zu einem Fehler. Dann muß der gesamte Zugriff vom Host wiederholt werden, alle Daten sind ungültig. Zusätzlich muß der Status durch einen Schreibzugriff auf MBA + 18h mit Data = 0 gelöscht werden, damit Zugriffe auf das DPRAM wieder möglich sind.

Hochsprachenbibliotheken

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M044_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Prozedur **m044_bib_startup** einmal aufgerufen werden.

Verwendung der Bibliothek mit der Treibertask M044TASK

Mit den Bibliotheken M044_LIB wird zusätzlich eine NI-Task M044TASK.EXE und eine zugehörige Task-Bibliothek M044_MDD mitgeliefert. Die Bibliothek M044_MDD hat die gleichen Funktionen und Prozeduren wie die M044_LIB, jedoch werden alle PROFIBUS-Funktionen per Taskfunktion ausgeführt. Das hat den Vorteil, daß in einer Multitasking-Umgebung mehrere Anwendungen sich ein Modul teilen können. Evtl. auftretende Konflikte werden durch die Task abgefangen. Die M044_MDD-Bibliothek ist sowohl für PC- als auch für Echtzeitprogramme geeignet. Zum Einbinden der Bibliothek können die Header-Dateien der M044_LIB verwendet werden.

Die Task M044TASK.EXE hat die Programmnummer 329h und muß mit den Flags 0180h auf der MODULAR-4/486 installiert werden. Eine globale Variable *tasknum* vom Typ *ushort* muß vom Anwendungsprogramm *extern* deklariert und mit der Tasknummer voreingestellt sein. Dies kann auch mit m044_set_tasknum (tasknummer) vor Aufruf der ersten Bibliotheksfunktion geschehen.

Fehlerbehandlung

Alle Bibliotheksfunktionen liefern als Rückgabewert einen Fehlerstatus zurück, der vom Anwendungsprogramm ausgewertet werden muß.

Fehler 1bh Kein M-DPM-12 auf angegebenem Modulsteckplatz bzw. der in der Funktion angegebene Steckplatz stimmt nicht mit dem per m044_configure eingestellten Slot überein.

Fehler 40h Fehler bei der Ausführung der Funktion, die Fehlerdiagnose muß aus Fehlerpuffer mit 'm044_get_error' gelesen werden.

Fehler 41h Funktion derzeit nicht ausführbar, da durch andere Anwendung belegt.

Nach Aufruf einer Bibliotheksfunktion muß im Fehlerfall die Diagnosemeldung gelesen und entsprechend ausgewertet werden. Dazu dient die Funktion (1)

m044_get_error — Lies Fehlerdiagnose

Pascal FUNCTION m044_get_error (micro_slot: byte, VAR data) : word

C ushort EXPORT m044_get_error (byte micro_slot, ushort *data);

Funktion Die Fehlerdiagnose wird gelesen.

Parameter *data*: Das 1. Wort enthält die Funktionsnummer (x = 1..33) bei der der Fehler aufgetreten ist, ab Wort 2 stehen die Diagnosedaten (derzeit immer ein Wort). Insgesamt werden 10 Wörter gelesen.

Hinweis Liefert eine Bibliotheksfunktion in der Diagnose einen Timeout-Fehler, wurde ein Kommando an den Master nicht quittiert. Dies kann unter anderem daran liegen, daß das C-Link nicht aufgesteckt ist. !

Konfiguration

m044_bib_startup — Initialisiere Modulbibliothek

Pascal FUNCTION m044_bib_startup : word;

C ushort EXPORT m044_bib_startup (void);

Funktion Diese Funktion (2) initialisiert die Modulbibliothek. Es werden u.a. die Initialisierungsdaten aus den EEPROMs aller aufgesteckten Module M-DPM-12 in die Bibliothek übernommen.

m044_configure — Konfiguriere Modul

Pascal FUNCTION m044_configure (micro_slot: byte) : word;

C ushort EXPORT m044_configure (byte micro_slot);

Funktion Diese Funktion (3) konfiguriert ein Modul nach den Vorgaben im EEPROM des Moduls.

m044_get_master_mode — Lies aktuellen Master-Modus

Pascal FUNCTION m044_get_master_mode (micro_slot; VAR status: byte): word;

C ushort EXPORT m044_get_master_mode (byte micro_slot,
byte *status);

Funktion Diese Funktion (4) liest den aktuellen Mastermodus zur Koordinierung beim Hochlaufen des Masters. Ist der Mastermodus nach einem Hardware-Reset (z.B. Power-On) _M044_STOP, so ist das Modul betriebsbereit.

Parameter	*status*:	_M044_STOP:	Master ist im STOP-Modus
		_M044_CLEAR:	Master ist im CLEAR-Modus
		_M044_OPERATE:	Master ist im OPERATE-Modus

Parametrierung des PROFIBUS

Binärdaten zur PROFIBUS-Parametrierung

Das Modul benötigt zur PROFIBUS-Parametrierung einen sogenannten Binärdatensatz. Dieser Binärdatensatz kann mit dem PC-Programm COM PROFIBUS (Siemens) unter Windows erstellt und in einer Binärdatei gespeichert werden. Diese Binärdatei muß auf das Modul übertragen (Download) werden.

Die Übertragung der Binärdaten ist sowohl vom Host zum Modul (Download) als auch vom Modul zum Host (Upload) möglich. Der Mastermodus muß 'STOP' sein.

m044_download — Binärdaten übertragen

Pascal: FUNCTION m044_download (micro_slot: byte; VAR data; data_size: byte; flag: byte) : word;

C: ushort EXPORT m044_download (byte micro_slot, void *data, byte data_size, byte flag)

Funktion: Diese Funktion (5) überträgt Binärdaten vom Host zum Modul. Der Download einer Binärdatei (Größe: etwa 7 KByte) auf das Modul muß in mehreren Schritten erfolgen, da nur Datenblöcke von maximal 200 Byte am Stück übertragen werden können. Zwischen dem Senden der Blöcke dürfen keine weiteren Bibliotheksfunktionen aufgerufen werden.

Parameter:
- *data*: Zeiger auf den zu übertragenden Datenblock
- *data_size*: Größe des Datenblockes (0 bis 200) in Byte
- *flag*: _M044_NO_MORE_DATA: keine weiteren Binärdaten; _M044_MORE_DATA: es folgen weitere Binärdaten

Diagnose:
- *0*: Befehl korrekt ausgeführt
- *2*: Fehler: Timeout, Abbruch der Funktion
- *256*: Fehler: Befehlssemaphore nicht gesetzt

Hinweis: Sind alle Datenblöcke übertragen, muß ein Reset des Moduls (mit der Funktion **m044_reset_master(..)**) erfolgen. Anschließend muß der Master im Betriebszustand **'STOP'** sein. Falls dies nicht der Fall ist, war die Übertragung der Binärdatei oder die Binärdatei selbst fehlerhaft. !

Von einem PC-Programm aus können die als Binärdatendatei vorliegenden Binärdaten mit der folgenden Funktion (6) vom Host zum Modul übertragen werden. Anschließend muß ein Reset des Moduls erfolgen.

m044_download_file — Binärdatei übertragen

Pascal
FUNCTION m044_download_file (micro_slot: byte;
binary_file : str80; VAR downloaded : word) : word;

C
ushort EXPORT m044_download_file (byte micro_slot,
const void *binary_file, ushort *downloaded)

Funktion
Übertragen einer Binärdatei zum Modul.

Parameter
binary_file: Name der Binärdatei

downloaded: Anzahl übertragener Byte

Diagnose
0: Befehl korrekt ausgeführt

1: Fehler beim Lesen der Datei

256: Fehler: Befehlssemaphore nicht gesetzt

m044_upload — Binärdaten lesen

Pascal
FUNCTION m044_upload (micro_slot: byte; VAR data;
VAR data_size: byte; VAR flag: byte) : word;

C
ushort EXPORT m044_upload (byte micro_slot, void *data,
byte *data_size, byte *flag)

Funktion
Diese Funktion (7) überträgt Binärdaten vom Modul zum Host. Eine Binärdatei ist in der Regel etwa 7 KByte groß. Da nur 250 Bytes übertragen werden können, ist es erforderlich, diese Funktion mehrfach aufzurufen. Vor dem 1. Aufruf muß im Parameter *flag* _M044_START_DATA eingetragen werden.

Parameter	*data*:	Zeiger auf Zieldatenbereich
	data_size:	Anzahl der übertragenen Binärdaten (in Byte)
	flag:	Übertragungsparameter: _M044_START_DATA: 1. zu übertragender Block _M044_NO_MORE_DATA: alle Binärdaten übertragen _M044_MORE_DATA: Übertragung unvollständig
Diagnose	*0*:	Befehl korrekt ausgeführt
	2:	Fehler: Timeout, Abbruch der Funktion
	5:	Fehler: falsche Befehlsreihenfolge
	256:	Fehler: Befehlssemaphore nicht gesetzt

Von einem PC-Programm aus können die vorliegenden Binärdaten mit der folgenden Funktion (8) vom Modul zum Host übertragen und in einer Datei gespeichert werden:

m044_upload_file Binärdatei lesen und abspeichern

Pascal FUNCTION m044_upload_file (micro_slot: byte; binary_file : str80; VAR uploaded : word) : word;

C ushort EXPORT m044_upload_file (byte micro_slot, const void *binary_file, ushort *uploaded)

Parameter	*binary_file*:	Name der Binärdatei
	uploaded:	Anzahl übertragener Byte
Diagnose	*0*:	Befehl korrekt ausgeführt
	1:	Fehler beim Schreiben der Datei
	5:	Fehler: falsche Befehlsreihenfolge
	256:	Fehler: Befehlssemaphore nicht gesetzt

m044_set_slave_address — Setze Slave-Adresse

Pascal: FUNCTION m044_set_slave_address (micro_slot: byte; VAR data; size: byte) : word;

C: ushort EXPORT m044_set_slave_address (byte micro_slot, void *data, byte size);

Funktion: Diese Funktion (9) setzt die Adresse eines Slaves. Bedingung ist, daß sich der Slave über den Bus programmieren läßt. Die Funktion ist für jeden Slave erforderlich, dessen Adresse konstruktionsbedingt größer als 123 ist. Neu zu adressierende Slaves müssen bereits bei der Projektierung mit COM PROFIBUS mit korrekten Adressen eingetragen sein.

Parameter:

data: Zeiger auf Slaveparameterstruktur
1. Byte: neue Slaveadresse
2. Byte: Adreßänderungskennung
3. Byte: alte bzw. aktuelle Slaveadresse
Optional können weitere Slavedaten nach Norm übertragen werden (siehe Handbuch zum Slave).

size: Größe der Slaveparameterstruktur (3 bis 250 Byte, s.o.)

Diagnose:

0: Befehl korrekt ausgeführt

1: Fehler: Slave kann Dienst nicht bearbeiten

2: Fehler: Slave hat keine Ressourcen

3: Fehler: Befehl bei Slave nicht aktiviert

5: Fehler: Buskurzschluß

6: Fehler: Timeout, Abbruch der Funktion

159: Fehler: Slave antwortet nicht

175: Fehler: Slave antwortet falsch

191: Fehler: Busstörung

256: Fehler: Befehlssemaphore nicht gesetzt

m044_restart Master-Software zurücksetzen

Pascal FUNCTION m044_restart (micro_slot: byte; restart_par: byte) : word;

C ushort EXPORT m044_restart (byte micro_slot, byte restart_par);

Funktion Diese Funktion (10) setzt die Mastersoftware auf dem Modul zurück. Das Rücksetzen dieser Software ist nur nach einem Download oder Setzen einer Slaveadresse erforderlich, um die neue PROFIBUS-Konfiguration zu aktivieren.

Parameter *restart_par*: Nach dem Restart einzunehmender Betriebsmodus:

_M044_STOP:	Master in STOP-Modus
_M044_CLEAR:	Master in CLEAR-Modus
_M044_OPERATE:	Master in OPERATE-Modus

Hiermit werden die momentan gesetzten Busparameter (z.B. die nach einem Download) übernommen. Sollen andere Busparameter übernommen werden, so kann einer der folgenden Parameter mit dem Betriebsmodusparameter verknüpft (=bitweises OR) werden:

_M044_DEFAULT_PAR:	Defaultparameter
_M044_EPROM_PAR:	Parameter aus EPROM

Soll beim Auftreten eines Profibussystemfehlers auf dem Modul ein Interrupt zur Basiskarte ausgelöst werden, so ist eine weitere OR-Verknüpfung mit folgendem Parameter erforderlich:

_M044_ENABLE_IRQ: Systemfehler-Interrupt

Diagnose

0:	Befehl korrekt ausgeführt
1:	Fehler: Timeout, Abbruch der Funktion
256:	Fehler: Befehlssemaphore nicht gesetzt

m044_refresh_slave_datastruct — Aktualisiere Datenstrukturpuffer

Pascal FUNCTION m044_refresh_slave_datastruct (micro_slot: byte):word;

C ushort EXPORT m044_refresh_slave_datastruct (byte micro_slot);

Funktion Diese Funktion (11) überträgt die Datenstrukturinformationen aller parametrierten Slaves in den Datenstrukturpuffer der Modulbibliothek. Der Aufruf dieser Funktion ist nach einem Restart erforderlich.

Diagnose
- *0*: Befehl korrekt ausgeführt
- *1*: Fehler: Timeout, Abbruch der Funktion
- *5*: Fehler: falsche Befehlsreihenfolge
- *256*: Fehler: Befehlssemaphore nicht gesetzt

Master-Steuerung

m044_set_master_mode — Setze Master-Betriebsmodus

Pascal FUNCTION m044_set_master_mode (micro_slot: byte; command: byte) : word;

C ushort EXPORT m044_set_master_mode (byte micro_slot, byte command);

Funktion Diese Funktion (12) steuert den Betriebsmodus des Masters.

Parameter *command*:
- _M044_CLEAR: Master in CLEAR-Modus
- _M044_STOP: Master in STOP-Modus
- _M044_OPERATE: Master in OPERATE-Modus
- _M044_SYNCHRONIZE: Synchronisierungsbefehl zwischen Host und Master

Diagnose
- *0*: Befehl korrekt ausgeführt
- *1*: Fehler: Timeout, Abbruch der Funktion
- *256*: Fehler: Befehlssemaphore nicht gesetzt

m044_watchdog_enable Watchdog des Masters aktivieren

Pascal FUNCTION m044_watchdog_enable (micro_slot: byte; timeout: word) : word;

C ushort EXPORT m044_watchdog_enable (byte micro_slot, ushort timeout);

Funktion Diese Funktion (13) aktiviert bzw deaktiviert den Watchdog des Masters. Ist der Watchdog aktiviert, muß die Anwendersoftware dafür sorgen, daß der Watchdog im eingestellten Zeitintervall zurückgesetzt wird.

Parameter *timeout*: Timeout als Faktor von 10 ms (0 bis 65535)
timeout = 0 deaktiviert den Watchdog

Diagnose *0*: Befehl korrekt ausgeführt

1: Fehler: Timeout, Abbruch der Funktion

256: Fehler: Befehlssemaphore nicht gesetzt

m044_watchdog_retrigger Watchdog triggern

Pascal FUNCTION m044_watchdog_retrigger (micro_slot: byte) : word;

C ushort EXPORT m044_watchdog_retrigger (byte micro_slot);

Funktion Diese Funktion (14) retriggert den Watchdog des Moduls und muß vom Anwenderprogramm zyklisch aufgerufen werden. Falls der Watchdog abgelaufen ist, bleibt der Master stehen und greift nicht mehr auf die Slaves zu. Die Slaves können daraufhin (bei eingeschalteter Ansprechüberwachung) in einen definierten Zustand gehen. Um den Master zu reaktivieren, muß ein Reset des Moduls bzw. des Masters erfolgen!

Diagnose *0*: Befehl korrekt ausgeführt

>0: Fehler: Timeout beim Retriggern

Reset des Moduls

m044_reset_master — Master zurücksetzen

Pascal FUNCTION m044_reset_master (micro_slot: byte) : word;

C ushort EXPORT m044_reset_master (byte micro_slot);

Funktion Diese Funktion (15) führt einen Reset des Masters durch. Die Registerinhalte des Moduls bleiben unverändert. Nach einem Reset des Masters befindet sich dieser im Stop-Zustand. Der Reset des Masters dauert mehrere Millisekunden. Während dieser Zeit darf nicht auf den Master zugegriffen werden!

m044_hard_reset — Modul zurücksetzen

Pascal FUNCTION m044_hard_reset (micro_slot: byte) : word;

C ushort EXPORT m044_hard_reset (byte micro_slot);

Funktion Diese Funktion (16) setzt das gesamte Modul (Gate-Array und PROFIBUS-Master) zurück. Nach einem Reset des Masters befindet sich dieser im Stop-Zustand. Der Reset des Masters dauert mehrere Millisekunden. Während dieser Zeit darf nicht auf den Master zugegriffen werden!

Slave-Zugriff

Definition des Datenkanals

Die Verbindung zwischen dem PROFIBUS-Master und den PROFIBUS-Slaves wird im folgenden als Dateneingangskanal bzw. Datenausgangskanal bezeichnet. Damit auf die Ein- und Ausgänge eines Slaves gezielt zugegriffen werden kann, werden die Datenkanäle in Subkanäle unterteilt. Diese Subkanäle werden bei 0 beginnend nach folgendem Schema (Beispiel für einen 16-Byte-Datenkanal) hochgezählt:

relativer Offset	**Byte-Zugriffe**	**Word-Zugriffe**
0	Subkanal 0	Subkanal 0
1	Subkanal 1	
2	Subkanal 2	Subkanal 1
3	Subkanal 3	
:	:	:
14	Subkanal 14	Subkanal 7
15	Subkanal 15	

m044_set_slave_byte
m044_set_slave_word — Schreibe Byte oder Word

Pascal: FUNCTION m044_set_slave_byte (micro_slot: byte; slave: byte; subchannel: byte; data: byte; cons: byte) : word;

Pascal: FUNCTION m044_set_slave_word (micro_slot: byte; slave: byte; subchannel: byte; data: word; cons: byte) : word;

C: ushort EXPORT m044_set_slave_byte (byte micro_slot, byte slave, byte subchannel, byte data, byte cons);

C: ushort EXPORT m044_set_slave_word (byte micro_slot, byte slave, byte subchannel, ushort data, byte cons);

Funktion: Diese Funktionen (17 und 18) schreiben ein Einzeldatum (Byte bzw. Wort) in einen Ausgangs-Subkanal eines Slaves.

Parameter:

slave: Slaveadresse (3 bis 123)

subchannel: Subkanal

data: zu setzendes Datum vom Typ Byte oder Word

cons: Konsistenzflag:
_M044_NO_CONS: ohne Konsistenz
_M044_CONS: mit Schreib-Konsistenz

Diagnose:

0: Befehl korrekt ausgeführt

100: Fehler: Bus anders parametriert

101: Fehler: Konsistenzkonflikt (Schreiben muß wiederholt werden)

m044_set_slave_block Schreibe Datenblock

Pascal FUNCTION m044_set_slave_block (micro_slot: byte; slave: byte; word : offset; size: byte; VAR data; cons:byte) : word;

C ushort EXPORT m044_set_slave_block (byte micro_slot, byte slave, ushort offset, byte size, void *data, byte cons);

Funktion Diese Funktion (19) schreibt einen Datenblock beginnend bei Subkanal 0+ Offset in den Dateneingangskanal eines Slaves.

Parameter

slave: Slaveadresse (3 bis 123)

offset: Offset auf Subkanal 0

size: Anzahl zu setzender Byte

data: Zeiger auf Quelldatenpuffer (zu setzender Block)

cons: Konsistenzflag:
_M044_NO_CONS: ohne Konsistenz
_M044_CONS: mit Schreib-Konsistenz

Diagnose

0: Befehl korrekt ausgeführt

100: Fehler: Bus anders parametriert

101: Fehler: Konsistenzkonflikt (Schreiben muß wiederholt werden)

102: Fehler: Parameter *size* ist größer als der Datenpuffer des Slave

m044_get_slave_byte
m044_get_slave_word — Lies Byte oder Wort

Pascal: FUNCTION m044_get_slave_byte (micro_slot: byte; slave: byte; subchannel: byte; VAR data: byte; cons: byte) : word;

Pascal: FUNCTION m044_get_slave_word (micro_slot: byte; slave: byte; subchannel: byte; VAR data: word; cons: byte) : word;

C: ushort EXPORT m044_get_slave_byte (byte micro_slot, byte slave, byte subchannel, byte *data, byte cons);

C: ushort EXPORT m044_get_slave_word (byte micro_slot, byte slave, byte subchannel, ushort *data, byte cons);

Funktion: Diese Funktionen (20 und 21) lesen ein Einzeldatum (Byte bzw. Wort) von einem Eingangs-Subkanal eines Slaves.

Parameter:

slave: Slaveadresse (3 bis 123)

subchannel: Subkanal

data: Zeiger auf Zieldatenpuffer (Byte oder Word)

cons: Konsistenzflag:
_M044_NO_CONS: ohne Konsistenz
_M044_CONS: mit Lese-Konsistenz

Diagnose:

0: Befehl korrekt ausgeführt

100: Fehler: Bus anders parametriert

101: Fehler: Konsistenzkonflikt (Lesen muß wiederholt werden)

m044_get_slave_block — Lies Datenblock

Pascal
FUNCTION m044_get_slave_block (micro_slot: byte; slave: byte; offset : word, size: byte; VAR data; cons: byte) : word;

C
ushort EXPORT m044_get_slave_block (byte micro_slot, byte slave,

ushort offset, byte size, void *data, byte cons);

Funktion
Diese Funktion (22) liest einen Datenblock beginnend bei Subkanal 0 + Offset von dem Dateneingangskanal eines Slaves.

Parameter

slave: Slaveadresse (3 bis 123)

offset: Offset auf Subkanal 0

size: Anzahl zu lesender Byte

data: Zeiger auf Zieldatenpuffer

cons: Konsistenzflag:
_M044_NO_CONS: ohne Konsistenz
_M044_CONS: mit Schreib-Konsistenz

Diagnose

0: Befehl korrekt ausgeführt

100: Fehler: Bus anders parametriert

101: Fehler: Konsistenzkonflikt (Lesen muß wiederholt werden)

102: Fehler: Parameter *size* ist größer als der Datenpuffer des Slave

m044_set_slave_command — Erteile Auftrag an Slave/Slave-Gruppe

Pascal: FUNCTION m044_set_slave_command (micro_slot: byte; order: byte; group: byte; slave: byte) : word;

C: ushort EXPORT m044_set_slave_command (byte micro_slot, byte order, byte group, byte slave);

Funktion: Diese Funktion (23) erteilt einem Slave bzw. einer Slavegruppe einen Auftrag. Gruppen können mit Hilfe von COM PROFIBUS definiert werden.

Parameter:

order:

_M044_FREEZE:	Eingänge einfrieren
_M044_UNFREEZE:	Einfrieren beenden
_M044_SYNC:	Ausgänge 'synchron' setzen
_M044_UNSYNC	Ausgänge 'asynchron' setzen

group: Gruppennummer (1..255, bei Gruppe 0: 255 eintragen)

slave: Slaveadresse (3..123) oder _M044_BROADCAST

Diagnose:

0: Befehl korrekt ausgeführt

1: Fehler: Timeout, Abbruch der Funktion

2: Fehler: Kommando nicht erlaubt

3: Fehler: Kommando an Slave oder Slavegruppe nicht erlaubt

256: Fehler: Befehlssemaphore nicht gesetzt

m044_get_slave_datastruct Lies Datenstruktur eines Slave

Pascal FUNCTION m044_get_slave_datastruct (micro_slot: byte; slave: byte; VAR data: m044_slv_datastr_type) : word;

C ushort EXPORT m044_get_slave_datastruct (byte micro_slot, byte slave, m044_slv_datastr_type *data);

Funktion Diese Funktion (24) liest eine Datenstruktur aus dem Datenstrukturpuffer der Bibliothek. Es wird vorausgesetzt, daß der Datenstrukturpuffer die aktuellen Datenstrukturen aller parametrierten Slaves enthält (siehe m044_refresh_slave_datastruct).

Parameter *slave*: Slaveadresse (3..123)

data: Adresse einer Datenstrukturvariablen vom Typ m044_slv_datastr_type (siehe folgende Tabelle)

Diagnose *0*: Befehl korrekt ausgeführt

1: Fehler: Slavenummer ungültig

Der Variablentyp m044_slv_datastr_type ist eine Datenstruktur, bestehend aus folgenden Feldern:

Feld	Datentyp	Bedeutung
.inp_ptr	ushort	DPRAM-Adresse der Eingangsdaten
.outp_ptr	ushort	DPRAM-Adresse der Ausgangsdaten
.diag_ptr	ushort	DPRAM-Adresse der Diagnosedaten
.diag_len_ptr	ushort	DPRAM-Adresse der Diagnoselänge
.diag_cnt_ptr	ushort	DPRAM-Adresse des Diagnosezählers
.inp_len	byte	Anzahl Eingangsdatenbyte
.outp_len	byte	Anzahl Ausgangsdatenbyte
.inout	byte	Information über die Konsistenz der Ein-/Ausgänge
.slave_type	byte	Typ des Slaves

m044_get_slave_diagnosis_list Lies Diagnoseliste aller Slaves

Pascal FUNCTION m044_get_slave_diagnosis_list (micro_slot: byte; VAR data: m044_slv_bitmap_type) : word;

C ushort EXPORT m044_get_slave_diagnosis_list (byte micro_slot, m044_slv_bitmap_type *data);

Funktion Diese Funktion (25) liest die Slavediagnoseliste. Jedes Bit der gelesenen Daten repräsentiert jeweils einen Slave (Beispiel: Slave 17=Bit-1 von Feld slave_16_31). Ein gesetztes Bit bedeutet, daß der entsprechende Slave eine Diagnose gemeldet hat.

Parameter *data*: Adresse einer Strukturvariablen vom Typ m044_slv_bitmap_type (siehe folgende Tabelle)

Der Variablentyp m044_slv_bitmap_type ist eine Datenstruktur, bestehend aus folgenden Feldern:

Feld	**Datentyp**	**Bedeutung**
.slave_0_15	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_16_31	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_32_47	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_48_63	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_64_79	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_80_95	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_96_111	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_112_123	ushort	Bitfeld (Bit-0 .. Bit-11 gültig)

m044_check_slave_diagnosis — Prüfe, ob Slave Diagnose gemeldet hat

Pascal FUNCTION m044_check_slave_diagnosis (micro_slot; slave; VAR diaglen: byte) : word;

C ushort EXPORT m044_check_slave_diagnosis (byte micro_slot, byte slave, byte *diaglen);

Funktion Diese Funktion (26) prüft, ob der angegebene Slave eine Diagnose gemeldet hat und liefert ggf. die Anzahl der Diagnosedaten in 'diaglen' zurück.

Parameter *slave*: Slaveadresse (3..123)

diaglen: Diagnoselänge in Byte (max. 250, 0=keine Diagnose)

m044_get_slave_diagnosis — Lies Diagnosedaten eines Slave

Pascal FUNCTION m044_get_slave_diagnosis (micro_slot: byte; slave: byte; size: byte; VAR data) : word;

C ushort EXPORT m044_get_slave_diagnosis (byte micro_slot, byte slave, byte size, void *data);

Funktion Diese Funktion (27) liest einen Diagnosedatenblock (Octet-1..Octet-6 siehe Norm, weitere Octets (Bytes) sind anwenderspezifisch) vom Diagnosekanal eines Slaves.

Parameter *slave*: Slaveadresse (3..123)

size: Anzahl zu lesender Byte

data: Adresse des Zieldatenpuffers

Diagnose *0*: Befehl korrekt ausgeführt

100: Fehler: Bus anders parametriert

101: Fehler: Konsistenzkonflikt

m044_get_data_transfer_list Lies Datentransferliste

Pascal FUNCTION m044_get_data_transfer_list (micro_slot: byte;
VAR data: m044_slv_bitmap_type) : word;

C ushort EXPORT m044_get_data_transfer_list (byte micro_slot,
m044_slv_bitmap_type *data);

Funktion Diese Funktion (28) liest die Datentransferliste. Jedes Bit der gelesenen Daten repräsentiert jeweils einen Slave (Beispiel: Slave 17=Bit-1 von Feld slave_16_31). Ein gesetztes Bit bedeutet, daß sich der entsprechende Slave im Zustand DATA (=Slave ist aktiviert) befindet.

Parameter *data*: Adresse einer Strukturvariablen vom Typ m044_slv_bitmap_type (siehe folgende Tabelle)

Der Variablentyp m044_slv_bitmap_type ist eine Datenstruktur, bestehend aus folgenden Feldern:

Feld	**Datentyp**	**Bedeutung**
.slave_0_15	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_16_31	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_32_47	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_48_63	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_64_79	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_80_95	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_96_111	ushort	Bitfeld (Bit-0 .. Bit-15 gültig)
.slave_112_123	ushort	Bitfeld (Bit-0 .. Bit-11 gültig)

m044_check_slave_active Prüfen, ob Slave aktiv ist

Pascal FUNCTION m044_check_slave_active (micro_slot; slave; VAR active: byte) : word;

C ushort EXPORT m044_check_slave_active (byte micro_slot, byte slave, byte *active);

Funktion Diese Funktion (29) prüft, ob sich der angegebene Slave im Zustand DATA (=aktiviert) befindet.

Parameter *slave*: Slaveadresse (3..123)

active: 0: Slave ist nicht aktiviert
1: Slave ist aktiviert

m044_get_master_status_struct Lies Statusstruktur des Masters

Pascal FUNCTION m044_get_master_status_struct (micro_slot: byte; VAR data: m044_mst_statusstr_type) : word;

C ushort EXPORT m044_get_master_status_struct (byte micro_slot, m044_mst_statusstr_type *data);

Funktion Diese Funktion (30) liest die Masterstatusstruktur.

Parameter *data*: Adresse einer Masterstatusvariablen vom Typ m044_mst_statusstr_type (siehe folgende Tabelle)

Der Variablentyp m044_mst_statusstr_type ist eine Datenstruktur, bestehend aus folgenden Feldern:

Feld	Datentyp	Bedeutung
.master_status	byte	Master-Status
.id_high	byte	Hardware-Identnummer (High)
.id_low	byte	Hardware-Identnummer (Low)
.master_hw_version	byte	Hardwareversion des Masters
.master_fw_version	byte	Firmware-Version des Masters
.user_hw_version	byte	Hardwareversion des Users
.user_fw_version	byte	Firmware-Version des Users

m044_get_system_error_struct Lies Systemfehlerstruktur

Pascal FUNCTION m044_get_system_error_struct (micro_slot: byte; VAR data: m044_mst_syserr_type) : word;

C ushort EXPORT m044_get_system_error_struct (byte micro_slot, m044_mst_syserr_type *data);

Funktion Diese Funktion (31) liest die Systemfehlerstruktur.

Parameter *data*: Adresse einer Systemfehlervariablen vom Typ m044_mst_syserr_type (siehe folgende Tabelle)

Der Variablentyp m044_mst_syserr_type ist eine Datenstruktur, bestehend aus folgenden Feldern:

Feld	Datentyp	Bedeutung
.component	ushort	Modulname der Firmware
.subcomponent	ushort	Unterkomponente des Moduls
.status	ushort	Statuswert
.error_number	ushort	Fehlernummer
.detail	ushort	Detail

Mögliche Einträge in der Datenstruktur:

.component	.subcomponent	.status	.error_number	.detail	Erklärung
0x1006	0x0001	Beliebig	0x0001	Beliebig	Antwortsemaphore in Antwortkanal noch gesetzt
0x1006	0x0002	Beliebig	0x0001	Beliebig	Fehler beim Löschen vom Parameterdatenblock
0x1006	0x0002	Beliebig	0x0002	Beliebig	Fehler beim Programmieren vom FLASH-EPROM
0x1006	0x0003	Beliebig	0x0001	Beliebig	Watchdog zum Host abgelaufen
0x1003	0x0001	Beliebig	Beliebig	Beliebig	Fehler in Parameterbinärdatei
0x1003	0x0002	Beliebig	Beliebig	Beliebig	Fehler in Parameterbinärdatei

m044_poll_system_error Lies Systemfehler per Polling

Pascal FUNCTION m044_poll_system_error (micro_slot: byte;
VAR syserr: word) : word;

C ushort EXPORT m044_poll_system_error (byte micro_slot,
ushort *syserr);

Funktion Diese Funktion (32) dient zum Pollen des Systemfehlers (=Feld "component" der Systemfehlerstruktur) des Masters. Sie ist nur dann erforderlich, falls beim Restart der Parameter _M044_ENABLE_IRQ nicht gesetzt wurde.

Parameter *syserr*: Feld "component" der Systemfehlerstruktur

Sonderfunktionen

m044_fw_timer Firmware-Timer aktivieren/deaktivieren

Pascal FUNCTION m044_fw_timer(micro_slot: byte; timer: word) : word;

C ushort EXPORT m044_fw_timer(byte micro_slot, ushort timer);

Funktion Diese Funktion (33) aktiviert bzw deaktiviert den Firmware-Timer des Masters. Dieser löst (wenn der Interrupt aktiviert ist) zyklisch den eingestellten Interrupt zur Basiskarte aus.

Parameter *timer*: Zyklusdauer als Faktor von 25,6 µs (x 0..65535)
timer = 0 deaktiviert den Timer

Diagnose *0*: Befehl korrekt ausgeführt

256: Fehler: Befehlssemaphore nicht gesetzt

Zugriff auf das Gate-Array des Moduls M-DPM-12

m044_get_fpga_version — Lies Version des Gate-Arrays

Pascal: FUNCTION m044_get_fpga_version (micro_slot: byte) : byte;

C: byte EXPORT m044_get_fpga_version (byte micro_slot);

Funktion: Diese Funktion liest die Version/Revision des Gate-Arrays:

Bit-0..Bit-3 : Revisionsnummer

Bit-4..Bit-7 : Versionsnummer

m044_set_modul_register — Modulregister setzen

Pascal: PROCEDURE m044_set_modul_register (micro_slot: byte; reg_type: byte; data: byte);

C: void EXPORT m044_set_modul_register (byte micro_slot, byte reg_type, byte data);

Funktion: Diese Prozedur setzt ein Modul-Register (LED- oder Interrupt-Select-Register).

Parameter:

reg_type: _M044_ISR: setze Interrupt-Select-Register

_M044_LER: setze LED-Register

data: zu setzendes Byte (Bedeutung siehe S. 7-13)

m044_get_modul_register — Modulregister lesen

Pascal: FUNCTION m044_get_modul_register (micro_slot: byte; reg_type: byte) : byte;

C: byte EXPORT m044_get_modul_register (byte micro_slot, byte reg_type);

Funktion: Diese Prozedur liest ein Modul-Register.

Parameter: *reg_type*:
- _M044_ISR: lies Interrupt-Select-Register
- _M044_IST: lies Interrupt-Status-Register
- _M044_LER: lies LED-Register

m044_set_timeout_counter — Timeout-Zähler setzen

Pascal: PROCEDURE m044_set_timeout_counter (micro_slot: byte; data: word);

C: void EXPORT m044_set_timeout_counter (byte micro_slot, ushort data);

Funktion: Diese Prozedur setzt den Timeout-Zählers des Moduls. Dieser Zähler wird zur Konsistenzsteuerung verwendet.

Parameter: *data*: zu setzender Wert als Faktor von 10 µs (0..65535)

m044_get_timeout_counter — Timeout-Zähler lesen

Pascal: FUNCTION m044_get_timeout_counter (micro_slot: byte) : word;

C: ushort EXPORT m044_get_timeout_counter (byte micro_slot;

Funktion: Diese Prozedur liest den Timeout-Zähler des Moduls. Timeout = Rückgabewert (0..65535 x 10 µs)

m044_set_cons — Konsistenzanforderung setzen

Konsistente Zugriffe:

Pascal PROCEDURE m044_set_cons (micro_slot: byte; mode: byte);

C void EXPORT m044_set_cons (byte micro_slot, byte mode);

Funktion Diese Prozedur setzt eine Konsistenzanforderung.

Parameter *mode*: Lesekonsistenz: 1
Schreibkonsistenz: 2

m044_clear_cons — Konsistenzanforderung beenden und Modulstatus lesen

Pascal FUNCTION m044_clear_cons (micro_slot: byte) : byte;

C byte EXPORT m044_clear_cons (byte micro_slot);

Funktion Diese Prozedur beendet die Konsistenzanforderung und liest den Modul-Status (siehe m044_get_modul_status).

m044_get_modul_status — Gate-Array-Status lesen

Pascal FUNCTION m044_get_modul_status (micro_slot: byte) : byte;

C byte EXPORT m044_get_modul_status (byte micro_slot);

Funktion Diese Prozedur liest den Gate-Array-Status (siehe nachfolgende Übersicht).

	0	**0**	**0**	**0**	**0**	**0**	**0**
							0
						0	
					0		
				0			
			0				
		0					
	0						

Modulstatus

Timeout-Status
0=kein Fehler, 1=Fehler

Konsistenzstatus
0=kein Fehler, 1=Fehler

Lesekonsistenzanforderung
0=aktiv, 1=nicht aktiv

Schreibkonsistenzanforderung
0=aktiv, 1=nicht aktiv

ASPC 2 Konsistenzanforderung
0=nicht aktiv, 1=aktiv

Host-Konsistenz Acknowledge
0=aktiv, 1=nicht aktiv

Host Zugriff auf DPRAM
0=aktiv, 1=nicht aktiv

Inbetriebnahme

Zur Inbetriebnahme des PROFIBUS-DP Masters M-DPM-12 ist es notwendig, die Konfiguration der Anlage (bestehend aus Master und Slaves) in Form einer Binärdatei auf das Modul M-DPM-12 zu übertragen. Die Konfiguration wird auf dem Modul im Flash-Memory gespeichert, d.h. sie steht auch nach einem Reset (z.B. durch Power-Down) zur Verfügung und sollte nur einmal übertragen werden. Bei Änderungen der Konfiguration der Anlage muß eine neue Binärdatei erzeugt und die alte Konfiguration auf dem Modul überschrieben werden.

Die mitgelieferte Hochsprachenbibliothek M044_LIB stellt Funktionen zur Verfügung, um eine Binärdatei auf das Modul zu übertragen (Download). Die aktuelle Konfiguration kann auch vom Modul geladen werden (Upload).

Zur Erstellung der Konfigurationsdatei (Binärdatei) wird die Software "COM PROFIBUS" von Siemens eingesetzt.

Installationshinweise

Die Siemens Software COM PROFIBUS muß unter Windows installiert werden. Legen Sie Diskette 1 ein und rufen Sie das Programm INSTALL.EXE auf. Bitte beachten Sie, daß bei der Installierung unter 'Optionen' die Memory Card Treiber deaktiviert werden müssen!

Mit dem Modul M-DPM-12 wird die Typdatei für das M-DPM-12 Modul auf Diskette mitgeliefert (Unterverzeichnis "TYPDATEI"). Bitte kopieren Sie alle Dateien aus den Unterverzeichnissen \MASTERS\ und \BITMAPS\ in die gleichnamigen Unterverzeichnisse von COM PROFIBUS.

Bedienungshinweise (siehe auch COM PROFIBUS Online-Hilfe)

Erstellen Sie ein neues Projekt (Datei\Neu) und wählen Sie den **Master M-DPM-12** als **Stationsnummer 1** aus.

Anschließend können z.B. Siemens ET200 Slave-Stationen angeklickt und an den PROFIBUS 'angehängt' werden. Vergeben Sie eine Stationsnummer, und konfigurieren Sie den Slave. Im Untermenü 'Konfigurierung' kann der Eintrag in den Spalten 'E-Adr.' und 'A-Adr.' entfallen, da diese Adressierungsart nicht unterstützt wird.

Sind alle Slaves eingetragen, müssen die Busparameter (z.B. die Baudrate) eingestellt werden. Im Menüpunkt Parametrieren\Busparameter können Sie die Baudrate eingeben. Die maximal einstellbare Baudrate wird durch den 'langsamsten' Slave begrenzt.

Anschließend kann das Projekt gespeichert werden. Unter Datei\Export kann eine Binärdatei erstellt werden.

Programmierung mit Hilfe der Hochsprachenbibliothek

Die Binärdatei muß mit der Bibliotheksfunktion **m044_download_file** auf das Modul M-DPM-12 übertragen werden. Nach einem Download muß ein Reset des Moduls erfolgen. Danach befindet sich das M-DPM-12 im Zustand 'STOP'. Der aktuelle Betriebszustand kann mit der Funktion **m044_get_master_mode** ermittelt werden.

Die Funktion **m044_set_master_mode** setzt den Master in den Operate-Zustand, die bei der Konfiguration angegebenen Slaves werden zyklisch angesprochen. Für jeden aktiven Slave wird in die Datentransferliste ein Bit gesetzt. Ob ein Slave aktiv ist, kann auch mit der Funktion **m044_check_slave_active** überprüft werden. Liegen Diagnosedaten eines Slave vor, wird ein Bit in der Diagnoseliste gesetzt. Ob ein Slave Diagnosedaten gemeldet hat, kann mit der Funktion **m044_check_slave_diagnosis** abgefragt werden.

Der Datenaustausch zwischen Master und Slave erfolgt mit den Bibliotheksfunktionen **m044_get_slave_xxx** bzw. **m044_set_slave_xxx**. Dabei können die Nutzdaten byte-, wort- oder blockweise übergeben werden.

Ob ein Fehler aufgetreten ist, kann mit der Funktion **m044_poll_system_error** ermittelt werden. Diese Funktion sollte zyklisch aufgerufen werden.

Programmierbeispiel

Ansprechen eines Slave mit 4 Byte Eingangs- und 4 Byte Ausgangs-Nutzdaten:

```
/* Master in Operate-Zustand setzen (einmalig)         */
 m044_set_master_mode(microslot, _M044_OPERATE);
 m044_refresh_slave_datastruct(microslot);

/* Prüfen, ob Slave aktiv ist                          */
 m044_check_slave_active(microslot, slave, &active);
 if(active)
  {
/* Austausch der Nutzdaten (je 4 Byte)                 */
   m044_set_slave_data_block(microslot, slave, 4, &dout);
   m044_get_slave_data_block(microslot, slave, 4, &din);
  }
/* Prüfen, ob Slave Diagnose gemeldet hat              */
m044_check_slave_diagnosis(microslot, slave, &diaglen);
if(diaglen > 0)
  {
   m044_get_slave_diagnosis(microslot, slave, diaglen, &diag);
  }
```

Programmierung mit I/O-Zugriffen

Dieses Kapitel ist für jene Anwender gedacht, die eigene Anwendungsprogramme für die Basiskarte MODULAR-4/486 schreiben wollen.

Lokale I/O-Adressen

Alle Adressen sind in hexadezimaler Schreibweise angegeben. Nicht benutzte Bits sind reserviert und sollten beim Schreiben zu 0 gesetzt werden. Beim Lesen sind diese Bits ungültig.

Adresse	Zugriff[1]	Funktion
MBA+00h	RW16	DPRAM-Pointer setzen/lesen (0000h-3fffh) Bit 0 dient zur Selektion von Low- oder High-Byte bei Byte-Zugriffen auf das DPRAM (Adreßleitung A0).
MBA+03h	RW8	LED Register (LER) Beim Lesen geben die Bits 0 und 1 den Zustand der LEDs 1 und 2 an (0 = aus, 1 = ein). Beim Schreiben selektieren die Bits 0 und 1 die LEDs 1 und 2, die Bits 2 und 3 geben den zu setzenden Wert an.

[1] R16 : 16 Bit Lesezugriff, W16 : 16 Bit Schreibzugriff, RW16 : 16 Bit Lese- oder Schreibzugriff, R8 : 8 Bit Lesezugriff, W8 : 8 Bit Schreibzugriff, RW8 : 8 Bit Lese- oder Schreibzugriff, W8x = 8 Bit Schreibzugriff, beliebige Daten

Adresse	Zugriff[1]	Funktion
MBA+01h	RW8	Interrupt-Select-Register (ISR) / Konflikterkennung Bit 0 bis 2 bestimmen die Interrupt-Leitung zur Basiskarte: 0 0 0 keine Interrupt-Leitung angewählt 0 0 1 IRQ-A 0 1 0 IRQ-B 0 1 1 IRQ-C 1 0 0 IRQ-D 1 0 1 IRQ-E 1 1 0 IRQ-F 1 1 1 NMI Bit 4 Interrupt durch XINTH (1 = enable) Bit 5 gibt an, wie auf Konsistenzkonflikte reagiert wird: 0 Timeout-Zähler aktiv (kein Interrupt) 1 Interrupt bei A-CONS = 1 & XHKAK = 0 Bit 6 Interrupt per Firmware-Timer (1 = enable)
MBA+05h	R8	Interrupt-Status-Register (IST) Ein gesetztes Bit (=1) signalisiert einen Interrupt: Bit 0 XINTH Interrupt aktiv Bit 1 KSTATUS Interrupt aktiv Bit 2 Firmware-Timeout Interrupt aktiv
MBA+05h	W8	Reset PROFIBUS-Master Data = 8fh aktiviert XRESET Data = 0fh deaktivert XRESET
MBA+07h	R8	Gate-Array-Version lesen Bit 0 bis 3 = Revisionsnummer, Bit 4 bis 7 = Versionsnummer z.B.: 0010 0101 = 25h = Version 2, Revision 5

Adresse	**Zugriff**[1]	**Funktion**
MBA+07h	W8x	Gate-Array und PROFIBUS-Master-Reset durchführen Alle Register werden = 0 gesetzt. Der Timeout-Zähler wird auf ffffh gesetzt.
MBA+08h	RW16	Timeout-Zähler TOR (16 Bit) setzen/lesen Einstellbar sind Werte von 0 bis 65535 (x 10µs).
MBA+10h	RW16	DPRAM-Wort lesen/schreiben
MBA+12h	RW16	DPRAM-Byte[2] lesen/schreiben
MBA+14h	RW16	DPRAM-Wort lesen/schreiben, anschließend wird der Adreß-Pointer um 2 inkrementiert
MBA+16h	RW16	DPRAM-Byte lesen/schreiben, anschließend wird der Adreß-Pointer um 1 inkrementiert
MBA+18h	W8	Konsistenzanforderung setzen/löschen Data = 00h: Konsistenzanford. und Modulstatus löschen Data = 01h: Lesekonsistenz (XRHCONS = 0) Data = 02h: Schreibkonsistenz (XWHCONS = 0)

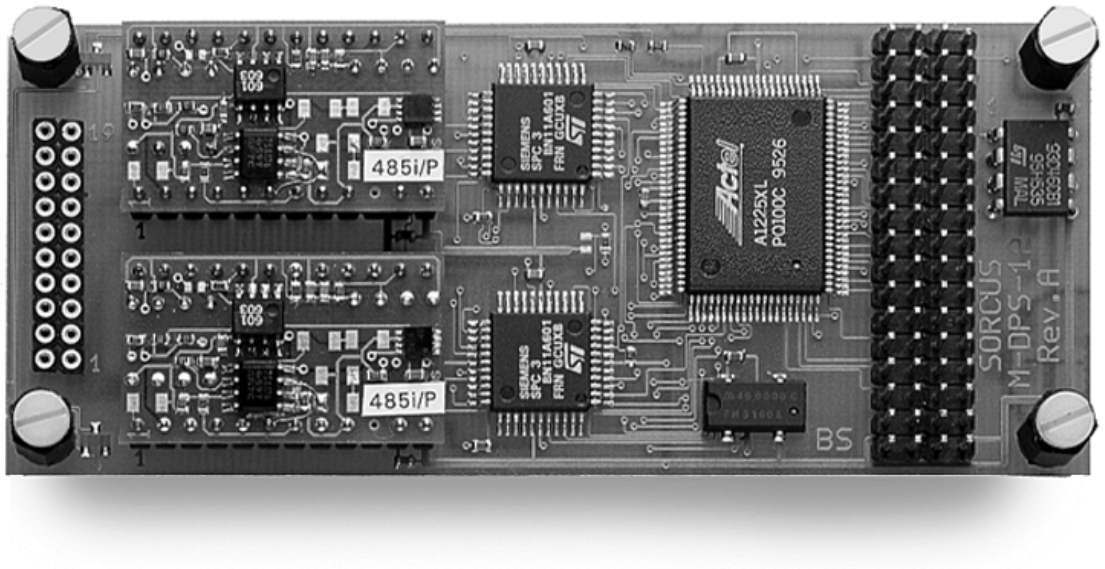
[1] R16 : 16 Bit Lesezugriff, W16 : 16 Bit Schreibzugriff, RW16 : 16 Bit Lese- oder Schreibzugriff, R8 : 8 Bit Lesezugriff, W8 : 8 Bit Schreibzugriff, RW8 : 8 Bit Lese- oder Schreibzugriff, W8x = 8 Bit Schreibzugriff, beliebige Daten

[2] Bei den Bytezugriffen auf das DPRAM selektiert Bit 0 des Adreß-Pointers ob das Low- oder das High-Byte verwendet wird (Bit 0 = 0 : Lowbyte, Bit 0 = 1 : Highbyte). Die Basiskarte muß allerdings Wortzugriffe durchführen.

Adresse	Zugriff[1]	Funktion
MBA+18h	R8	Konsistenzanforderung beenden und Modulstatus lesen Bit 0 = TIMEOUT-STATUS (0=OK, 1=ERROR) Bit 1 = KONSISTENZ-STATUS (0=OK, 1=ERROR) Bit 2 = R-CONS (Lesekonsistenzanforderung, low-aktiv) Bit 3 = W-CONS (Schreibkonsistenzanford., low-aktiv) Bit 4 = A-CONS (APSC 2 Konsistenzanforderung) Bit 5 = XHKAK (Host-Konsistenz Acknowledge, low-aktiv) Bit 6 = XCSDPR2 (Host-Zugriff auf DPRAM, low-aktiv)
MBA+19h	R8	Modulstatus lesen siehe MBA+0x18h

8. M-DPS-12

PROFIBUS Slave, 2 Kanäle bis 12 MBaud



Programmierung 8-13

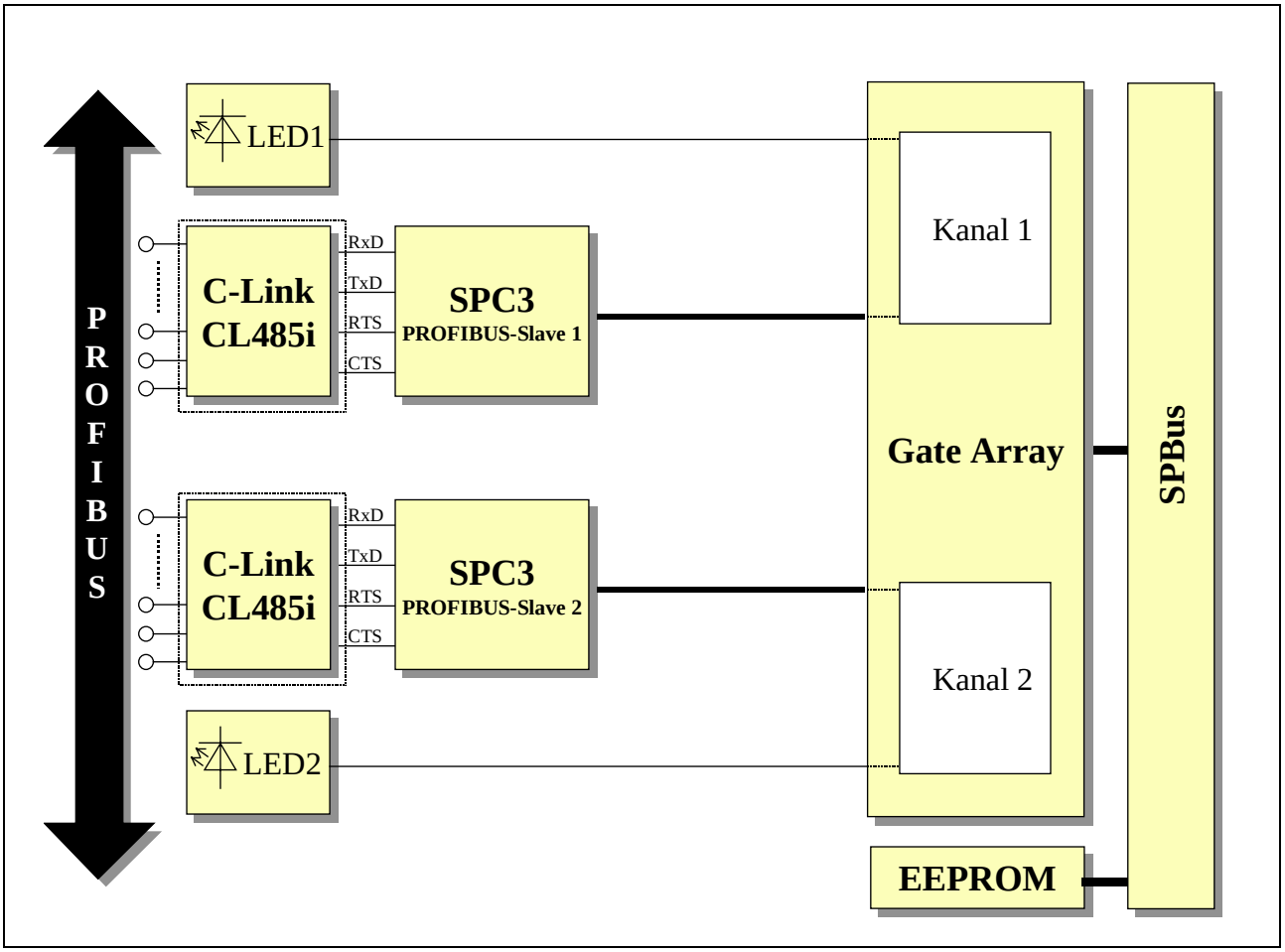
Hochsprachenbibliotheken 8-17

Programmierung mit I/O-Zugriffen 8-36

Funktionsbeschreibung

M-DPS-12 ist ein PROFIBUS-DP-Slave-Modul für SORCUS MODULAR-4/486 Basiskarten. Es enthält zwei unabhängige Slave-Kanäle, die alle Baudraten inkl. 12 MBaud unterstützen. Die verwendeten PROFIBUS Slave-Controller (Siemens SPC3[1]) entlasten die Basiskarte durch vollständige interne Abwicklung des PROFIBUS-DP[2] Protokolls.

Blockschaltbild



[1] SPC3 :Siemens PROFIBUS Controller, dritte Generation

[2] DP : Dezentrale Peripherie

Technische Daten

Parameter	Wert	Einheit
PROFIBUS Slave-Controller	2 x Siemens SPC3	-
physikalische Schnittstellen per C-Link (Standard: CL485i)	2	-
serielles EEPROM für Konfigurationsdaten	32	Wörter
Interruptfähig zur Basiskarte (Interrupt-Kanal je Kanal (Slave) per Software anwählbar)	ja	-
Versorgungsspannungen (von der Basiskarte)	+5	V
Stromaufnahme bei +5 V (typ., extern nichts angeschlossen, LED 1 und 2 aus)	170	mA
Betriebstemperaturbereich	0 bis 60	° C
Abmessungen (L x B x H)	106 x 45 x 15	mm

Lieferumfang

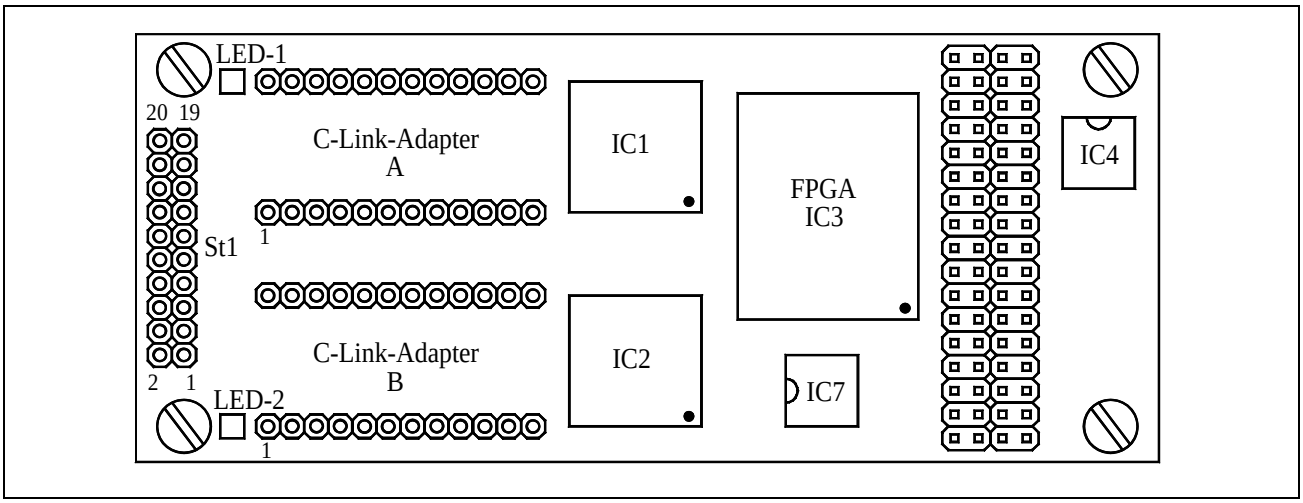
Zum Lieferumfang des Moduls gehören:

- Modul M-DPS-12
- 20-poliger Pfostenstecker für Flachbandkabel
- Datenträger mit Programmbibliotheken

Konfiguration und Einbau

Vor dem Aufstecken des Moduls auf die Basiskarte muß der bzw. müssen die beiden C-Link-Adapter aufgesteckt werden (Pin 1 ist auf dem Modul und auf dem C-Link gekennzeichnet). Das Modul und die C-Links enthalten keine Jumper, alle Einstellungen werden nach dem Einbau per Software vorgenommen.

Lageplan



EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0001	0011 0000	212dh	Modultyp M-DPS-12
1	0000 0000	0000 0000	0000h	Initialisierung
2	0000 0000	0000 0000	0000h	Interrupt-Kanal für Slave-Kanal 1
3	0000 0000	0000 0000	0000h	LED 1 Einstellung
4	0000 0000	0000 0000	0000h	Interrupt-Kanal für Slave-Kanal 2
5	0000 0000	0000 0000	0000h	LED 2 Einstellung
6	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	0	1	1	0	1	WORT-0: Kennung
								0	0	1	0	1	1	0	1	Modultyp (45 = M-DPS-12)
				0	0	0	1									Revision: 1 = Rev. A, 2 = Rev. B
			0													Reserve
0	0	1														Kennung

WORT-1: Initialisierung

Hier hat nur Bit 0 z. Zt. eine Bedeutung. Wenn dieses Bit = 1 gesetzt ist, werden die Register des Moduls nach einem Hardware-Reset entsprechend den Daten in EEPROM konfiguriert und initialisiert.
Wenn Bit 0 = 0 gesetzt ist, wird das Modul nach einem Hardware-Reset (bzw. nach dem Einschalten des Systems) nicht automatisch konfiguriert und initialisiert.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															0	Init nach Hardware-Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

Die EEPROM-Inhalte der Wörter 2 bis 5 dienen zur Abspeicherung einer anwenderspezifischen Modulkonfiguration. Die EEPROM-Inhalte werden nicht direkt (per Hardware) in die entsprechenden Register des Moduls übernommen, sondern können vom Anwenderprogramm gelesen und zur Programmierung der Register (siehe Abschnitt 'Programmierung') verwendet werden. **!**

WORT-2: Interrupt-Kanal zur Basiskarte für PROFIBUS-Slave 1 (SPC3/1)

Hier kann der Anwender die Konfiguration für den Interrupt-Kanal der Basiskarte abspeichern, mit dem der PROFIBUS-Slave 1 (Kanal 1) des Moduls verbunden wird.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-2: Interrupt-Kanal für Slave 1 (werks. Einst.)
																geändert am: von:
													0	0	0	Interrupt-Kanal 0 = kein Interrupt 1 = IRQ-A 2 = IRQ-B 3 = IRQ-C 4 = IRQ-D 5 = IRQ-E 6 = IRQ-F 7 = NMI
0	0	0	0	0	0	0	0	0	0	0	0	0				Reserviert

! *Bitte beachten Sie, daß sowohl unterschiedliche Interrupts wie auch der gleiche Interrupt für die beiden Slaves verwendet werden kann.*

WORT-3: LED 1

Hier kann der Sollstatus bzw. die Betriebsart der Leuchtdiode LED 1 für Kanal 1 abgespeichert werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-3: LED 1 (werks. Einst.)
																geändert am: von:
															0	LED 1 Select (Betriebsart) 0 : SPC3/1 steuert LED 1 1 : LED 1 entsprechend Bit 1
														0		LED 1 0 : ein, 1 : aus
0	0	0	0	0	0	0	0	0	0	0	0	0	0			Reserviert

WORT-4: Interrupt-Kanal zur Basiskarte für PROFIBUS-Slave 2 (SPC3/2)

Hier kann der Anwender die Konfiguration für den Interrupt-Kanal der Basiskarte abspeichern, mit dem der PROFIBUS-Slave 2 (Kanal 2) des Moduls verbunden wird.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-4: Interrupt-Kanal für Slave 2 (werks. Einst.)
																geändert am: von:
													0	0	0	Interrupt-Kanal 0 = kein Interrupt 1 = IRQ-A 2 = IRQ-B 3 = IRQ-C 4 = IRQ-D 5 = IRQ-E 6 = IRQ-F 7 = NMI
0	0	0	0	0	0	0	0	0	0	0	0	0				Reserviert

! *Bitte beachten Sie, daß sowohl unterschiedliche Interrupts wie auch der gleiche Interrupt für die beiden Slaves verwendet werden kann.*

WORT-5: LED 2

Hier kann der Sollstatus bzw. die Betriebsart der Leuchtdiode LED 2 für Kanal 2 abgespeichert werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-5: LED 2 (werks. Einst.)
																geändert am: von:
															0	LED 2 Select (Betriebsart) 0 : SPC3/2 steuert LED 2 1 : LED 2 entsprechend Bit 1
														0		LED 2 0 : ein, 1 : aus
0	0	0	0	0	0	0	0	0	0	0	0	0	0			Reserviert

Steckerbelegung

Kanal 2			Kanal 1		
Pin	**Signal**	**Bedeutung**	**Pin**	**Signal**	**Bedeutung**
1	DPPE	optional	11	DPPE	optional
2	DP5V	+ 5 Volt, isoliert	12	DP5V	+ 5 Volt, isoliert
3	-	n.c.	13	-	n.c.
4	-	n.c.	14	-	n.c.
5	DPB	DPB, isoliert	15	DPB	DPB, isoliert
6	DPA	DPA, isoliert	16	DPA	DPA, isoliert
7	DPRTS	Request To Send, isol.	17	DPRTS	Request To Send, isol.
8	-	n.c.	18	-	n.c.
9	DPGND	Ground, isoliert	19	DPGND	Ground, isoliert
10	-	n.c.	20	-	n.c.

***Tabelle 8-1**: Pinbelegung des Pfostensteckers St1*

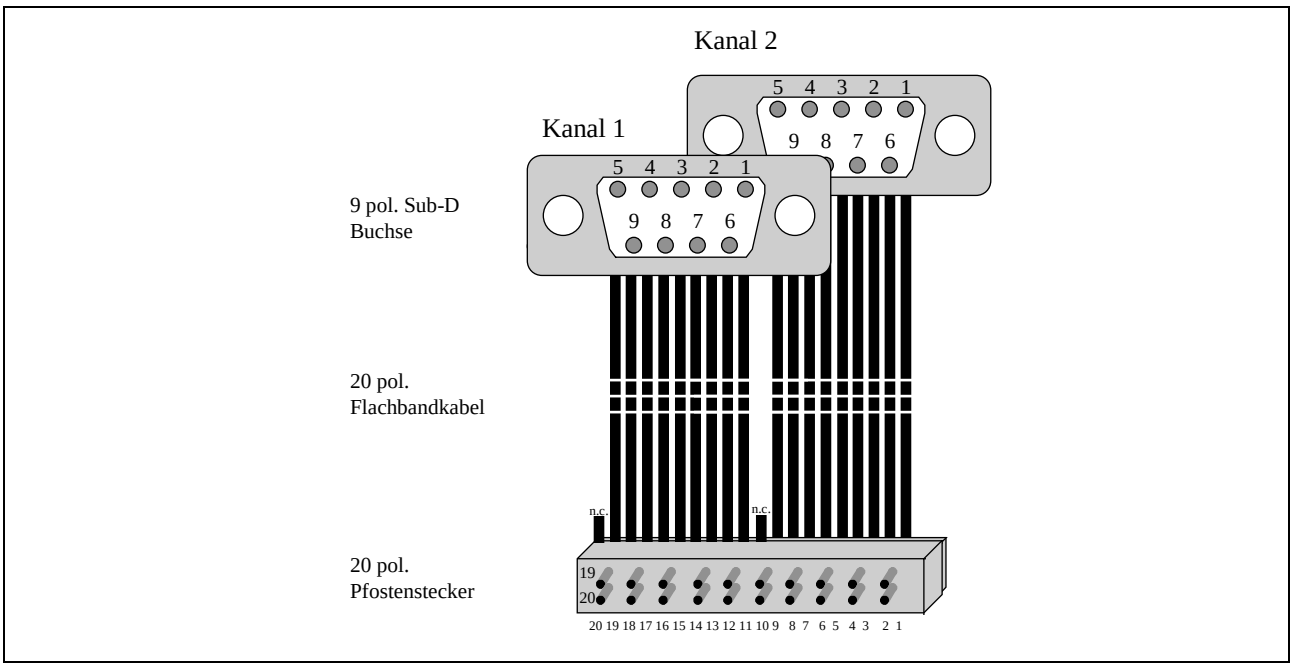


Abbildung 8-1: Kabel für M-DPS-12 an Siemens ET200

Programmierung

Auf dem Modul müssen (für jeden Slave-Kanal getrennt) folgende Funktionsgruppen programmiert werden:

- Anwahl einer Interrupt-Leitung vom Modul zur Basiskarte

(Interrupt-Select-Register IR1 und IR2, je 3 Bit)

- LED 1 und 2

(LED-Register LR1 und LR2, je 2 Bit)

- Die Adreß-Pointer der beiden PROFIBUS Slave-Controller (SPC3/1 und SPC3/2)

(AP1 und AP2, je 11 Bit)

- Die beiden Slave-Controller SPC3/1 und SPC3/2

Reset des Moduls

Die Initialisierung des Moduls kann auf verschiedene Arten durchgeführt werden. Ein Hardware-Reset initialisiert sowohl das Gate-Array sowie die beiden Slave-Controller. Dies kann auch durch einen I/O-Schreibzugriff (siehe Abschnitt 'Lokale I/O-Adressen', Modul-Reset) erfolgen.

Im laufenden Betrieb können die beiden Kanäle durch entsprechende I/O-Schreibzugriffe getrennt voneinander initialisiert werden. Dabei wird sowohl der Slave-Controller als auch die zugehörigen Register und Adreß-Pointer zurückgesetzt.

Achtung: Ein Reset eines Slave-Controllers benötigt 400 Takte (bei 48 MHz). Der Slave-Controller kann während dieser Zeit (ca. 10 µs, wird vom Gate-Array erzeugt) nicht angesprochen werden. Die Basiskarte kann den Zustand der Reset-Leitungen XRESET1 und XRESET2 (high-aktiv) zurücklesen. **!**

Anwahl einer Interrupt-Leitung vom Modul zur Basiskarte

Das Modul ist interruptfähig, d.h. es kann bei bestimmten Ereignissen einen Interrupt auf der Basiskarte auslösen. Die Interrupt-Leitung eines Slaves kann per Software mit einem der Interrupt-Eingänge der Basiskarte verbunden werden. Es ist möglich, daß beide Slave-Controller denselben Interrupt verwenden. Die Anwender-Software auf der Basiskarte muß dann zusätzlich den auslösenden Controller (die Interrupt-Quelle) ermitteln.

Während der Einstellung des Interrupt-Kanals darf das Modul keinen Interrupt anfordern bzw. auf der Basiskarte müssen die Interrupts (vorübergehend) maskiert werden.
Die Anwahl eines Interrupts geschieht durch Setzen des Interrupt-Select-Register IR1 bzw. IR2 des Moduls (nur Bit 0 bis 2 werden benutzt, die restlichen Bits sind reserviert und sollten auf 0 gesetzt werden).

Die Interrupt-Select-Register können auch von der Basiskarte gelesen werden. Dabei wird zusätzlich der aktuelle Zustand der _XINT1- und der _XINT2-Leitung (in Bit 3 und 4) geliefert. Die Bits 5 bis 7 sind ungültig.

Ein Interrupt bleibt solange aktiviert (_XINT1 bzw. _XINT2 = 0), bis der Anwender ein End-Of-Interrupt (EOI) an den Slave-Controller sendet.

Bit 2	Bit 1	Bit 0	Interrupt-Leitung zur MODULAR-4/486
0	0	0	keine
0	0	1	IRQ-A
0	1	0	IRQ-B
0	1	1	IRQ-C
1	0	0	IRQ-D
1	0	1	IRQ-E
1	0	1	IRQ-F[1]
1	1	1	NMI

[1] IRQ-F wird auch für Zugriffe auf das EEPROM des Moduls verwendet.

LED 1 und 2

Die beiden LEDs auf dem Modul werden getrennt voneinander programmiert. Dazu gibt es auf dem Modul zwei Register (LED-Register 1 und 2, LR1/2), in denen die Betriebsart einer LED eingestellt wird.

Bit 0 eines LED-Registers legt fest, ob die LED durch den zugehörigen Slave-Controller (Bit 0 = 0) oder durch das Anwenderprogramm angesteuert wird. Ist Bit 0 = 1, wird die Leuchtdiode entsprechend Bit 1 (0 = ein, 1 = aus) geschaltet.
Die Bits 2 bis 7 sind ungültig und sollten zu 0 gesetzt werden.

Bit 0	**LED 1/2 Mode Select**
0	Ansteuerung von LED 1 bzw. 2 durch Slave-Controller SPC3/1 bzw. /2
1	LED 1 bzw. 2 entsprechend Bit 1

Bit 1	**LED 1/2 On/Off**
0	LED 1 bzw. 2 eingeschaltet
1	LED 1 bzw. 2 ausgeschaltet

Den Status der Leuchtdioden bzw. das LED-Register kann auch von der Basiskarte gelesen werden (jeweils 3 Bit, Bit 2 gibt den aktuellen Zustand der LED an: 0 = ein, 1 = aus; die Bits 3 bis 7 sind ungültig).

Adreß-Pointer für die Slave-Bausteine SPC3

Zur Adressierung der beiden Slave-Controller enthält das Modul zwei unabhängige, programmierbare Adreß-Pointer (AP1 und AP2, je 11 Bit). Bei I/O-Lese- bzw. Schreibzugriffen auf die Slave-Controller wird der jeweilige Adreß-Pointer verwendet. Spezielle Zugriffe ermöglichen eine automatische Inkrementierung des gerade verwendeten Adreß-Pointers. Einstellbar sind Adressen von 0000h bis 07ffh, gültige SPC3 Adressen liegen im Bereich von 0000h bis 05ffh.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Slave-Controller Adreß-Pointer
0	0	0	0	0												11 Bit Adreß-Pointer (A0 bis A10)
0	0	0	0	0												Reserviert

Die Bits 11 bis 15 werden nicht verwendet und sollten auf 0 gesetzt werden.

Hochsprachenbibliotheken

Wie die Bibliothek eingebunden und verwendet wird, finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliotheken'. Der Name der Bibliothek (*libname*) lautet **M045_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Prozedur **m045_bib_startup** einmal aufgerufen werden.

Die Parameter **micro_slot** und **channel** werden in allen Routinen verwendet und geben den Modulsteckplatz an bzw. die Kanalnummer (0 oder 1) an. Der Modulsteckplatz kann die Werte 1 bis 4 annehmen, bei Einsatz einer MODULAR-4/486 Karte mit Modulextender auch die Werte 5 bis 9.

Verwendung der Bibliothek mit der Treibertask M045TASK

Mit den Bibliotheken M045_LIB wird zusätzlich eine II-Task M045TASK.EXE mitgeliefert. In den Funktionen und Prozeduren der Bibliothek werden alle PROFIBUS-Funktionen per Taskfunktion ausgeführt. Das hat den Vorteil, daß in einer Multitasking-Umgebung mehrere Anwendungen sich ein Modul teilen können. Evtl. auftretende Konflikte werden durch die Task abgefangen.

Die Task M045TASK.EXE hat die Programmnummer 0331h und muß mit den Flags 0989h auf der MODULAR-4/486 Karte installiert werden. Eine globale Variable *tasknum* vom Typ *ushort* muß vom Anwenderprogramm deklariert werden und mit der Tasknummer voreingestellt sein. Dies kann auch mit **m045_set_tasknum** (tasknummer) vor Aufruf der ersten Bibliotheksfunktion geschehen.

m045_bib_startup — Modulbibliothek initialisieren

Pascal	PROCEDURE m045_bib_startup(micro_slot, channel: byte);
C	void EXPORT m045_bib_startup(byte micro_slot, byte channel);
Funktion	Diese Prozedur initialisiert die Modul-Bibliothek M045_LIB.
Parameter	*channel*: definiert den Kanal, der initialisiert werden soll

m045_init Initialisierung eines Slave-Kanals

Pascal PROCEDURE m045_init(micro_slot, channel, irq_num: byte; m_dps_12 m_dps_12_type);

C void EXPORT m045_init(byte micro_slot, byte channel, byte irq_num, m_dps_12_type m_dps_12);

Funktion Die Prozedur **m045_init** dient zur Initialisierung eines Slave-Kanals. Alle relevanten Einstellungen können per Parameter eingestellt werden.

Parameter *channel*: definiert den Kanal, der initialisiert werden soll.

irq_num: definiert die Interruptleitung der Basiskarte, mit dem die Interruptleitung des angegebenen Kanals verbunden werden soll.

m_dps_12: definiert eine Struktur (m_dps_12_type), die alle Parameter enthält, die für die Initialisierung der Register des angegebenen Kanals erforderlich sind:

Der Aufbau des RECORD (bzw. STRUCT in C) m_dps_12_type:

Feldbezeichner	Typ	Bedeutung
.indication	Word	Benachrichtigungsgründe (Wort), siehe unten
.user_wd_val	Word	User-Watchdog-Wert
.this_station	Byte	Stationsadresse
.ident_num	Word	PNO-Identnummer
.change_station_number	Byte	Gibt an, ob die Stationsadresse über PROFIBUS DP geändert werden kann: TRUE (=1) = ja, FALSE (=0) = nein
.hw_mode	Word	Hardware Modus (Wort), siehe unten
.mintsdr	Byte	Antwortzeit beim Hochlauf
.dps2_buf	Struktur dps2_bufinit	Struktur für Pufferinitialisierung (Längen der Puffer)
.cfg_akt	Array	Konfigurationspuffer
.cfg_len_akt	Byte	Länge des aktuellen Konfigurationspuffers
.i_data	Pointer	Pointer auf den Eingangsdatenbereich (Daten vom Master)

Aufbau des RECORD (STRUCT in C) dps2_bufinit:

Feldbezeichner	**Typ**	**Bedeutung**
.din_dout_len	Word	Länge der Eingangs- und Ausgangsdatenbereiche (0 - 488)
.diag_buf_len	Byte	Länge des Diagnosepuffers (6 - 244)
.prm_buf_len	Byte	Länge des Parameterpuffers (7 - 244)
.cfg_buf_len	Byte	Länge des Konfigurationspuffers (1..244)
.ssa_buf_len	Byte	Länge des Set-Slave-Adreß-Puffers (0 / 4..244)

Mögliche Werte für Parameter **hw_mode (**in Struktur m_dps_12_type):

Wert	**Konstantenbezeichner**	**Bedeutung**
0x0000	INT_POL_LOW	Der Interrupt-Ausgang ist low-aktiv
0x0010	INT_POL_HIGH	Der Interrupt-Ausgang ist high-aktiv
0x0020	EARLY_RDY	Ready wird um einen Takt verzögert
0x0040	SYNC_SUPPORTED	Der Sync-Mode wird unterstützt
0x0080	FREEZE_SUPPORTED	Der Freeze-Mode wird unterstützt
0x0100	DP_MODE	DP-Mode freigeben (ist immer gesetzt)
0x0000	EOI_TIMEBASE_1u	Die Interrupt-Inaktivzeit ist mindestens 1 µs lang
0x0200	EOI_TIMEBASE_1m	Die Interrupt-Inaktivzeit ist mindestens 1 ms lang
0x0000	USER_TIMEBASE_1m	Der User-Timer kommt alle 1 ms
0x0400	USER_TIMEBASE_10m	Der User-Time-Clock Interrupt kommt alle 10 ms

Mögliche Werte für Parameter **indication** (in Struktur m_dps_12_type):

Wert	**Konstantenbezeichner**	**Bedeutung**
0x0001	OFFLINE	Der SPC3 ist, nachdem er den aktuellen Auftrag abgearbeitet hat, in den Offline-Zustand gekommen (durch Setzen des Bits 'GO_OFFLINE').
0x0002	GO_LEAVE_DATA_EX	Die DP-State-Machine ist in den 'DATA_EX' Zustand eingetreten oder hat ihn verlassen.
0x0004	BAUDRATE_DETECT	Der SPC3 hat den 'BAUD_SEARCH' Zustand verlassen und eine Baudrate gefunden.

Wert	Konstantenbezeichner	Bedeutung
0x0008	WD_TIMEOUT	Im WD-Zustand 'DP_CONTROL' ist der Watchdog-Timer abgelaufen.
0x0010	USER_TIMER_CLOCK	Die Zeitbasis des frei verfügbaren User-Timer-Clocks ist abgelaufen (1/10 ms Timer-Tick)
0x0100	NEW_GC_COMMAND	Der SPC3 hat ein 'GLOBAL_CONTROL' Telegramm mit einem geänderten 'GC_COMMAND_BYTE' empfangen und dieses abgelegt.
0x0200	NEW_SSA_DATA	Der SPC3 hat ein 'SET_SLAVE_ADDRESS' Telegramm empfangen und die Daten im SSA-Puffer bereitgestellt.
0x0400	NEW_CFG_DATA	Der SPC3 hat ein 'CHECK_CFG' Telegramm empfangen und die Daten im CFG-Puffer bereitgestellt.
0x0800	NEW_PRM_DATA	Der SPC3 hat ein 'SET_PARAM' Telegramm empfangen und die Daten im PRM-Puffer bereitgestellt.
0x1000	DIAG_BUFFER_-CHANGED	Auf Anforderung durch 'NEW_DIAG_CMD' hat der SPC3 die Diagnosepuffer ausgetauscht und den alten wieder zur Verfügung gestellt.
0x2000	DATA_IN_OUT	Der SPC3 hat ein 'WRITE_READ_DATA' Telegramm empfangen und die neuen Output-Daten im Output-Puffer bereitgestellt. Bei 'Power-On' bzw. einem 'LEAVE_MASTER' löscht der SPC3 den Inhalt des Puffers und generiert auch diesen Interrupt.

m045_start_spc3 SPC3 starten

Pascal PROCEDURE m045_start_spc3(micro_slot, channel: byte);

C void EXPORT m045_start_spc3(byte micro_slot, byte channel);

Funktion Mit **m045_start_spc3** wird der SPC3 auf ´ONLINE´ geschaltet.

Parameter *channel*: definiert den Kanal, dessen SPC3 gestartet werden soll.

m045_retrigger_user_wd User-Watchdog zurücksetzen

Pascal PROCEDURE m045_retrigger_user_wd(micro_slot, channel: byte);

C void EXPORT m045_retrigger_user_wd(byte micro_slot,
byte channel);

Funktion Mit der Prozedur **m045_init** wird der User-Watchdog auf den Wert von *user_wd_val* gesetzt. Der User-Watchdog sorgt dafür, daß beim Ausfall der angeschlossenen CPU (der MODULAR-4/486 Basiskarte) nach einer definierten Anzahl von Datentelegrammen der Datenzyklus verlassen wird. Die Basiskarte bzw. der Anwender muß den User-Watchdog mit **m045_reset_user_wd** fortlaufend nachtriggern.

Parameter *channel*: definiert den Kanal, dessen User-Watchdog nachgetriggert werden soll.

user_wd_val:
definiert die Anzahl von Datentelegrammen, nach denen der Datenzyklus verlassen wird.

m045_get_status Watchdog- und DPS-Status lesen

Pascal PROCEDURE m045_get_status(micro_slot, channel: byte;
var user_wd_state, user_dps_state: byte);

C void EXPORT m045_get_status(byte micro_slot, byte channel,
byte *user_wd_state, byte *user_dps_state);

Funktion Der Zustand der Watchdog-State-Machine kann mit der Prozedur m045_get_status abgefragt werden. Außerdem kann ermittelt werden, ob die DP-State-Machine in den Data-Exchange Zustand eingetreten ist oder diesen verlassen hat. Die Ursache dafür kann z.B. ein fehlerhaftes Parametriertelegramm in der Datentransfer-Phase gewesen sein.

Parameter *channel*: definiert den Kanal, dessen Statusinformationen gelesen werden sollen.

user_wd_state:
liefert den Status der Watchdog-State-Machine:

Wert	Konstantenbezeichner	Bedeutung
0x00	SPC3_WD_STATE_BAUD_SEARCH	Baudratensuche
0x01	SPC3_WD_STATE_BAUD_CONTROL	Überprüfung der Baudrate
0x02	SPC3_WD_STATE_DP_MODE	DP-Mode, d.h. Bus-Watchdog aktiviert

user_dps_state:
liefert den Status der DP-State-Machine:

Wert	Konstantenbezeichner	Bedeutung
0x00	DPS2_DP_STATE_WAIT_PRM	Warte auf Parametrierung
0x01	DPS2_DP_STATE_WAIT_CFG	Warte auf Konfigurierung
0x02	DPS2_DP_STATE_DATE_EX	Datenaustausch
0x03	DPS2_DP_STATE_ERROR	Fehler

m045_read_data Eingangsdaten lesen

Pascal PROCEDURE m045_read_data(micro_slot, channel: byte; var count: word; var flag: byte; var i_data);

C void EXPORT m045_read_data(byte micro_slot, byte channel, ushort *count, byte *flag, byte *i_data);

Funktion Eingangsdaten (vom Master) werden (falls vorhanden) aus dem SPC3 gelesen und in das Array **i_data** geschrieben.

Parameter
- *channel*: definiert den Kanal, dessen Daten gelesen werden sollen.
- *count*: Zeiger, gibt die Anzahl der zu lesenden Daten an.
- *flag*: liefert zurück, ob neue Daten vorhanden sind (= 1) oder nicht (= 0).
- *i_data*: Adresse des Datenbereichs (Array), in den die Daten geschrieben werden sollen.

m045_read_data_direct Eingangsdaten direkt lesen

Pascal PROCEDURE m045_read_data_direct(micro_slot, channel: byte; var count: word; var i_data; offset: byte);

C void EXPORT m045_read_data_direct(byte micro_slot, byte channel, ushort *count, byte *i_data, byte offset);

Funktion Eingangsdaten (vom Master) werden (falls vorhanden) direkt aus dem SPC3 gelesen, und zwar ab Beginn des Datenbereichs im SPC3 zzgl. Offset, und in das Array **i_data** geschrieben. Der Pointer muß mit **m045_set_read_pointer** gesetzt werden.

Parameter
- *channel*: definiert den Kanal, dessen Daten gelesen werden sollen.
- *count*: Zeiger, gibt die Anzahl der zu lesenden Daten an.
- *i_data*: Adresse des Datenbereichs (Array), in den die Daten geschrieben werden sollen.
- *offset:* Offset zum Beginn des Datenbereichs im SPC3

m045_set_read_pointer Read-Pointer im SPC3 setzen

Pascal PROCEDURE m045_set_read_pointer(micro_slot, channel: byte);

C void EXPORT m045_set_read_pointer(byte micro_slot, byte channel);

Funktion Diese Prozedur setzt einen neuen Pointer für das Lesen von Daten aus dem SPC3.

Parameter *channel*: gibt den Kanal an, dessen Read_pointer gesetzt wird.

m045_write_data Ausgangsdaten schreiben

Pascal PROCEDURE m045_write_data(micro_slot, channel: byte; count: word; var o_data);

C void EXPORT m045_write_data(byte micro_slot, byte channel, ushort count, byte *o_data);

Funktion Ausgangsdaten (die im Array **o_data** stehen) werden zum Master gesendet.

Parameter *channel*: definiert den Kanal, dessen Daten zum Master gesendet werden sollen.

count: gibt die Anzahl der Daten an, die übertragen werden sollen.

o_data: Adresse des Datenbereichs (Array), in dem die Daten stehen.

m045_write_data_direct Ausgangsdaten direkt schreiben

Pascal PROCEDURE m045_write_data_direct(micro_slot, channel: byte; count: word; var o_data; offset: byte);

C void EXPORT m045_write_data_direct(byte micro_slot, byte channel,

ushort count, byte *o_data, byte offset);

Funktion Ausgangsdaten (die im Array **o_data** stehen) werden direkt in den SPC3 geschrieben, und zwar ab Beginn des Datenbereichs im SPC3 zzgl. Offset. Durch Aufruf der Prozedur m045_get_write_pointer werden die Daten zum Master gesendet und ein neuer Pointer gelesen.

Parameter *channel*: definiert den Kanal, dessen Daten zum Master gesendet werden sollen.

count: gibt die Anzahl der Daten an, die übertragen werden sollen.

o_data: Adresse des Datenbereichs (Array), in dem die Daten stehen.

m045_get_write_pointer Write-Pointer aus SPC3 lesen

Pascal PROCEDURE m045_get_write_pointer(micro_slot, channel: byte);

C void EXPORT m045_get_write_pointer(byte micro_slot, byte channel);

Funktion Diese Prozedur veranlaßt das Senden von Daten zum Master und liest einen neuen Pointer für das Schreiben von Daten in den SPC3.

Parameter *channel*: gibt den Kanal an, dessen Write-Pointer gelesen wird.

m045_write_diag Schreiben von Diagnosedaten

Pascal PROCEDURE m045_write_diag(micro_slot, channel: byte;
var user_diag_flag: byte; diag_service, diag_len: byte;var new_diag);

C void EXPORT m045_write_diag(byte micro_slot, byte channel,
byte *user_diag_flag, byte diag_service, byte diag_len,
diag_data_blk *new_diag);

Funktion Diese Prozedur sendet Diagnosedaten zum Master.

Parameter *channel*: definiert den Kanal, dessen Diagnosedaten gesendet werden sollen.

user_diag_flag:
liefert den Zustand des internen Diagnoseflags zurück: = TRUE: Diagnosedaten vom Master abgeholt.

diag_service:
definiert, welche Art von Diagnosedaten gesendet werden sollen:

Bit	Bezeichnung	Bedeutung
0	EXT_DIAG	Dieses Bit zeigt an, ob erweiterte Diagnosedaten oder Statusmeldungen vorliegen. Die Länge wird mit *diag_len* übergeben. Die Diagnosedaten der Anwendung beginnen mit dem ersten Headerbyte ab Byte 6 und müssen wie in der PROFIBUS DP Norm beschrieben vorliegen, d.h. eine Blockstruktur mit einem Headerbyte je Block, in dem der Blocktyp (geräte-, kennungs- oder kanalbezogene Diagnose) und die Länge des nachfolgenden Blocks definiert sind. Wird die Länge 6 mit übergeben, dann wird dieses Bit von der Software/Bibliothek fest auf 0 gesetzt.
1	STAT_DIAG	Ist dieses Bit 1, dann wird das Bit 1 von Byte 0 der Diagnosedaten (Stat Diag) gesetzt, im anderen Fall wird das Bit zurückgesetzt.
2	EXT_DIAG_OVF	Wenn dieses Bit gleich 1 ist, dann wird das Bit 2 von Byte 0 der Diagnosedaten (Ext_Diag_Overf) gesetzt, im anderen Fall wird das Bit zurückgesetzt.

diag_len: definiert die Anzahl der zu sendenden Diagnosedaten.

new_diag: Adresse des Datenbereichs, in dem die Diagnosedaten stehen. Der Datenbereich hat folgende Struktur:

Feldbezeichner	**Typ**	**Bedeutung**
stationstatus_1	Byte	Byte 0 bis 5: fester Diagnose-Header
stationstatus_2	Byte	
stationstatus_3	Byte	
master_add	Byte	
ident_high	Byte	
ident_low	Byte	
header	Byte	Anzahl der externen Diagnosedaten
ext_diag	Byte-Array	Externe Diagnosedaten

m045_error_function Fehlerbehandlung

Pascal PROCEDURE m045_error_function(micro_slot, channel: byte; error: integer);

C void EXPORT m045_error_function(byte micro_slot, byte channel, short error);

Funktion Sie können eine eigene Fehlerbehandlung schreiben. Hierzu schreiben Sie lediglich eine Prozedur **m045_error_function**, die beim Auftreten eines Fehlers automatisch aufgerufen wird.

Parameter *channel*: definiert den Kanal, der eine Fehlermeldung liefert.

error: liefert eine Fehlermeldung zurück, mögliche Werte:

Wert	Konstantenbezeichner	Bedeutung
0x0000	SPC3_INIT_OK	Pufferlänge OK, SPC3 fehlerfrei initialisiert.
0x0001	SPC3_INITF_LESS_-MEM	Zuviel Speicher verbraucht, es konnten nicht alle Puffer angelegt werden.
0x0002	SPC3_INITF_NOFF	Der SPC3 befindet sich nicht im Offline-Zustand.
0x0003	DPS2_INITF_DIN_ DOUT_LEN	Fehler bei Din-Dout-Länge.
0x0004	DPS2_INITF_DIAG_ LEN	Fehler bei Diagnoselänge.
0x0005	DPS2_INITF_PRM_LEN	Fehler bei Parametrierdatenlänge.
0x0007	DPS2_INITF_SSA_LEN	Fehler bei Adreßdatenlänge.
0x0014	IO_LENGTH_ERROR	Fehler bei I/O-Länge
0x0015	SPC3_READ_ERROR	Fehler beim Lesen von Daten aus dem SPC3 (Anzahl oder Offset falsch)
0x0016	SPC3_WRITE_ERROR	Fehler beim Schreiben von Daten in den SPC3 (Anzahl oder Offset falsch)

Behandlung der Indications (Interrupts)

Auftretende Indications müssen abgearbeitet und bestätigt werden. Sie können die Bearbeitung erweitern, indem Sie die folgenden Prozeduren einbinden (diese sind von Ihnen zu erstellen).

m045_offline OFFLINE

Pascal PROCEDURE m045_offline(micro_slot, channel: byte);

C void EXPORT m045_offline(byte micro_slot, byte channel);

Funktion Reaktion auf die Indication, daß der SPC3 in den Offline-Zustand gekommen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

m045_baudrate_detect BAUDRATE_DETECT

Pascal PROCEDURE m045_baudrate_detect(micro_slot, channel, baudrate: byte);

C void EXPORT m045_baudrate_detect(byte micro_slot, byte channel, byte baudrate);

Funktion Reaktion auf die Indication, das eine Baudrate erkannt wurde.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

baudrate: gibt die vom SPC3 erkannte Baudrate zurück:

Wert	Konstantenbezeichner	Bedeutung
0x00	BD_12M	12 MBaud
0x01	BD_6M	6 MBaud
0x02	BD_3M	3 MBaud
0x03	BD_1_5M	1,5 MBaud
0x04	BD_500k	500 kBaud
0x05	BD_187_5k	187,5 kBaud
0x06	BD_93_75k	93,75 kBaud
0x07	BD_45_45k	45,45 kBaud
0x08	BD_19_2k	19,2 kBaud
0x09	BD_9_6k	9,6 kBaud

m045_user_timer_clock USER_TIMER_CLOCK

Pascal PROCEDURE m045_user_timer_clock(micro_slot, channel: byte);

C void EXPORT m045_user_timer_clock(byte micro_slot, byte channel);

Funktion Reaktion auf die Indication, daß die Zeitbasis des User_Timer_Clocks abgelaufen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

m045_go_leave_data_ex GO_LEAVE_DATA_EX

Pascal PROCEDURE m045_go_leave_data_ex(micro_slot, channel, user_dps_state: byte);

C void EXPORT m045_go_leave_data_ex(byte micro_slot, byte channel, byte user_dps_state);

Funktion Reaktion auf die Indication, daß der SPC3 in den Zustand ´DATA_EX´ eingetreten ist oder ihn verlassen hat.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

user_dps_state:

Zustand der DP-State-Machine:

Wert	Konstantenbezeichner	Bedeutung
0x00	DPS2_DP_STATE_WAIT_PRM	Warte auf Parametrierung
0x01	DPS2_DP_STATE_WAIT_CFG	Warte auf Konfigurierung
0x02	DPS2_DP_STATE_DATE_EX	Datenaustausch
0x03	DPS2_DP_STATE_ERROR	Fehler

m045_wd_timeout WD_TIMEOUT

Pascal PROCEDURE m045_wd_timeout(micro_slot, channel, user_wd_state: byte);

C void EXPORT m045_wd_timeout(byte micro_slot, byte channel, byte user_wd_state);

Funktion Reaktion auf die Indication, daß der Watchdog-Timer abgelaufen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

user_wd_state:
Zustand der Watchdog-State-Machine:

Wert	Konstantenbezeichner	Bedeutung
0x00	SPC3_WD_STATE_BAUD_SEARCH	Baudraten suche
0x01	SPC3_WD_STATE_BAUD_CONTROL	Überprüfung der Baudrate
0x02	SPC3_WD_STATE_DP_MODE	DP-Mode, d.h. Bus-Watchdog aktiviert

m045_global_ctrl_command GLOBAL_CTRL_COMMAND

Pascal PROCEDURE m045_global_ctrl_command(micro_slot, channel, user_global_ctrl_command: byte);

C void EXPORT m045_global_ctrl_command(byte micro_slot, byte channel, byte user_global_ctrl_command);

Funktion Reaktion auf die Indication, daß ein ´Global_Control-Telegramm´ eingetroffen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

user_global_ctrl_command:
liefert das empfangene Kommando:

Bit	Konstanten-bezeichner	Bedeutung
0	Reserviert	Für zukünftige Erweiterungen
1	CLEAR_DATA	Mit diesem Kommando werden die Output-Daten gelöscht und dem Anwender zur Verfügung gestellt.
2	UNFREEZE	Mit 'Unfreeze' wird das Einfrieren der Input-Daten aufgehoben.
3	FREEZE	Die Input-Daten werden 'eingefroren'. Neue Input-Daten werden erst wieder von der Anwendung geholt, wenn der Master das nächste 'Freeze' Kommando sendet.
4	UNSYNC	Das Kommando Unsync hebt das 'Sync' Kommando auf.
5	SYNC	Die letzten empfangenen Output-Daten werden der Anwendung zur Verfügung gestellt. Die nachfolgend übertragenen Output-Daten werden solange nicht zur Anwendung weitergereicht, bis das nächste 'Sync'-Kommando erfolgt.
6,7	Reserviert	Für zukünftige Erweiterungen

m045_new_ssa_data NEW_SSA_DATA

Pascal PROCEDURE m045_address_data(micro_slot, channel, this_station, change_station_number: byte);

C void EXPORT m045_address_data(byte micro_slot, byte channel, byte this_station, byte change_station_number);

Funktion Reaktion auf die Indication, daß ein ´Set_Slave_Address-Telegramm´ eingetroffen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

this_station:
gibt die neue Adresse an.

change_station_number:
Flag, das angibt, ob Adresse geändert werden darf.

m045_check_config NEW_CFG_DATA

Pascal FUNCTION m045_check_config(micro_slot, channel: byte; var cfg_parameter: byte; cfg_data_len, cfg_result: byte): byte;

C byte EXPORT m045_check_config(byte micro_slot, byte channel, byte *cfg_parameter, byte cfg_data_len, byte cfg_result);

Funktion Reaktion auf die Indication, daß eine ´Check_Cfg-Telegramm´ eingetroffen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

cfg_parameter:
enthält die empfangenen Konfigurationsdaten

cfg_data_len:
enthält die Anzahl der empfangenen Konfigurationsdaten.

cfg_result: Ergebnis der letzten Konfiguration.

Rückgabewert:

Wert	Konstantenbezeichner	Bedeutung
0x00	DPS2_CFG_OK	Konfiguration ist O.K.
0x01	DPS2_CFG_FAULT	Falsche Konfiguration.
0x02	DPS2_CFG_UPDATE	Konfiguration muß aktualisiert werden.

m045_check_parameter NEW_PRM_DATA

Pascal FUNCTION m045_check_parameter(micro_slot, channel: byte, var prm_parameter; param_data_len, prm_result: byte): byte;

C byte EXPORT m045_check_parameter(byte micro_slot, byte channel, byte *prm_parameter, byte param_data_len, byte prm_result);

Funktion Reaktion auf die Indication, daß ein ´Set_Param-Telegramm´ eingetroffen ist.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

prm_parameter:
enthält die empfangenen Parameterdaten

prm_data_len:
enthält die Anzahl der empfangenen Parameterdaten.

prm_result: Ergebnis der letzten Parametrierung.

Wert	Konstantenbezeichner	Bedeutung
0x00	DPS2_PRM_FINISHED	Keine weiteren Parametriertelegramme, Ende der Sequenz.
0x01	DPS2_PRM_CONFLICT	Es liegt ein weiteres Parametriertelegramm vor, es muß eine nochmalige Überprüfung der Parametrierung erfolgen.
0x03	DPS2_PRM_NOT_ ALLOWED	Zugriff im derzeitigen Buszustand nicht erlaubt. Es könnte z.B. während der Überprüfung der Watchdog abgelaufen sein. Die Überprüfung der Parametrierung ist zu verwerfen.

Rückgabewert: TRUE (= 1), wenn Parametrierung O.K., oder FALSE (= 0), wenn Parametrierung nicht O.K.

m045_diag_buffer_changed DIAG_BUFFER_CHANGED

Pascal PROCEDURE m045_diag_buffer_changed(micro_slot, channel: byte);

C void EXPORT m045_diag_buffer_changed(byte micro_slot, byte channel);

Funktion Reaktion auf die Indication, das der Diagnosepuffer vom Master abgeholt wurde.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

m045_data_in_out DATA_IN_OUT

Pascal PROCEDURE m045_data_in_out (micro_slot, channel: byte; var out_len: word; var i_data);

C void EXPORT m045_data_in_out(byte micro_slot, byte channel, ushort *out_len, byte *i_data);

Funktion Reaktion auf die Indication, daß ein ´Write_Read_Data-Telegramm´ empfangen wurde.

Parameter *channel*: gibt den Kanal an, der von der Indication betroffen ist.

out_len: Anzahl der Daten die empfangen wurden.

i_data: Adresse des Datenbereichs, in den die Daten geschrieben werden sollen.

Programmierung mit I/O-Zugriffen

Dieses Kapitel ist für jene Anwender gedacht, die eigene Anwendungsprogramme für die Basiskarte MODULAR-4/486 schreiben wollen.

Lokale I/O-Adressen

Alle Adressen sind in hexadezimaler Schreibweise angegeben.

Kanal 1 (Slave Controller SPC3/1):

Adresse	Zugriff[1]	Funktion
MBA+00h	RW16	Adreß-Pointer 1 (AP1) lesen/schreiben (0000h bis 05ffh)
MBA+01h	RW8	Interrupt-Select-Register 1 (IR1) setzen/lesen Bit 2, 1, und 0 bestimmen die Interrupt-Leitung zur Basiskarte: 0 0 0 keine Interrupt-Leitung angewählt 0 0 1 IRQ-A 0 1 0 IRQ-B 0 1 1 IRQ-C 1 0 0 IRQ-D 1 0 1 IRQ-E 1 1 0 IRQ-F 1 1 1 NMI Zurücklesen des Registerinhaltes liefert zusätzlich die aktuellen Werte der Slave-Interrupt Leitungen _XINT1 und _XINT2 (Bit 3 und 4).

[1] R16 : 16 Bit Lesezugriff, W16 : 16 Bit Schreibzugriff, RW16 : Sowohl 16 Bit Lese- als auch Schreibzugriffe möglich. R8 : 8 Bit Lesezugriff, W8 : 8 Bit Schreibzugriff, RW8 : Sowohl 8 Bit Lese- als auch Schreibzugriffe möglich.

Adresse	**Zugriff**[1]	**Funktion**
MBA+02h	RW8	LED Register 1 (LR1) lesen/schreiben Bit 0 selektiert, ob die Leuchtdiode vom Slave-Baustein 1 angesteuert wird oder entsprechend Bit 1 (0 = ein, 1 = aus). Beim Lesen werden die Bits 0 und 1 entsprechend zurückgeliefert. Bit 2 gibt den aktuellen Zustand von LED 1 an (0 = ein, 1 = aus).
MBA+03h	R8	Gate-Array-Version lesen Bit 0 bis 3 = Revision, Bit 4 bis 7 = Version z.B.: 0010 0101 = 25h = Version 2, Revision 5
MBA+04h	RW8	Slave 1 lesen/schreiben
MBA+05h	RW8	Slave 1 lesen/schreiben, anschl. inkrementieren von AP1 um 1
MBA+06h	RW8	Reset Kanal 1 (Slave-Controller 1 (SPC3/1), IR1 und LR1) Beim Lesen enthält Bit 0 den Status der Reset-Leitung von Slave 1, Bit 1 den Status der Reset-Leitung von Slave 2.
MBA+07h	RW8	Modul-Reset durchführen (inkl. SPC3/1 und SPC3/2) Beim Lesen enthält Bit 0 den Status der Reset-Leitung von Slave 1, Bit 1 den Status der Reset-Leitung von Slave 2.

Kanal 2 (Slave Controller SPC3/2):

Adresse	Zugriff[1]	Funktion
MBA+08h	RW16	Adreß-Pointer 2 (AP2) lesen/schreiben (0000h bis 05ffh)
MBA+09h	RW8	Interrupt-Select-Register 2 (IR2) setzen/lesen Bit 2, 1, und 0 bestimmen die Interrupt-Leitung zur Basiskarte: 0 0 0 keine Interrupt-Leitung angewählt 0 0 1 IRQ-A 0 1 0 IRQ-B 0 1 1 IRQ-C 1 0 0 IRQ-D 1 0 1 IRQ-E 1 1 0 IRQ-F 1 1 1 NMI Zurücklesen des Registerinhaltes liefert zusätzlich die aktuellen Werte der Slave-Interrupt Leitungen _XINT2 und _XINT1 (Bit 3 und 4).
MBA+0ah	RW8	LED Register 2 (LR2) lesen/schreiben Bit 0 selektiert, ob die Leuchtdiode vom Slave-Baustein 1 angesteuert wird oder entsprechend Bit 1 (0 = ein, 1 = aus). Beim Lesen werden die Bits 0 und 1 entsprechend zurückgeliefert. Bit 2 gibt den aktuellen Zustand von LED 2 an (0 = ein, 1 = aus).
MBA+0bh	R8	Gate-Array-Version lesen Bit 0 bis 3 = Revision, Bit 4 bis 7 = Version z.B.: 0010 0101 = 25h = Version 2, Revision 5

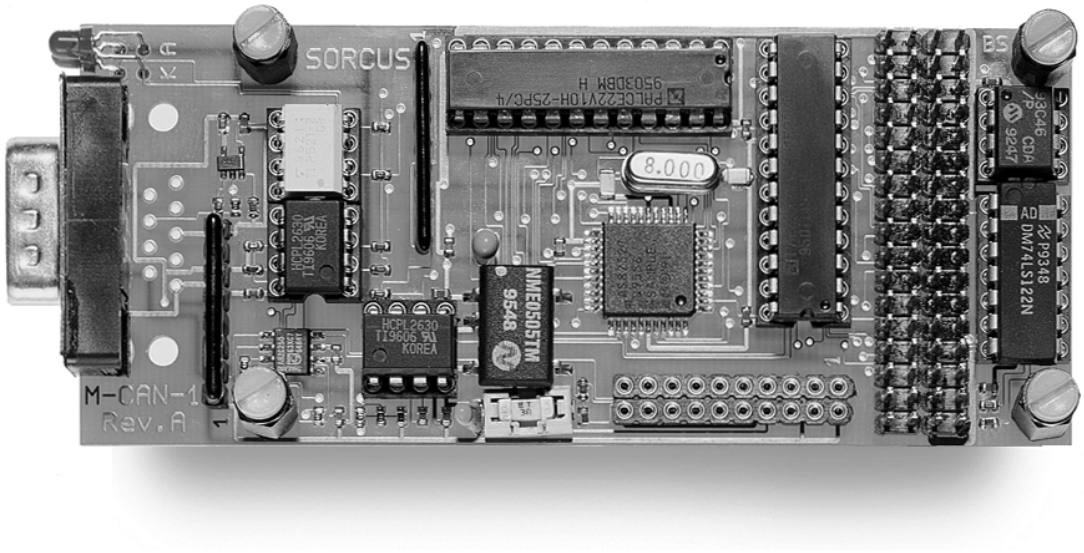
[1] R16 : 16 Bit Lesezugriff, W16 : 16 Bit Schreibzugriff, RW16 : Sowohl 16 Bit Lese- als auch Schreibzugriffe möglich. R8 : 8 Bit Lesezugriff, W8 : 8 Bit Schreibzugriff, RW8 : Sowohl 8 Bit Lese- als auch Schreibzugriffe möglich.

Adresse	**Zugriff**[1]	**Funktion**
MBA+0ch	RW8	Slave 2 lesen/schreiben
MBA+0dh	RW8	Slave 2 lesen/schreiben, anschl. inkrementieren von AP2 um 1
MBA+0eh	RW8	Reset Kanal 2 (Slave-Controller SPC3/2, IR2 und LR2) Beim Lesen enthält Bit 0 den Status der Reset-Leitung von Slave 2, Bit 1 den Status der Reset-Leitung von Slave 1.
MBA+0fh	RW8	Modul-Reset durchführen (inkl. SPC3/1 und SPC3/2) Beim Lesen enthält Bit 0 den Status der Reset-Leitung von Slave 2, Bit 1 den Status der Reset-Leitung von Slave 1.

Notizen

9. M-CAN-1

CAN-Bus-Interface



Funktionsbeschreibung

Das SP-Bus-Modul M-CAN-1 dient zur Ankopplung des MODULAR-4 Systems an ein CAN-Bus-Netzwerk. Es besitzt folgende Funktionseinheiten und Eigenschaften:

- CAN-Bus-Controller Intel 82527 für die unteren beiden Schichten des CAN-Protokolls. Dieser stellt die Dienste 'Senden von Daten' (Data-Frame), 'Anfordern von Daten' (Remote-Frame) und 'Empfangen von Daten' zur Verfügung.
- Galvanisch getrennte physikalische Schnittstelle zum CAN-Bus entsprechend ISO/DIS 11898 Standard
- Unterstützt CAN-Spezifikationen 2.0 A und 2.0 B (11- und 29-Bit-Identifier)
- Übertragungsgeschwindigkeiten von 5 kBit/s bis 1 MBit/s einstellbar
- Bit-Timing-Parameter frei konfigurierbar
- Automatische Fehlererkennung durch den CAN-Controller on-board
- Full-CAN-Funktionalität, d.h. die MODULAR-4 Karte muß nur auf die Busnachrichten reagieren, die für die jeweilige Anwendung von Interesse sind
- 2 Digitalausgänge (galvanisch getrennte potentialfreie Open-Collector-Ausgänge, frei programmierbar), parallel dazu für Kontrollzwecke 2 LEDs auf dem Modul
- Galvanisch getrennte +5V stehen über den Anschlußstecker auch anderen Busteilnehmern zur Verfügung
- CAN-Bus-Abschlußwiderstand per Software zuschaltbar
- 9-poliger D-SUB-Stecker nach CiA Standard 102, V2.0 auf dem Modul
- Flankensteilheit per steckbarem Widerstandsnetzwerk einstellbar
- EEPROM zur Speicherung von Initialisierungs- und Konfigurationsdaten
- Ansprechen des Moduls über mitgeliefertes Treiberprogramm

Der CAN-Bus

Ein CAN-Bus-Netzwerk ist ein Bussystem, bei dem mehrere Teilnehmer den Bus gleichzeitig anfordern können. Ein Teilnehmer sendet eine Nachricht, alle anderen Teilnehmer hören diese Nachricht mit und entscheiden dann selbst, ob diese für sie von Interesse ist oder nicht. Eine gesendete Nachricht kann sowohl Daten an andere Teilnehmer verschicken als auch Daten eines anderen Teilnehmers anfordern.

Die Daten werden hierbei immer zusammen mit einem CAN-Identifier, der im folgenden als ID bezeichnet wird, auf dem CAN-Bus verschickt. Die ID wird von anderen Teilnehmern als Kriterium verwendet, ob die Nachricht für sie von Interesse ist oder nicht. Weiterhin legt der Wert der ID die Priorität der Nachricht auf dem CAN-Bus fest. Wenn mehrere Nachrichten gleichzeitig gesendet werden, setzt sich im zeitlichen Verlauf die Nachricht mit der kleinsten ID gegenüber den anderen Nachrichten durch.

Alle erforderlichen Dienste zur Kommunikation über den CAN-Bus mit dem Modul M-CAN-1 werden durch das Treiberprogramm M049TASK zur Verfügung gestellt. Das Ansprechen dieses Treiberprogrammes erfolgt auf Hochsprachenebene mittels geeigneter Bibliotheksfunktionen.

Message-Objekte

Eine über den CAN-Bus verschickte Nachricht besteht immer aus maximal 8 Byte Nutzdaten und der diesen Daten zugeordneten ID. Die Nutzdaten und die ID werden zu einem Message-Objekt zusammengefaßt und vom Treiberprogramm M049TASK verwaltet.

Alle benötigten Message-Objekte müssen mit **m049_config_msg_object** konfiguriert werden. Diese Funktion vergibt für das konfigurierte Message-Objekt eine eindeutige Objektnummer und übergibt sie Ihrem Anwendungsprogramm als *Handle* für das Message-Objekt. Dieses Handle müssen Sie allen Funktionen, die sich auf dieses konfigurierte Message-Objekt beziehen, als Funktionsparameter übergeben.

Bei der Konfiguration eines Message-Objekts ordnen Sie diesem eine eindeutige ID zu. Hierbei sollten Sie folgendes beachten:

- Die ID muß für einen Busteilnehmer eindeutig sein, d.h. er darf sie höchstens einem Nutzdatensatz zuordnen. Die ID kann jedoch von beliebig vielen Busteilnehmern verwendet werden.
- Eine Nachricht darf nur einem Teilnehmer als aktiver Sender zugeordnet sein, d.h. nur dieser eine Teilnehmer darf die der ID zugeordneten Nutzdaten auf dem Bus versenden. Es können mehrere Busteilnehmer als Empfänger der Daten auftreten.
- Das Modul M-CAN-1 unterstützt die CAN-Spezifikationen 2.0 A und 2.0 B. Es kann somit sowohl 11-Bit-Identifier als auch 29-Bit-Identifier verarbeiten.
- Nicht alle auf dem Markt erhältlichen CAN-Controller unterstützen 29-Bit-Identifier! Befinden sich solche Busteilnehmer im CAN-Netzwerk, wird es bei der Verwendung von 29-Bit-Identifiern zu Fehlern kommen. Verwenden Sie in diesem Fall 11-Bit-Identifier.
- Alle Message-Objekte müssen dasselbe Identifier-Format verwenden. Es wird bei der Konfiguration des Moduls festlegt (siehe EEPROM-Wort 9).

Ein als ***Sendeobjekt*** konfiguriertes Message-Objekt (durch Setzen des Flags *_M049_SEND*) kann als *aktives* (durch Setzen des Flags *_M049_ACTIVE*) oder/und als *passives* (durch Setzen des Flags *_M049_PASSIVE*) Sendeobjekt konfiguriert werden:

- Bei einem aktiven Sendeobjekt geben Sie den Zeitpunkt, wann das Message-Objekt Daten sendet, in Ihrem Anwendungsprogramm durch Aufruf der Funktion **m049_send_data** selbst vor. In diesem Falle wird ein *Data-Frame* auf dem CAN-Bus verschickt.
- Bei einem passiven Sendeobjekt wird der Zeitpunkt, wann das Message-Objekt Daten sendet, von einem anderen Busteilnehmer, der von Ihrem Anwendungsprogramm Daten per *Remote-Frame* anfordert, bestimmt. Das Treiberprogramm M049TASK sendet *automatisch* die zu diesem Zeitpunkt gültigen Daten des passiven Sendeobjekts durch Verschicken eines Data-Frames.
- Die Übergabe der zu sendenden Daten an das Message-Objekt erfolgt mit der Funktion **m049_set_data**.

Ein als ***Empfangsobjekt*** konfiguriertes Message-Objekt (durch Nicht-Setzen des Flags *_M049_SEND*) kann als *aktives* (durch Setzen des Flags *_M049_ACTIVE*) oder/und als *passives* (durch Setzen des Flags *_M049_PASSIVE*) Empfangsobjekt konfiguriert werden:

- Bei einem aktiven Empfangsobjekt geben Sie den Zeitpunkt, wann das Message-Objekt Daten empfängt, in Ihrem Anwendungsprogramm durch Aufruf der Funktion **m049_send_data_request** selbst vor. In diesem Falle werden Daten durch Senden eines *Remote-Frames* von einem anderen Busteilnehmer angefordert.
- Ein passives Empfangsobjekt empfängt Daten, ohne daß Sie diese in ihrem Anwendungsprogramm speziell anfordern. Die Daten stammen von einem anderen Busteilnehmer, der einen Data-Frame verschickt hat.
- Die Übernahme der empfangenen Daten vom Empfangsobjekt erfolgt mit der Funktion **m049_get_data**.

Die zu setzenden Flags sind durch bitweise OR-Verknüpfung beliebig miteinander kombinierbar. So kann z.B. durch Setzen der Flags *_M049_SEND*, *_M049_ACTIVE* und *_M049_PASSIVE* ein Sendeobjekt konfiguriert werden, das sowohl passiv (d.h. auf Anforderung eines anderen Busteilnehmers) als auch aktiv (durch Aufruf von **m049_send_data** in Ihrem Anwendungsprogramm) Daten senden kann.

Interruptgesteuerte Kommunikation

Das Modul M-CAN-1 löst beim Auftreten bestimmter Ereignisse einen Interrupt (möglich sind INT-A bis INT-F oder der NMI) auf der MODULAR-4 Karte aus. Dieser Interrupt wird durch das entsprechend installierte Treiberprogramm M049TASK (DI-Task) bedient. Die Interrupt-Nummer, unter der das Treiberprogramm installiert wird (siehe Seite 9-25), bestimmt den durch das Modul M-CAN-1 ausgelösten Interrupt.

Die Interrupt-Serviceroutine des Treiberprogrammes wertet das aufgetretene Ereignis aus und ruft in einem der folgenden Fälle eine Serviceroutine, die eine Echtzeitfunktion sein muß, auf:

- Ein Message-Objekt hat entweder Daten per Data-Frame erfolgreich gesendet oder Daten per Remote-Frame erfolgreich empfangen. Die Nachricht ist von mindestens einem anderen Busteilnehmer richtig empfangen und von keinem Busteilnehmer als fehlerhaft erkannt worden. Für diesen Fall müssen Sie eine **Message-Serviceroutine** (siehe unten) schreiben.

- Ein passives Message-Objekt, das als Empfangsobjekt konfiguriert ist, hat eine Nachricht empfangen. Für diesen Fall müssen Sie eine **Message-Serviceroutine** (siehe unten) schreiben.
- Die automatische Fehlerbehandlung des CAN-Controllers auf dem Modul erkennt einen stark gestörten Bus. Für diesen Fall müssen Sie eine **Error-Serviceroutine** (siehe unten) schreiben.

Die Serviceroutinen können in jeder beliebigen Task der MODULAR-4 Karte stehen. Möchten Sie die Echtzeitprogrammierung umgehen, bietet Ihnen das Treiberprogramm eine universelle Serviceroutine (Funktion 14 des Treiberprogrammes) an, die einen Service-Request an den PC sendet. Hinweise dazu finden Sie auf Seite 9-37. Sie müssen sich dann um die Behandlung des Service-Requests innerhalb eines Programmes kümmern. Für zeitkritische Aufgaben wird jedoch in jedem Fall die Programmierung eines Echtzeitprogrammes empfohlen.

Message-Serviceroutine

Sie können für jedes Message-Objekt individuell entscheiden, wie Sie auf den Interrupt reagieren möchten. Hierzu weisen Sie bei der Konfiguration des Message-Objekts (mit **m049_config_msg_object**) die Task- und Funktionsnummer einer Funktion zu, die vom Treiberprogramm aufgerufen wird. Sie können auch für verschiedene Message-Objekte dieselbe Message-Serviceroutine benutzen.

Eine Message-Serviceroutine muß folgenden Aufbau haben:

```
void far pascal message_service
    (ushort task, ushort insize, M049Handle *handle,
    ushort outsize, ulong *outdata)
{
   ml8rt_entry_function();        // save registers and prepare

   /* TODO */                     // user code

   ml8rt_exit_function(0,0,0);    // restore registers
}
```

Die Funktion erhält in *handle* das Handle für das Message-Objekt. Die anderen Parameter der Funktion sind ohne Bedeutung. Die Funktion darf jedoch keine Daten zurückgeben!

Error-Serviceroutine

Die MODULAR-4 Karte wird durch den CAN-Controller auf dem Modul weitgehend von der Behandlung von Übertragungsfehlern entlastet. Der Controller übernimmt die gesamte Fehlererkennung und -begrenzung (Bit-Monitoring, Frame- und CRC-Check, Acknowledgement-Überwachung, Überwachung der Bit-Stuffing-Codierungsregel, wiederholtes Senden im Fehlerfall).

Der CAN-Controller besitzt hierzu einen Sende- und einen Empfangsfehlerzähler, die er beim Auftreten von Fehlern um einen vom Fehlertyp abhängigen Betrag inkrementiert. Nach jeder erfolgreichen Übertragung wird der Zähler um einen bestimmten Betrag dekrementiert.

In Abhängigkeit dieser beiden Zähler nimmt der Controller einen der drei folgenden Zustände ein:

- Beide Zähler < 128 (Betriebsart 'Fehleraktiv'): Das Modul nimmt voll an der Bus-Kommunikation teil und sendet bei der Erkennung von Fehlern automatisch einen aktiven Error-Frame. Die als fehlerhaft erkannte Nachricht wird hierbei durch gezieltes Verletzen des Nachrichtenprotokolls zerstört.
- Einer der beiden Zähler überschreitet den Wert 128 (Betriebsart 'Fehlerpassiv'): Das Modul nimmt weiter voll an der Bus-Kommunikation teil, sendet nun aber bei der Erkennung von Fehlern einen passiven Error-Frame, der das Nachrichtenprotokoll nicht zerstört.
- Einer der beiden Zähler überschreitet den Wert 256 (Betriebsart 'Bus Off'): Das Modul wird von der Bus-Kommunikation abgeschaltet. Diese Betriebsart kann nur durch die Funktion **m049_stop_com** und anschließend **m049_start_com** verlassen werden. In diesem Falle werden beide Fehlerzähler wieder zurückgesetzt.

Das Modul schaltet zwischen den Betriebsarten 'Fehleraktiv' und 'Fehlerpassiv' selbständig hin und her.

Ein Fehlerstand>96 wird als 'Stark gestörter Bus' interpretiert und genau wie der Übergang in die Betriebsart 'Bus-Off' mit einem Interrupt signalisiert. Für diesen Fall müssen Sie eine Error-Serviceroutine schreiben. Hierzu weisen Sie bei der Konfiguration des Moduls (mit **m049_config_module**) die Task- und Funktionsnummer einer Funktion zu, die vom Treiberprogramm M049TASK aufgerufen wird.

Die Error-Serviceroutine muß folgenden Aufbau haben:

```
void far pascal error_service
    (ushort task, ushort insize, ulong *status,
    ushort outsize, byte *outdata)
{
   ml8rt_entry_function();         // save registers and prepare

   /* TODO */                      // user code

   ml8rt_exit_function(0,0,0);     // restore registers
}
```

Die Funktion erhält in *status* den Fehlerstatus: 1='Stark gestörter Bus', 2='Bus Off'. Die anderen Funktionsparameter sind ohne Bedeutung. Die Funktion darf jedoch keine Daten zurückgeben!

Verwaltung der Message-Puffer des CAN-Controllers

Damit eine Nachricht gesendet oder empfangen werden kann, muß das zugehörige Message-Objekt in einem der Message-Puffer des CAN-Contollers auf dem Modul eingetragen sein. Der CAN-Controller verfügt hierzu über 14 Message-Puffer (Puffer 1 bis 14), die zum Senden und Empfangen genutzt werden können, und einen speziellen Message-Puffer (Puffer 15), der ausschließlich zum Empfangen genutzt werden kann.

Das Eintragen eines Message-Objekts in einen Message-Puffer nimmt eine gewisse Zeit (einige µs) in Anspruch, um die das Verschicken der Nachricht gegenüber einem bereits im Message-Puffer eingetragenen Message-Objekt länger dauert. Wenn ein Message-Objekt bereits vorab in einen Message-Puffer eingetragen ist, so kann die Reaktionszeit auf empfangene Nachrichten oder die Latenzzeit für zu sendende Nachrichten verkürzt werden.

Aus diesem Grunde werden alle 15 Message-Puffer des Controllers vom Treiberprogramm M049TASK verwaltet. Das Programm ist speziell darauf ausgerichtet, die Puffer möglichst effektiv nutzen zu können. Hierbei hat das richtige Verständnis zur Pufferverwaltung entscheidenden Einfluß auf die Effektivität Ihres Anwendungsprogrammes.

Wenn Sie ein Modul M-CAN-1 für maximal 14 Message-Objekte konfigurieren, dann wird auf diesem Modul jedes Message-Objekt dauerhaft in einen der dafür exklusiv reservierten Puffer 1 bis 14 eingetragen. Dieser Fall ist die Optimal-Konfiguration des Moduls M-CAN-1: Weder muß zum aktiven Senden oder Anfordern von Daten die ID in einen Puffer eingetragen werden, noch muß zum Empfangen einer

Nachricht die zugehörige ID aus einem Puffer gelesen und ausgewertet werden. Weiterhin wird die MODULAR-4 Karte minimal belastet, da ausschließlich die gewünschten (d.h. die konfigurierten) Nachrichten empfangen werden. Sie können in diesem Fall die nachfolgenden Abschnitte überspringen und auf Seite 9-15 weiterlesen.

Wenn Sie ein Modul M-CAN-1 für mehr als 14 Message-Objekte konfigurieren, so sollten sie sich intensiv mit den folgenden Abschnitten beschäftigen.

Wenn Sie beim Konfigurieren (mit **m049_config_msg_object**) des Message-Objekts das Flag *_M049_EXCL_BUFFER* setzen, so zwingen Sie die Pufferverwaltung dazu, einen der dafür möglichen Puffer 1 bis 13 abzugeben und ihn exklusiv für dieses Message-Objekt zu reservieren. In diesem Falle wird das Message-Objekt bereits zum Zeitpunkt der Konfiguration dauerhaft in einen der freien Puffer 1 bis 13 eingetragen und der dafür genutzte Puffer exklusiv für dieses Message-Objekt reserviert. Bei einem **exklusiven Puffer** muß weder zum aktiven Senden oder Anfordern von Daten die ID in den Puffer eingetragen werden, noch muß zum Empfangen einer Nachricht die zugehörige ID aus dem Puffer gelesen und ausgewertet werden.

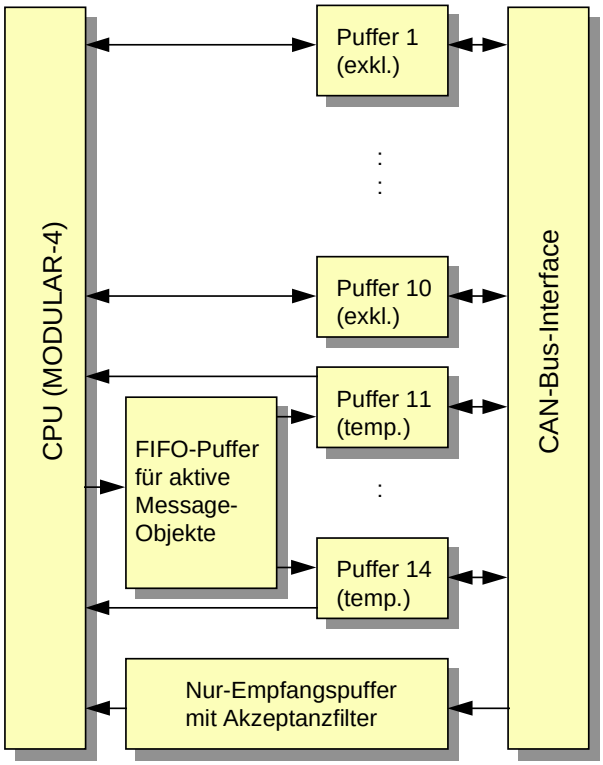
Die von den Puffern 1 bis 13 nicht exklusiv reservierten Puffer sowie Puffer 14 werden von der Pufferverwaltung als **temporäre Puffer** verwendet. Diese Puffer stehen für aktive Message-Objekte zur Verfügung. Ein Empfangsobjekt sendet über einen solchen Puffer einen Remote-Frame, ein Sendeobjekt sendet über einen solchen Puffer einen Data-Frame. Das Treiberprogramm M049TASK trägt hierzu das Message-Objekt - falls es nicht bereits in einem temporären Puffer gespeichert ist - in einen dieser temporären Puffer ein, sobald es gesendet werden soll. Der so belegte temporäre Puffer ist danach solange nicht verfügbar, bis das gesendete Objekt die Empfangsbestätigung (signalisiert durch Interrupt) erhält. Danach bleibt das Message-Objekt in diesem Puffer solange gespeichert, bis es durch ein anderes Message-Objekt überschrieben wird.

! *Wenn gerade kein temporärer Puffer verfügbar ist, so wird die Nachricht in einen* ***FIFO-Puffer*** *des Treiberprogramms eingetragen. Wenn einer der temporären Puffer frei wird, so rückt eine Nachricht aus dem FIFO-Puffer nach und wird automatisch gesendet. Zu beachten ist, daß nur das Handle des Message-Objekts im FIFO-Puffer gespeichert wird und nicht dessen Daten. Zu dem Zeitpunkt, an dem das Treiberprogramm das Handle aus dem FIFO entnimmt und das zugehörige Message-Objekt versendet, sind die Daten unter Umständen nicht die selben wie zu dem Zeitpunkt an dem das Handle in den FIFO-Puffer eingetragen wurde. Aus diesem Grund liefert das Treiberprogramm eine Warnung, wenn ein Message-Objekt nicht sofort versendet werden kann.*

Alle passiven Message-Objekte, die nicht in einen exklusiven Puffer eingetragen werden, werden zum Zeitpunkt der Konfiguration (mit **m049_config_msg_object**) in den **Nur-Empfangspuffer** (Puffer 15) eingetragen. Hierbei werden bereits eingetragene Message-Objekte nicht überschrieben. Statt dessen überlagern sich die IDs aller eingetragenen Message-Objekte und bilden ein digitales Akzeptanzfilter, das zur Entlastung der MODULAR-4 Karte nicht gewünschte Nachrichten ausfiltert.

Sie können mit dem Treiberprogramm beliebig viele aktive Message-Objekte verwalten. Bei den passiven Message-Objekten ist eine durch den CAN-Controller bedingte Einschränkung zu beachten: Alle Message-Objekte, die über den Nur-Empfangspuffer empfangen werden sollen, müssen vom selben Typ sein, d.h. es können darüber entweder nur passive Sendeobjekte oder passive Empfangsobjekte empfangen werden. Die exklusiv genutzten Puffer sind hiervon nicht betroffen. Darin können sowohl passive Sende- als auch Empfangsobjekte eingetragen werden.

Die Abbildung rechts zeigt, wie die Pufferverwaltung des Treiberprogrammes M049TASK die Message-Puffer nutzt, wenn mehr als 14 Message-Objekte konfiguriert sind und davon 10 Message-Objekte die Eigenschaft *_M049_EXCL_BUFFER* haben.



Damit das **digitale Akzeptanzfilter** unerwünschte Nachrichten möglichst gut ausfiltern kann, besteht es aus einer ID-Maske und einer Akzeptanzmaske. Beide Masken werden bitweise aus allen in den Nur-Empfangspuffer eingetragenen IDs gebildet. An den Bitpositionen, an denen alle eingetragenen IDs den gleichen Bitwert haben, ist das Bit in der Akzeptanzmaske = 1 und in der ID-Maske gleich dem Bitwert. An den übrigen Bitposition enthalten beide Masken eine 0. Logisch bedeutet dies: Die Akzeptanzmaske entsteht durch eine XNOR-Verknüpfung aller eingetragenen IDs, die ID-Maske entsteht durch eine AND-Verknüpfung aller eingetragenen IDs.

! *Nur Message-Objekte mit der Eigenschaft _M049_PASSIVE gehen in die Berechnung des Akzeptanzfilters ein!*

Eine Nachricht passiert das Akzeptanzfilter nur dann, wenn in der ID der Nachricht an allen Positionen, an denen in der Akzeptanzmaske eine 1 steht, der Bitwert gleich dem Bitwert in der ID-Maske ist.

Beispiel: In den Nur-Empfangspuffer werden die IDs 03h und 07h eingetragen.

IDs:	03h=	0	0	0	0	0	0	0	0	0	1	1
	07h=	0	0	0	0	0	0	0	0	1	1	1
Akzeptanzmaske:		1	1	1	1	1	1	1	1	0	1	1
ID-Maske:		0	0	0	0	0	0	0	0	0	1	1

Es gilt: Akzeptanzmaske = 03h XNOR 07h, ID-Maske = 03h AND 07h. Die IDs, die das Filter passieren können, lassen sich leicht aus der entstehenden ID-Maske herauslesen. Bis auf die Position, die grau unterlegt ist (d.h. das Bit in der Akzeptanzmaske ist = 0), müssen die IDs mit der ID-Maske übereinstimmen. An der grau unterlegten Position spielt der Bitwert der ID keine Rolle. Schlußfolgerung: Das Akzeptanzfilter akzeptiert in diesem Falle tatsächlich nur die IDs 03h und 07.

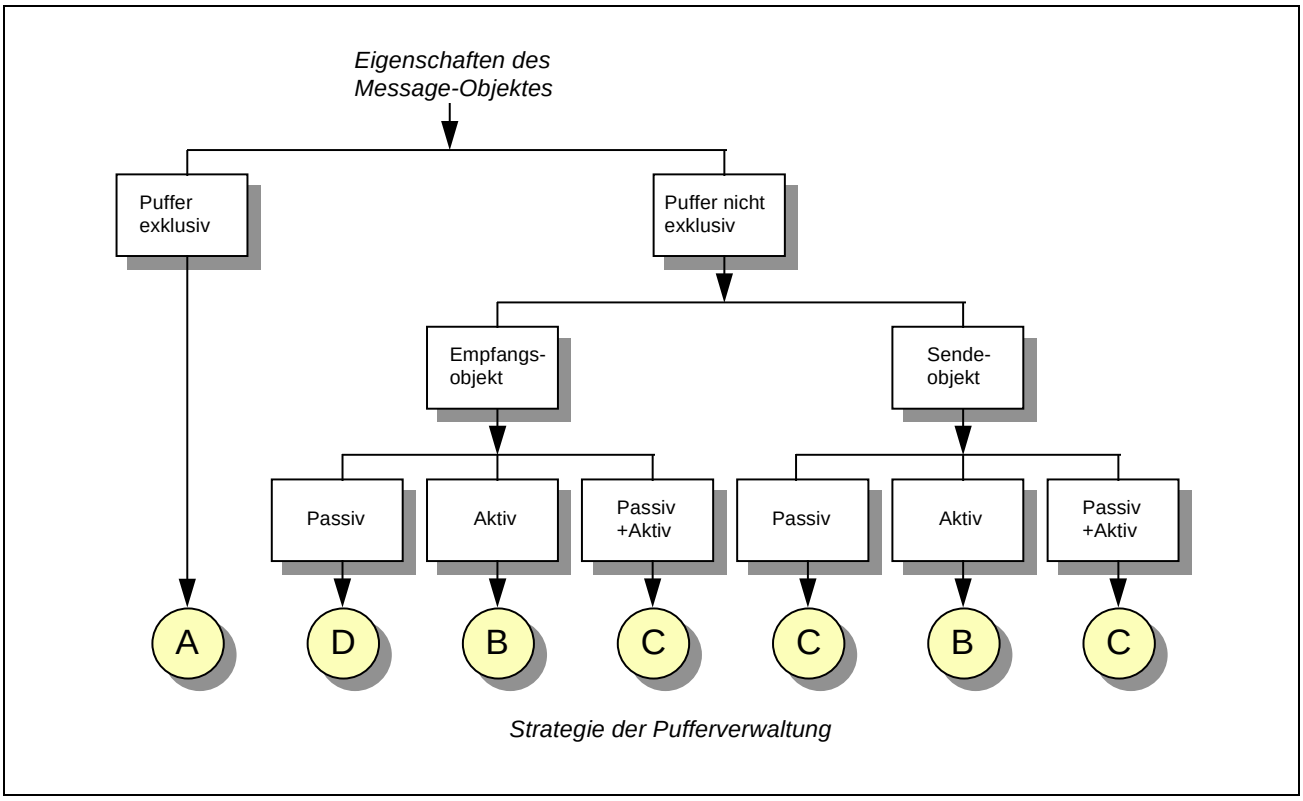
Allerdings ist es nicht immer möglich, das Akzeptanzfilter so optimal einzustellen, daß wirklich nur die gewünschten Nachrichten vom CAN-Controller empfangen werden. Bei ungünstiger Vergabe der IDs kann es dazu kommen, daß die Masken des Akzeptanzfilters so ungünstig gebildet werden, daß sehr viele unerwünschte Nachrichten das Filter passieren. Die unerwünschten Nachrichten werden zwar vom Treiberprogramm verworfen, belasten aber dennoch die CPU der MODULAR-4 Karte!

Beispiel: In den Nur-Empfangspuffer werden die IDs 20h und 07h eingetragen.

IDs:	20h=	0	0	0	0	0	1	0	0	0	0	0
	07h=	0	0	0	0	0	0	0	0	1	1	1
Akzeptanzmaske:		1	1	1	1	1	0	1	1	0	0	0
ID-Maske:		0	0	0	0	0	0	0	0	0	0	0

Im Vergleich zum vorigen Beispiel sind nun mehrere Positionen grau unterlegt (d.h. das Bit in der Akzeptanzmaske ist = 0). Das Filter läßt in diesem Falle die IDs 0h bis 7h und 20h bis 27h passieren. Schlußfolgerung: Das Akzeptanzfilter akzeptiert in diesem Falle außer den gewünschten IDs 20h und 07 auch weitere 14 unerwünschte IDs.

Das folgende Diagramm dient dazu, die Wirkungsweise der Pufferverwaltung noch weiter zu vertiefen. Ausgehend von den zugewiesenen Eigenschaften eines Message-Objekts liefert es die von der Pufferverwaltung angewandte Strategie (A, B, C oder D), welche Puffer und wie diese Puffer genutzt werden.

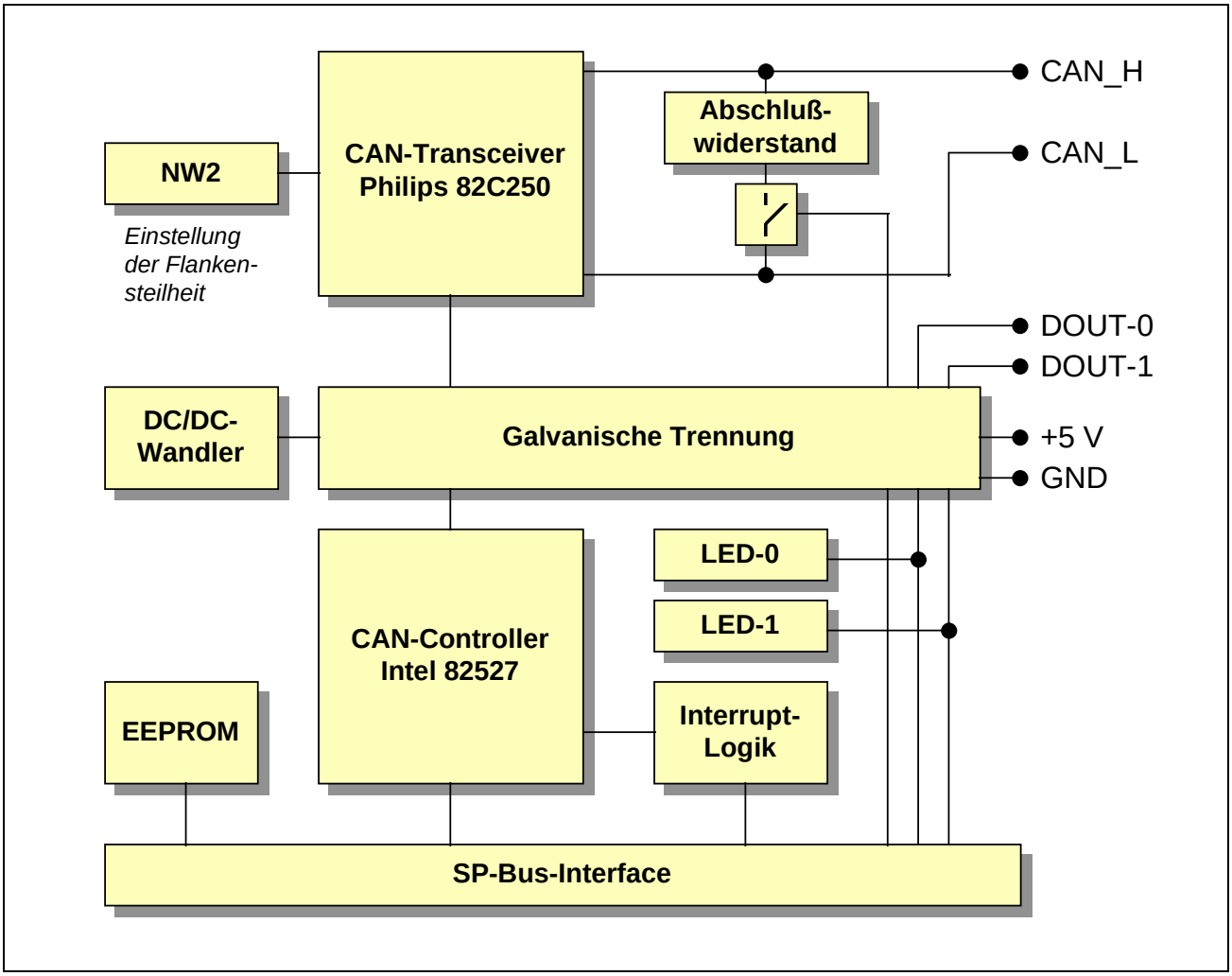


Strategie	Bedeutung
A	Das Message-Objekt wird zum Zeitpunkt seiner Konfiguration dauerhaft in einen der dafür exklusiv reservierten Puffer eingetragen. Es ist das Senden und Empfangen von Nachrichten, sowohl passiv als auch aktiv, möglich, ohne daß das Message-Objekt vorher nochmals in einen Puffer eingetragen werden muß.
B	Das Message-Objekt wird zum Zeitpunkt, wann es aktiv Daten sendet oder anfordert, in einen der freien temporären Puffer eingetragen (falls es nicht schon eingetragen ist). Der genutzte Puffer wird vorübergehend gesperrt. Daraufhin wird die Nachricht auf dem CAN-Bus verschickt. Nach der Empfangsbestätigung steht der verwendete temporäre Message-Puffer wieder zur Verfügung.
C	Das Message-Objekt wird zum Zeitpunkt seiner Konfiguration dauerhaft in den Nur-Empfangspuffer eingetragen. Zum Zeitpunkt, wann es aktiv Daten sendet oder anfordert, wird es in einen der temporären Message-Puffer eingetragen (falls es nicht schon eingetragen ist). Daraufhin wird die Nachricht auf dem CAN-Bus verschickt.
D	Das Message-Objekt wird zum Zeitpunkt seiner Konfiguration dauerhaft in den Nur-Empfangspuffer eingetragen und wartet darauf, bis es eine Nachricht erhält.

Zusammenfassend sollten Sie die folgenden Hinweise beachten:

- Um eine optimale Bearbeitungsgeschwindigkeit zu erreichen (d.h. die Anzahl und die Dauer der Zugriffe auf das Modul zu minimieren), legen Sie eine optimale Anzahl an exklusiven Puffern fest und weisen den gewünschten Message-Objekten die Eigenschaft *_M049_EXCL_BUFFER* zu. Dies sollte für Message-Objekte gelten, die sich durch einen besonders häufigen Zugriff auf den CAN-Bus auszeichnen, oder die sehr zeitkritisch sind.
- Gibt es eine sehr hohe Zahl an Message-Objekten, die alle gleich häufig an der CAN-Bus-Kommunikation teilnehmen, kann es auch sinnvoll sein, keine exklusiven Puffer zu verwenden, damit das Treiberprogramm M049TASK die Message-Objekte möglichst gleichmäßig auf die 14 Puffer verteilen kann. Würden sich alle Message-Objekte z.B. nur einen temporären Puffer teilen, müßte dieser praktisch vor jeder Kommunikation neu mit dem jeweiligen Message-Objekt geladen werden.
- Achten Sie bei der Vergabe der IDs für die Message-Objekte, die in den Nur-Empfangspuffer eingetragen werden, darauf, daß nicht zu viele unerwünschte Nachrichten das Akzeptanzfilter des Nur-Empfangspuffers passieren. Dies erreichen Sie, in dem Sie IDs verwenden, die möglichst ähnliche Bitmuster aufweisen.
- Wenn sie ein Message-Objekt als passives Sendeobjekt konfigurieren, so sollten Sie die Eigenschaft *_M049_EXCL_BUFFER* zuweisen.

Blockschaltbild



Technische Daten

Parameter	Wert	Einheit
CAN-Bus Controller	Intel 82527	-
CAN-Bus Transceiver	Philips 82C250	-
Einstellbare Bitrate	5 bis 1000	kBit/s
Trennspannung	500	V
Digitalausgänge DOUT-0 bis DOUT-1:[1]		
Anzahl	2	-
Maximaler Kollektorstrom bei log. 0, max.	10	mA
Maximaler Kollektorstrom bei log. 1, max.	10	µA
Ausgangsspannung bei log. 0, max.	0,8	V
Ausgangsspannung bei log. 1, max.	50	V
+5V Ausgang für externe Zwecke:		
Maximal zulässige Stromentnahme	80	mA
Busabschlußwiderstand, typ.[2]	110	Ω
Stromaufnahme (+5V, ±5%), typ.[3]:	280	mA
Betriebstemperatur	0 bis 60	°C
Abmessungen (L x B x H)	106 x 45 x 15	mm

Lieferumfang

- Modul M-CAN-1
- Datenträger mit Treiberprogramm M049TASK.EXE, Programmbibliotheken (Pascal und C) und Beispielprogrammen

---

[1] Parallel zu jedem der beiden Digitalausgänge befindet sich auf dem Modul je eine LED, die jeweils zusammen mit dem Digitalausgang geschaltet wird.

[2] Der Busabschlußwiderstand besteht aus einem 100 Ω Festwiderstand und einem dazu in Reihe geschalteten Analogschalter, dessen Widerstand typ. bei 10 Ω liegt. Der Wert ist temperatur- und spannungsabhängig und kann sich bis auf max. 125 Ω erhöhen. In CAN-Bus Netzwerken in denen ein exakter Wert des Widerstands erforderlich ist, sollte der Widerstand abgeschaltet werden (EEPROM-Wort 7) und durch einen externen Busabschlußwiderstand ersetzt werden.

[3] Stromaufnahme gemessen, wenn beide Digitalausgänge auf log. 1 gesetzt sind und kein Strom für externe Zwecke entnommen wird.

Konfiguration und Einbau

Das Modul ist fertig konfiguriert für den Betrieb im High Speed Mode. Die durch die Transistoren im CAN-Transceiver vorgegebene Flankensteilheit wird nicht begrenzt.

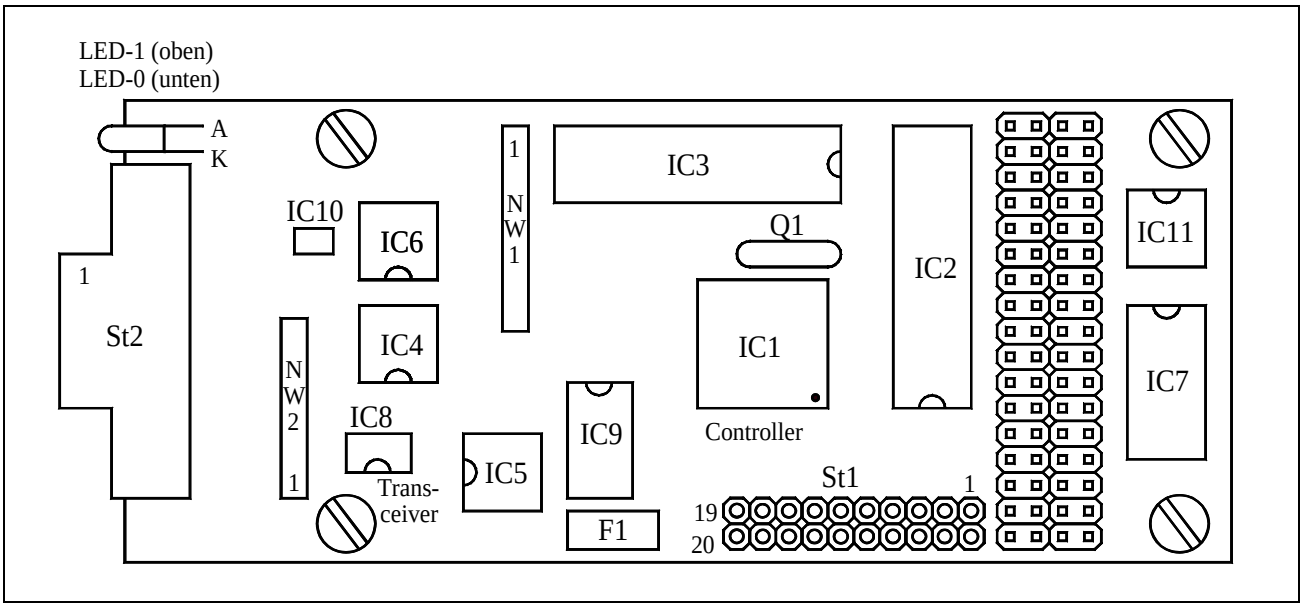
Bei kleineren Übertragungsraten kann zur Verbesserung der EMV durch Austauschen des Widerstandsnetzwerkes NW2 die Flankensteilheit (Slew-Rate) auf Werte zwischen 4,7 V/µs und 40 V/µs begrenzt[1] werden. Der einzusetzende Widerstandswert für NW2 berechnet sich nach der folgenden Formel:

$$R = \frac{658}{SR} \cdot k\Omega$$

mit SR: Flankensteilheit in V/µs (Wertebereich: 4,7 bis 40)

Sonstige Einstellungen erfolgen nach dem Einbau des Moduls per Software.

Lageplan



[1] Durch die Begrenzung der Flankensteilheit sinkt die maximal mögliche Buslänge gegenüber dem High Speed Mode ab: Mit R = 24 kΩ um ca. 30 m, mit R = 47 kΩ um ca. 70 m (Angaben laut Philips 82C250 Datenblatt).

EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt.

WORT	Binär		Hex.	Bedeutung (Kurzinfo)
0	0010 0001	0011 0001	2131h	Modultyp M-CAN-1, Rev. A
1	0000 0000	0000 0001	0001h	Initialisierung
2	0000 1001	1100 0100	09c4h	Bit-Timing-Parameter t_Q
3	0000 0000	0001 0000	0010h	Bit-Timing-Parameter t_{SEG1}/t_Q
4	0000 0000	0000 0011	0003h	Bit-Timing-Parameter t_{SEG2}/t_Q
5	0000 0000	0000 0010	0002h	Bit-Timing-Parameter t_{SJW}/t_Q
6	0000 0000	0000 0001	0001h	Bit-Timing-Parameter Samples per Bit
7	0000 0000	0000 0000	0000h	Abschlußwiderstand
8	0000 0000	0000 0011	0003h	Digitalausgänge und LEDs
9	0000 0000	0000 0000	0000h	ID-Format
10	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
31	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	1	0	0	0	1	WORT-0: Kennung
								0	0	1	1	0	0	0	1	Modultyp: 49 = M-CAN-1
				0	0	0	1									Revision: 1=A, 2=B, 3=C, etc.
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort wird eingestellt, ob das Modul nach dem Einschalten und bei einem Hardware-Reset der MODULAR-4 Karte entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-1: Initialisierung
																geändert am: von:
															1	Init nach Hardware-Reset: 0=nein, 1=ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

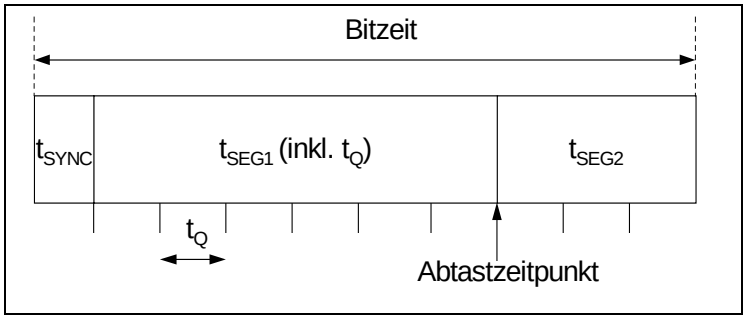
WORT-2 bis WORT-6: Einstellung der Bit-Timing-Parameter

Laut CAN-Spezifikation ist ein Bit in vier Zeitabschnitte unterteilt:

- Zeitabschnitt t_{SYNC} zur Synchronisation
- Zeitabschnitt t_{PROP} zur Kompensation der physikalischen Verzögerungszeiten
- Zeitabschnitt t_{SEG1} vor dem nominellen Abtastzeitpunkt
- Zeitabschnitt t_{SEG2} nach dem nominellen Abtastzeitpunkt

Die beiden Zeitabschnitte t_{SEG1} und t_{SEG2} dienen der Nachsynchronisation zum Ausgleich der Phasendifferenz zwischen Sende- und Empfangsoszillator. Für den auf dem Modul verwendeten CAN-Controller Intel 82527 werden t_{PROP} und t_{SEG1} zu einem Segment t_{SEG1} zusammengefaßt.

Alle Zeitabschnitte werden in Vielfachen des Time-Quantums t_Q angegeben. Das Time-Quantum t_Q ist immer ein ganzzahliges Vielfaches der Periodendauer (= 125 ns) des Oszillators Q1 auf dem Modul.



Die Bitzeit berechnet sich nach der folgenden Formel, in der alle Größen frei konfigurierbar sind:

$$\text{Bitzeit} = (t_{SEG1}/t_Q + t_{SEG2}/t_Q + 1) \cdot t_Q$$

Zusätzlich kann die *maximale Synchronisationssprungweite* (Resynchronisation Jump Width t_{SJW}) eingestellt werden. Dies ist die maximale, pro Nachsynchronisation zulässige Verlängerung bzw. Verkürzung von t_{SEG1} und t_{SEG2}. Diese Größe wird ebenfalls als Vielfaches von t_Q angegeben.

Zur Verbesserung der Störsicherheit kann die *Abtastrate pro Bit* (Samples per Bit) auf den Wert = 3 eingestellt werden.

Die Bit-Timing-Parameter sind frei konfigurierbar. Allerdings müssen folgende Bedingungen[1] erfüllt sein:

- $t_{SEG1}/t_Q + t_{SEG2}/t_Q \geq 7$
- $t_{SEG2} \geq t_{SJW}$
- $t_{SEG1} \geq t_{SJW} + t_{PROP}$ für Samples per Bit=1
- $t_{SEG1} \geq t_{SJW} + t_{PROP} + 2 \cdot t_Q$ für Samples per Bit=3

Wenn Sie sich an der CiA-Empfehlung 102, Version 2.0 orientieren wollen, so verwenden Sie eine der in der folgenden Tabelle aufgeführten *Standard-Bitraten* und stellen die dafür angegebenen Bit-Timing-Parameter ein:

Bitrate (kBit/s)	**t_Q (ns)**	**t_{SEG1}/t_Q**	**t_{SEG2}/t_Q**	**t_{SJW}/t_Q**	**Samples per Bit**
10	5000	16	3	2	1
20 [2]	2500	16	3	2	1
50	1000	16	3	2	1
125	500	13	2	1	1
250	250	13	2	1	1
500	125	13	2	1	1
800	125	7	2	1	1
1000	125	5	2	1	1

[1] Der Zeitabschnitt t_{PROP} ist physikalisch vorgegeben: $t_{PROP} = 2 \cdot (t_{PHYS} + t_{DRV} + t_{CC})$. t_{PHYS}, t_{DRV} und t_{CC} sind die maximalen Verzögerungen, entstehend durch die physikalische Signallaufzeit auf dem Bus (abhängig von Buslänge und Übertragungsrate), durch den Ausgangstreiber (max. 80 ns) sowie durch den Eingangskomparator des CAN-Controllers (max. 60 ns).

[2] Werkseitig eingetragene Bit-Timing-Parameter.

Beachten Sie auch, daß das Erzielen hoher Bitraten Auswirkungen auf die maximal zulässige Buslänge hat! In der folgenden Tabelle sind Richtwerte für die mit den jeweiligen Bitraten - jeweils gültig für den High Speed Mode, also ohne Begrenzung der Flankensteilheit - erreichbaren maximalen Buslängen angegeben.

Bitrate (kBit/s)	10	20	50	125	250	500	800	1000
Maximale Buslänge (m)	5000	2500	1000	500	250	100	50	25

In WORT-2 bis WORT-6 werden die Bit-Timing-Parameter eingestellt:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0	WORT-2: t_Q (ns)
																geändert am: von:
			0	1	0	0	1	1	1	0	0	0	1	0	0	Wertebereich: 125 bis 8000
0	0	0														Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	WORT-3: t_{SEG1}/t_Q
																geändert am: von:
											1	0	0	0	0	Wertebereich: 3 bis 16
0	0	0	0	0	0	0	0	0	0	0						Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	WORT-4: t_{SEG2}/t_Q
																geändert am: von:
												0	0	1	1	Wertebereich: 2 bis 8
0	0	0	0	0	0	0	0	0	0	0	0					Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	WORT-5: t_{SJW}/t_Q
																geändert am: von:
													0	1	0	Wertebereich: 1 bis 4
0	0	0	0	0	0	0	0	0	0	0	0	0				Reserviert

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-6: Samples per Bit
																geändert am: von:
														0	1	Wertebereich: 1 oder 3
0	0	0	0	0	0	0	0	0	0	0	0	0	0			Reserviert

WORT-7: Einstellung des CAN-Bus-Abschlußwiderstandes

In WORT-7 kann eingestellt werden, ob der CAN-Bus-Abschlußwiderstand auf dem Modul zugeschaltet wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-7: CAN-Bus-Abschlußwiderstand
																geändert am: von:
															0	Widerstand zuschalten: 0=nein, 1=ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

WORT-8: Initialisierung der Digitalausgänge und LEDs

In WORT-8 werden die Digitalausgänge DOUT-0 und DOUT-1 initialisiert. Eine logische Null bedeutet, daß der Open-Collector-Ausgang leitend ist. Eine logische Eins bedeutet, daß der Open-Collector-Ausgang gesperrt ist.

Die beiden LEDs auf dem Modul signalisieren, ob der zugehörige Open-Collector-Ausgang leitend ist oder nicht: LED-0 für DOUT-0, LED-1 für DOUT-1. Wenn der Ausgang leitend ist, ist die zugehörige LED eingeschaltet. Anderenfalls ist sie ausgeschaltet.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	WORT-8: Initialisierung DOUT-0 und -1
																geändert am: von:
															1	DOUT-0: 0=Tr. leitend, 1=Tr. gesperrt
														1		DOUT-1: 0=Tr. leitend, 1=Tr. gesperrt
0	0	0	0	0	0	0	0	0	0	0	0	0	0			Reserviert

WORT-9: Einstellung des ID-Formats

In WORT-9 kann eingestellt werden, ob die CAN-Bus-Telegramme 11-Bit-IDs (Bit-0 = 0) oder 29-Bit-IDs verwenden (Bit-0 = 1).

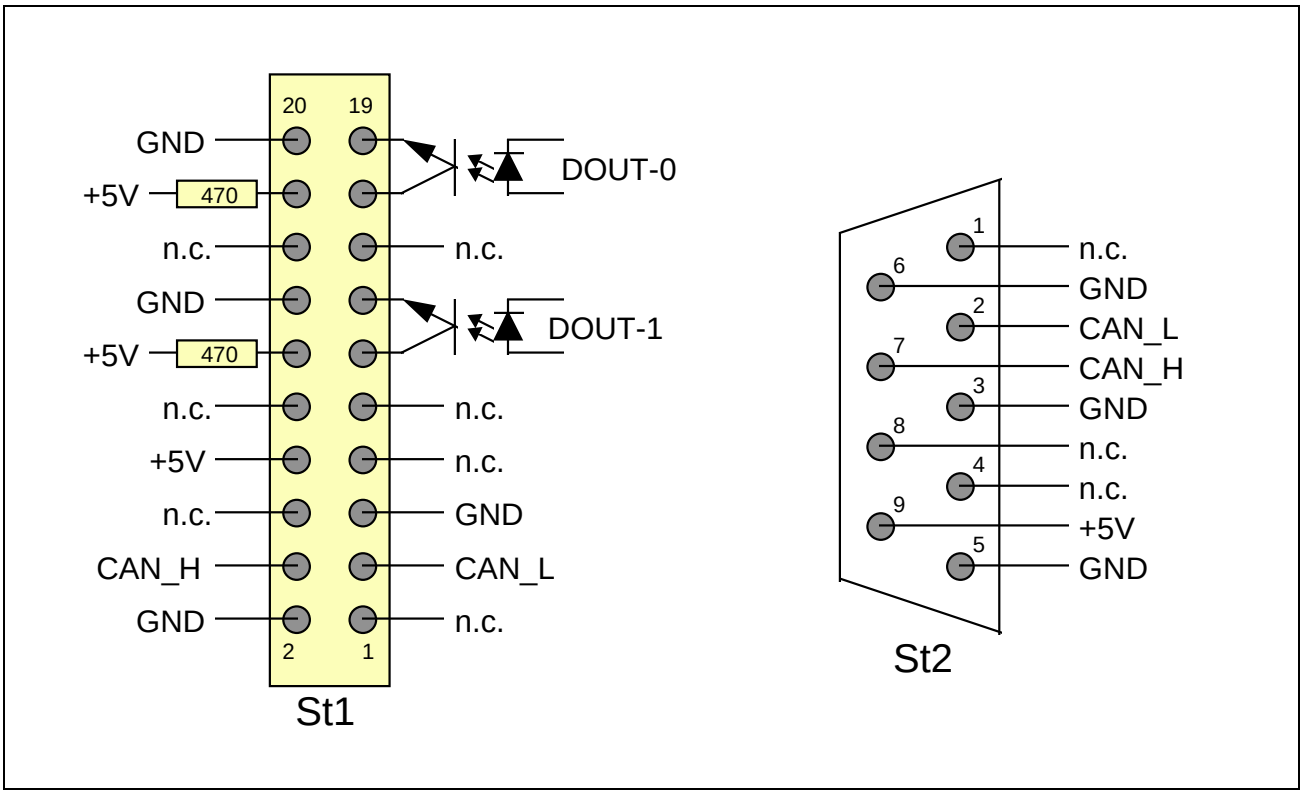
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-7: ID-Format
																geändert am: von:
															0	ID-Format: 0=11-Bit, 1=29-Bit
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

Steckerbelegung

Das Modul verfügt über zwei Anschlußstecker. Der CAN-Bus kann wahlweise entweder an St1 oder an St2 angeschlossen werden. Beim Anschluß von CAN_L und CAN_H muß auf die richtige Polung geachtet werden.

St1 ist ein 20-poliger Pfostensteckverbinder. Die Pins 1 bis 10 entsprechen dem CiA-Standard 102 Version 2.0 für Multipole Connectors. Der CAN-Bus wird über eine Zweidrahtleitung, die verdrillt oder abgeschirmt sein kein, angeschlossen. Zusätzlich liegen an den Pins 11 bis 20 die beiden Digitalausgänge des Moduls. Die beiden potentialfreien Digitalausgänge können über die gegenüberliegenden Kontakte mit GND und 5V (über 470 Ω) verbunden werden. Ebenso können externe Geräte vom +5V-Ausgang versorgt werden. Dieser Ausgang ist über die Sicherung F1 (125 mA) geschützt.

St2 ist ein 9-poliger D-SUB-Stecker nach CiA-Standard 102, Version 2.0.



Hochsprachenbibliothek

Hinweise zum Einbinden der Bibliothek finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliothek'. Der Name der Bibliothek (*libname*) lautet **M049_LIB**, Sie finden sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Funktion **m049_bib_startup** einmal aufgerufen werden. Der Rückgabewert aller Funktionen ist ein Fehlerstatus (siehe Seite 9-39).

Während der Konfigurationsphase müssen Sie unbedingt die folgende Reihenfolge der Funktionsaufrufe einhalten:

Schritt	Aktion
1	M049TASK installieren.
2	Bibliothek mit **m049_bib_startup** initialisieren.
3	Das Modul M-CAN-1 mit **m049_config_module** konfigurieren.
4	Message-Objekte mit **m049_config_msg_object** konfigurieren.
5	*Optional:* Akzeptanzmaske mit **m049_set_acceptance_mask** setzen
6	CAN-Kommunikation mit **m049_start_com** starten.

Zur fehlerfreien Funktion der Bibliothek muß das Treiberprogramm M049TASK.EXE pro aufgestecktem Modul M-CAN-1 einmal auf der MODULAR-4 Karte installiert sein (Tasknummer und Interrupt-Nummer dürfen nicht mehrfach verwendet werden). Dieses Programm wird zusätzlich zur Bibliothek M049_LIB mitgeliefert. Das Treiberprogramm kann entweder mit einer INS-Datei und dem Hilfsprogramm SNW oder mit der Bibliotheksfunktion **ml8_transfer_and_install** installiert werden (Hinweise dazu finden Sie in Ihrem Benutzerhandbuch zur MODULAR-4 Karte).

Installationsparameter	Wert
Programmnummer	410h
Flags	0f8ah
Interrupt-Nummer	INT-A bis INT-F oder NMI
Tasknummer	Frei wählbar (20 bis 1024)
Datenbereichsgröße	0

m049_bib_startup — Initialisiere die Modulbibliothek

Pascal FUNCTION m049_bib_startup : WORD;

C ushort EXPORT m049_bib_startup (void);

Funktion Diese Funktion initialisiert die Modulbibliothek und prüft, ob für jedes aufgesteckte Modul M-CAN-1 je ein Treiberprogramm M049TASK installiert ist. Jede Anwendung, die das Modul benutzt, muß diese Funktion vor allen anderen aufrufen. Die Funktion sorgt für eine Zuordnung zwischen Modul-Steckplatznummer und Tasknummer des Treiberprogramms. Das Modul mit der niedrigsten Steckplatznummer wird der Instanz des Treiberprogramms mit der niedrigsten Tasknummer zugeordnet.

m049_config_module — Konfiguriere das Modul

Pascal FUNCTION m049_config_module (micro_slot: BYTE; VAR config: M049ConfigType) : WORD;

C ushort EXPORT m049_config_module (byte micro_slot, M049ConfigType *config);

Funktion Diese Funktion konfiguriert das Modul M-CAN-1. Die Funktion muß für jedes aufgesteckte Modul M-CAN-1 genau einmal ausgeführt werden.

Parameter *config:* Zeiger auf eine Struktur vom Typ **M049ConfigType**.

Die Datenstruktur **M049ConfigType** enthält die Parameter zur Konfiguration des Moduls M-CAN-1:

Feldbezeichner	Typ	Beschreibung
messages	ushort	Gibt die genaue Anzahl der benötigten Message-Objekte an[1].
serv_task	ushort	Tasknummer zur Error-Serviceroutine.
serv_func	ushort	Funktionsnummer zur Error-Serviceroutine.

[1] Dieser Parameter hat ab Version 2.A des Treiberprogramms keine Bedeutung mehr.

Hinweis Wenn Sie eine ungültige Task- oder Funktionsnummer übergeben, so kann das zum späteren Programmabsturz führen! !

m049_config_msg_object Konfiguriere Message-Objekt

Pascal FUNCTION m049_config_msg_object (micro_slot: BYTE; VAR handle: M049Handle; VAR config: M049MsgType): WORD;

C ushort EXPORT m049_config_msg_object (byte micro_slot, M049Handle *handle, M049MsgType *config);

Funktion Diese Funktion konfiguriert ein Message-Objekt und liefert für Zugriffe auf das Message-Objekt das zugehörige Handle. Die Nutzdaten des Message-Objekts werden mit 0 initialisiert.

Parameter *handle*: Zeiger auf eine Variable, in die das Handle für das konfigurierte Message-Objekt eingetragen wird. Im Fehlerfall trägt die Funktion eine Null ein.

config: Zeiger auf eine Datenstruktur vom Typ **M049MsgType**.

Die Datenstruktur **M049MsgType** enthält die Parameter zur Konfiguration eines Message-Objekts:

Feldbezeichner	Typ	Beschreibung
id	ulong	Eindeutiger CAN-Identifier, der dem Message-Objekt zugeordnet wird (Wertebereich 0 bis 2031 für CAN-Spezifikation 2.0 A (11-Bit- Identifier) und 0 bis 268435447 für CAN-Spezifikation 2.0 B (29-Bit-Identifier)). Das verwendete Format wird im EEPROM WORT-9 festgelegt.[1]
control_flags	byte	Eigenschaften des Message-Objekts (siehe unten).
serv_task	ushort	Tasknummer zur Message-Serviceroutine.
serv_func	ushort	Funktionsnummer zur Message-Serviceroutine.

! Hinweis Wenn Sie eine ungültige Task- oder Funktionsnummer übergeben, so kann das zum späteren Programmabsturz führen!

[1] Unter gewissen Umständen könnte es aufgrund eines Fehlers des CAN-Controllers dazu kommen, daß dieser von sich aus einen Remote-Frame mit der ID 0 verschickt. Dieses ließ sich beim Testen nicht feststellen. Eventuell hat Intel dieses Problem bereits beseitigt. Sollte es dennoch auftreten, so können Sie es einfach umgehen, indem Sie in Ihrem Anwendungsprogramm kein Message-Objekt mit der ID 0 konfigurieren.

In *control_flags* weisen Sie dem Message-Objekt bestimmte Eigenschaften zu. Die gewünschten Eigenschaften können durch bitweise OR-Verknüpfung der entsprechenden Flags beliebig kombiniert werden.

Flags zur Konfiguration als Sendeobjekt

Zur Konfiguration als Sendeobjekt dürfen zunächst ausschließlich die in der folgenden Tabelle aufgeführten Flags gesetzt werden. Das Flag *_M049_SEND* muß gesetzt werden. Ebenso muß mindestens eines der beiden Flags *_M049_ACTIVE* und *_M049_PASSIVE* gesetzt werden.

Flag	Bedeutung
_M049_SEND	Das Message-Objekt wird als Sende-Objekt konfiguriert.
_M049_ACTIVE	Daten können auf Ihren Wunsch hin mit **m049_send_data** verschickt werden.
_M049_PASSIVE	Das Message-Objekt wartet solange, bis Daten per Remote-Frame angefordert werden. Daraufhin sendet das Message-Objekt seine Daten *automatisch*.

Flags zur Konfiguration als Empfangsobjekt

Zur Konfiguration als Empfangsobjekt dürfen zunächst ausschließlich die in der folgenden Tabelle aufgeführten Flags gesetzt werden. Mindestens eines der beiden Flags *_M049_ACTIVE* und *_M049_PASSIVE* muß gesetzt werden.

Flag	Bedeutung
_M049_ACTIVE	Daten können auf Ihren Wunsch hin per Remote-Frame mit **m049_send_data_request** angefordert werden.
_M049_PASSIVE	Das Message-Objekt wartet auf den Empfang von Daten.

Flag zur Steuerung der Pufferverwaltung

Das Treiberprogramm M049TASK verwendet standardmäßig für alle Message-Objekte zum Senden einen temporären Puffer und zum Empfangen den Nur-Empfangspuffer. Durch Setzen des Flags *_M049_EXCL_BUFFER* wird einer der Message-Puffer 1 bis 13 exklusiv für das Message-Objekt - unabhängig davon, ob es als Sende- oder Empfangsobjekt konfiguriert wird - reserviert. Diese Eigenschaft darf nur maximal 13 Message-Objekten zugeteilt werden!

! *Beachten Sie, daß alle Message-Objekte, bei denen das Flag* _M049_PASSIVE *ohne gleichzeitig gesetztes Flag* _M049_EXCL_BUFFER *gesetzt ist, vom selben Typ sein müssen: Message-Objekte mit diesen Eigenschaften müssen entweder alle Sendeobjekte (*_M049_SEND *gesetzt) oder alle Empfangsobjekte (*_M049_SEND *nicht gesetzt) sein.*

Flag zur Interrupt-Unterdrückung

Ab der Version 2.A des Treiberprogramms M049TASK können die Daten eines Sendeobjekts versendet werden, ohne daß darauf mit einem Interrupt reagiert wird. Dazu muß das Flag *_M049_DISABLE_INT* bei der Konfiguration des Message-Objekts gesetzt werden. Dies kann bei Sendeobjekten, die keine Reaktion erfordern sinnvoll sein, um die CPU-Belastung zu verringern.

Ein Aufruf von m049_send_data bietet für solche Message-Objekte keine Garantie, daß die Daten auch tatsächlich auf den Bus gesendet werden. Der CAN-Controller kann die Daten erst versenden, wenn der Bus frei ist. Im Fall, daß der Bus stark belastet ist, kann das einige Zeit dauern. Wenn in dieser Zeit Sendeanforderungen für weitere Message-Objekte an das Treiberprogramm übergeben werden, können diese die Sendeanforderung eines Message-Objekts mit gesetztem _M049_DISABLE_INT-*Flag löschen. Dies kann durch zusätzliches Setzen von* _M049_EXCL_BUFFER *verhindert werden.*
Message-Objekte mit _M049_DISABLE_INT *sind immer aktiv und passiv.*

Beispiel: Die folgenden Anweisungen konfigurieren mit dem Modul M-CAN-1, das auf Steckplatz 1 aufgesteckt ist, ein Sendeobjekt, dem der Identifier ID = 7 zugeordnet wird. Das Message-Objekt soll sowohl auf Wunsch als auch auf Anforderung eines anderen Teilnehmers Daten verschicken können. Als Message-Serviceroutine wird die Funktion 4 der Task 100 zugeteilt.

```
/* variables to configure message object */
M049MsgType msg_config;
M049Handle hMessage1;
ushort error;

/* parameters for message object to be configured */
msg_config.id                 = 7L;
msg_config.control_flags      = _M049_SEND | _M049_ACTIVE |
                                _M049_PASSIVE;
msg_config.serv_task          = 100;
msg_config.serv_func          = 4;

/* configure message object and get handle for this object */
error = m049_config_msg_object (1, &hMessage1, &msg_config);
```

m049_start_com — Starte die CAN-Kommunikation

Pascal FUNCTION m049_start_com (micro_slot: BYTE) : WORD;

C ushort EXPORT m049_start_com (byte micro_slot);

Funktion Diese Prozedur startet die CAN-Kommunikation. Sie darf erst dann aufgerufen werden, wenn alle Message-Objekte konfiguriert sind.

m049_stop_com — Beende die CAN-Kommunikation

Pascal FUNCTION m049_stop_com (micro_slot: BYTE) : WORD;

C ushort EXPORT m049_stop_com (byte micro_slot);

Funktion Diese Prozedur beendet die CAN-Kommunikation.

m049_set_data Setze Nutzdaten eines Message-Objekts

Pascal FUNCTION m049_set_data (micro_slot: BYTE; handle: M049Handle; VAR data: BYTE; size: BYTE): WORD;

C ushort EXPORT m049_set_data (byte micro_slot, M049Handle handle, byte *data, byte size);

Funktion Diese Funktion setzt die Nutzdaten des durch *handle* repräsentierten Message-Objekts, das als Sendeobjekt konfiguriert sein muß. Das Message-Objekt wird allerdings noch nicht gesendet. Hierzu muß ggf. **m049_send_data** ausgeführt werden.

Parameter *handle*: Handle des Message-Objekts.

data: Zeiger auf die einzutragenden Nutzdaten. Von diesen Nutzdaten werden *size* Byte in das Nutzdatenfeld des Message-Objekts eingetragen.

size: Anzahl Nutzdatenbyte (Wertebereich: 0 bis 8).

m049_get_data Lies die Nutzdaten eines Message-Objekts

Pascal FUNCTION m049_get_data (micro_slot: BYTE; handle: M049Handle; VAR data_var: BYTE; VAR size_var: BYTE) : WORD;

C ushort EXPORT m049_get_data (byte micro_slot, M049Handle handle, byte *data_var, byte *size_var);

Funktion Diese Funktion liest Nutzdaten des durch *handle* repräsentierten Message-Objekts.

Parameter *handle*: Handle des Message-Objekts.

data_var: Zeiger auf eine Variable, die die Nutzdaten (max. 8 Byte) des Message-Objekts übernimmt.

size_var: Zeiger auf eine Variable, die der Funktion die maximal zu lesenden Datenbyte übergibt und von der Funktion die Anzahl der tatsächlich gelieferten Datenbyte übernimmt. Mögliche Werte für die Anzahl Datenbyte sind 0 bis 8.

m049_send_data Sende Daten

Pascal FUNCTION m049_send_data (micro_slot: BYTE; handle: M049Handle): WORD;

C ushort EXPORT m049_send_data (byte micro_slot; M049Handle handle);

Funktion Diese Funktion sendet das durch *handle* repräsentierte Message-Objekt, das als aktives Sendeobjekt konfiguriert sein muß. Beim Senden wird ein Data-Frame auf dem CAN-Bus verschickt.

Parameter *handle*: Handle des Message-Objekts.

m049_send_data_request Sende Datenanforderung

Pascal FUNCTION m049_send_data_request (micro_slot: BYTE; handle: M049Handle): WORD;

C ushort EXPORT m049_send_data_request (byte micro_slot, M049Handle handle);

Funktion Mit dieser Funktion wird ein Remote-Frame auf dem CAN-Bus verschickt um von einem anderen Busteilnehmer Daten anzufordern. Das durch *handle* repräsentierte Message-Objekt muß als aktives Empfangsobjekt konfiguriert sein.

Parameter *handle*: Handle des Message-Objekts.

m049_clear_msg Deaktiviere ein aktives Message-Objekt

PASCAL FUNCTION m049_clear_msg (micro_slot: BYTE; handle: M049Handle): WORD;

C ushort m049_clear_msg (byte micro_slot, M049Handle handle);

Funktion Das Modul M-CAN-1 wiederholt einen mit **m049_send_data** oder **m049_send_data_request** gestarteten Sendeversuch solange, bis ein anderer Busteilnehmer den Empfang der Nachricht bestätigt. Ist ein Busteilnehmer ausgefallen, so bleibt die Empfangsbestätigung aus. Dies hat zur Folge, daß die Message-Serviceroutine nicht aufgerufen wird und - wenn dafür ein temporärer Puffer genutzt wird - der temporäre Message-Puffer blockiert wird. Zudem wird durch die ständige Wiederholung des Sendeversuchs der CAN-Bus belastet.

Abhilfe schafft diese Funktion, die den Sendeversuch abbricht und den ggf. blockierten temporären Message-Puffer wieder freigibt.

Parameter *handle*: Handle des Message-Objekts.

m049_set_acceptance_mask Setze die Akzeptanzmaske

Pascal FUNCTION m049_set_acceptance_mask (micro_slot: BYTE; VAR mask: LONGINT; usage: BYTE; serv_task, serv_func: WORD): WORD;

C ushort EXPORT m049_set_acceptance_mask (byte micro_slot, ulong *mask, byte usage, ushort serv_task, ushort serv_func);

Funktion Mit dieser Funktion kann das Modul so konfiguriert werden, daß es über den Nur-Empfangspuffer mehr Nachrichten empfängt, als eigentlich konfiguriert wurden. Dies ist z.B. dann sinnvoll, wenn Sie ein Monitorprogramm für alle Nachrichten auf dem CAN-Bus erstellen möchten. Hierbei müssen Sie allerdings beachten, daß das bereits bestehende Filter in seiner Filterfunktion nicht eingeschränkt wird. Ebenso müssen sie beim Parameter *usage* beachten, daß der Nur-Empfangspuffer entweder Data-Frame- oder Remote-Frame-Telegramme empfangen kann. Die für den Nur-Empfangspuffer bereits konfigurierten Message-Objekte müssen somit berücksichtigt werden.

In Standardanwendungen benötigen Sie diese Funktion nicht, da das Treiberprogramm M049TASK die Akzeptanzmaske aus den IDs der konfigurierten Message-Objekte automatisch errechnet.

Wenn Sie diese Funktion ausführen, müssen Sie dafür ebenfalls eine Message-Serviceroutine (siehe Seite 9-8) schreiben und diese der Funktion durch die Angabe der Task- und Funktionsnummer übergeben. Diese Message-Serviceroutine wird immer dann aufgerufen, wenn eine unkonfigurierte Nachricht empfangen wird. Da diese Routine für Nachrichten mit verschiedenen Identifiern gültig ist, ist es die Aufgabe der Message-Serviceroutine, die ID mit **m049_get_id** auszuwerten.

Parameter *mask*: Übergibt an die Funktion die Bitmaske, mit der die Akzeptanzmaske des Nur-Empfangspuffers gesetzt werden soll (der Wert 0 bedeutet, daß alle Nachrichten empfangen werden) und übernimmt von der Funktion den tatsächlichen Wert der Akzeptanzmaske (im Fehlerfall wurde die Akzeptanzmaske nicht verändert).

usage: Gibt an, ob über den Nur-Empfangspuffer Remote-Frame-Telegramme (= 1) oder Data-Frame-Telegramme (= 0) empfangen werden sollen.

serv_task: Tasknummer zur Message-Serviceroutine.

serv_func: Funktionsnummer zur Message-Serviceroutine.

Wenn Sie eine ungültige Task- oder Funktionsnummer übergeben, kann das zum späteren Programmabsturz führen! Die Funktion darf pro Modul nur einmal ausgeführt werden! **!**

m049_clear_configuration — Lösche Treiberkonfiguration

Pascal FUNCTION m049_clear_configuration (micro_slot: BYTE): WORD;

C ushort EXPORT m049_clear_configuration (byte micro_slot);

Funktion Diese Funktion löscht die gesamte Message-Objekt-Konfiguration im Treiberprogramm. Danach kann ohne nochmaligen Aufruf von m049_config_module eine neue Message-Objekt-Konfiguration erstellt werden. Bevor das Modul wieder an der Bus-Kommunikation teilnehmen kann, muß m049_start_com ausgeführt werden.

m049_get_id — Lies die ID eines Message-Objekts

Pascal FUNCTION m049_get_id (micro_slot: BYTE; handle: M049Handle; VAR id_var: LONGINT): WORD;

C ushort EXPORT m049_get_id (byte micro_slot, M049Handle handle, ulong *id_var);

Funktion Diese Funktion liefert die ID des durch *handle* repräsentierten Message-Objekts.

Parameter *handle*: Handle des Message-Objekts.

id_var: Zeiger auf eine Variable, die die ID des Message-Objekts übernimmt.

m049_get_diagnosis Lies Diagnosemeldung des Treiberprog.

Pascal FUNCTION m049_get_diagnosis (micro_slot: BYTE): WORD;

C ushort EXPORT m049_get_diagnosis (byte micro_slot);

Funktion Mit dieser Funktion erhalten Sie genauen Aufschluß über die Ursache eines Fehlers in der zuletzt bearbeiteten Funktion des Treiberprogrammes M049TASK. Eine Übersicht über die möglichen Diagnosemeldungen und ihre Bedeutung finden Sie auf Seite 9-39. Die Diagnosemeldung wird vom Treiberprogramm nach dem Lesen = 0 gesetzt.

m049_set_dout Setze einen Digitalausgang

Pascal PROCEDURE m049_set_dout (micro_slot, dout, data: BYTE);

C void EXPORT m049_set_dout (byte micro_slot, byte dout, byte data);

Funktion Diese Funktion setzt einen der beiden Digitalausgänge (Open-Collector-Ausgänge) des Moduls M-CAN-1. Gleichzeitig wird zur optischen Kontrolle auch die zugehörige LED auf dem Modul geschaltet.

Parameter: *dout*: Nummer des Digitalausgangs bzw. der LED (0 oder 1).

data: Zu setzender Wert (Wertebereich: 0=Ausgang auf LOW setzen, d.h. der Ausgangstransistor ist leitend, die LED leuchtet; 1=Ausgang auf HIGH setzen, d.h. der Ausgangstransistor ist gesperrt, die LED leuchtet nicht).

m049_get_dout Lese den Zustand eines Digitalausgangs

Pascal FUNCTION m049_get_dout (micro_slot, dout: BYTE): BYTE ;

C byte EXPORT m049_get_dout (byte micro_slot, byte dout);

Funktion Diese Funktion liest den Zustand eines der beiden Digitalausgänge des Moduls (0=Ausgangstransistor ist leitend, die LED leuchtet; 1=Ausgangstransistor ist gesperrt, die LED leuchtet nicht).

Parameter: *dout*: Nummer des Digitalausgangs bzw. der LED (0 oder 1).

Reaktion auf CAN-Bus-Ereignisse im PC-Programm

Für den Fall, daß auf Echtzeitprogrammierung verzichtet werden soll, bietet das Treiberprogramm die Funktion 14 (M049_SRQ_SERVICE_FUNC). Diese kann als Message-Service-Routine und als Error-Service-Routine angegeben werden. Die Funktion sendet einen gepufferten Service-Request an den PC und speichert die Ursache des Requests im Treiberprogramm ab. Für die Auswertung des Service-Requests im PC-Programm steht die Funktion ***m049_evaluate_srq*** zur Verfügung.

Es ist zu beachten, daß es bei der Behandlung von CAN-Bus-Ereignissen im PC-Programm zu Dateninkonsistenzen kommen kann. Zu dem Zeitpunkt, an dem das PC-Programm den Service-Request auswertet, kann das Treiberprogramm bereits neue Daten für das Message-Objekt empfangen haben. Liest das PC-Programm in der Service-Routine die Daten des Message-Objekts aus, so sind dies unter Umständen nicht die, die den Service-Request ausgelöst haben, sondern neuere.

Um dieses zu verhindern besteht ab der Version 2.A des Treiberprogramms die Möglichkeit, den Empfang von Nachrichten im Anschluß an den Aufruf der Funktion 14 für eine bestimmte Zeit zu unterbinden. Um diese Funktionalität zu aktivieren, ist die Funktion ***m049_enable_int_mask*** aufzurufen (die Funktion steht nur in PC-Programmen zur Verfügung). Bei jedem Aufruf der Funktion 14 wird dann automatisch der für das M-CAN-1 Modul angewählte Interrupt maskiert. Der CAN-Controller auf dem Modul kann zwar weiterhin Nachrichten vom CAN-Bus empfangen, er kann diese dann aber nicht mehr an das Treiberprogramm weitergeben. Dadurch ist sichergestellt, daß die Daten eines Message-Objekts zwischen dem Auslösen des Service-Requests und dem Auslesen vom PC nicht verändert werden. Sobald das PC-Programm mit der Behandlung des Service-Requests fertig ist, muß es dies dem Treiberprogramm durch Aufruf der Funktion ***m049_end_of_srq*** mitteilen. Der Interrupt wird dann wieder freigegeben.

Je länger der Interrupt gesperrt bleibt, desto höher wird die Wahrscheinlichkeit, daß Nachrichten verloren gehen. Empfängt der CAN-Controller während dieser Zeit zwei Nachrichten, so kann das Treiberprogramm im Anschluß an die Freigabe des Interrupts nur auf die zweite Nachricht reagieren, die erste ist verloren. **!**

Bei Verwendung von Echtzeit-Service-Routinen bestehen keine Probleme bezüglich Datenkonsistenz und Message-Verlust.

m049_evaluate_srq Werte des Service-Request aus

Pascal FUNCTION m049_evaluate_srq (VAR micro_slot: BYTE;
VAR handle: M049Handle): WORD

C ushort EXPORT m049_evaluate_srq (byte *micro_slot,
M049Handle *handle);

Funktion Diese Funktion kann nur in **PC-Programmen** verwendet werden. Sie erfüllt nur dann ihren Zweck, wenn Sie als Message-Serviceroutine eines Message-Objekts oder als Error-Serviceroutine die Funktion **14** des Treiberprogrammes M049TASK angegeben haben. In diesem Fall können Sie auf die Echtzeitprogrammierung verzichten, da diese Funktion einen gepufferten Service-Request an den PC sendet und die Ursache des Requests im Treiberprogramm abspeichert. In Ihrem PC-Programm müssen Sie den Service-Request auswerten.

Die Funktion wertet einen empfangenen Service-Request aus. Das Treiberprogramm verwendet den Wert 80h im Low-Byte des Service-Request-Wortes. Im High-Byte steht der Steckplatz (1 bis 10) des auslösenden M-CAN-1 Moduls.

Wenn der Rückgabewert der Funktion = 0 und das von der Funktion erhaltene Handle ≠ 0 ist, so signalisiert der Service-Request eine erfolgreiche Bus-Kommunikation des durch das Handle repräsentierten Message-Objekts.

Wenn der Rückgabewert der Funktion ≠ 0 ist, so signalisiert der Service-Request das Auftreten eines Busfehlers (mögliche Rückgabewerte sind: 1 = 'Stark gestörter Bus', 2 = 'Bus Off').

Wenn der Rückgabewert der Funktion und das Handle = 0 ist, so ist vom Treiberprogramm kein Service-Request gesendet worden.

Parameter *micro_slot*: Zeiger auf eine Variable zur Übernahme des Steckplatzes des auslösenden Moduls.

handle: Zeiger auf eine Variable zur Übernahme des Handles für das Message-Objekt, das den Service-Request ausgelöst hat (d.h. das Daten empfangen bzw. erfolgreich gesendet hat).

m049_enable_int_mask — Aktiviere Interrupt-Sperrung durch Funktion 14 des Treiberprogramms

Pascal PROCEDURE m049_enable_int_mask (micro_slot: BYTE);

C void EXPORT m049_enable_int_mask (byte micro_slot);

Funktion Diese Funktion steht nur in PC-Programmen zur Verfügung. Sie aktiviert den auf Seite 8-36 beschriebenen Mechanismus der Sperrung des verwendeten Interrupts durch Funktion 14 des Treiberprogramms. Dadurch bleiben die Daten eines Message-Objekts konsistent.

m049_end_of_srq — Gebe gesperrten Interrupt frei

Pascal PROCEDURE m049_end_of_srq (micro_slot: BYTE);

C void EXPORT m049_end_of_srq (byte micro_slot);

Funktion Diese Funktion steht nur in PC-Programmen zur Verfügung. Sie muß nach der Beendigung der Service-Request-Behandlung aufgerufen werden um den Interrupt wieder freizugeben (nur wenn der Mechanismus zur Interruptsperrung durch m049_enable_int_mask aktiviert ist).

Fehlerstatus der Bibliotheksroutinen

Die Bibliotheksroutinen liefern Fehlercodes des Betriebssystems zurück. Der Rückgabewert nimmt den Wert 42e0h (in Echtzeitprogramm) bzw. 42h (in PC-Programm) an, wenn das Treiberprogramm einen Fehler signalisiert. Genaueren Aufschluß über die treiberinterne Ursache liefert die Funktion **m049_get_diagnosis** (siehe Seite 9-36), die eine Diagnosemeldung bezüglich des letzten Aufrufes einer Treiberfunktion liefert. Eine Diagnosemeldung kann außer der Ursache einer Fehlermeldung auch eine Warnung (dies wird nicht durch eine Fehlermeldung signalisiert) sein. Die Routine **m049_get_diagnosis** kann folgende Diagnosemeldungen liefern:

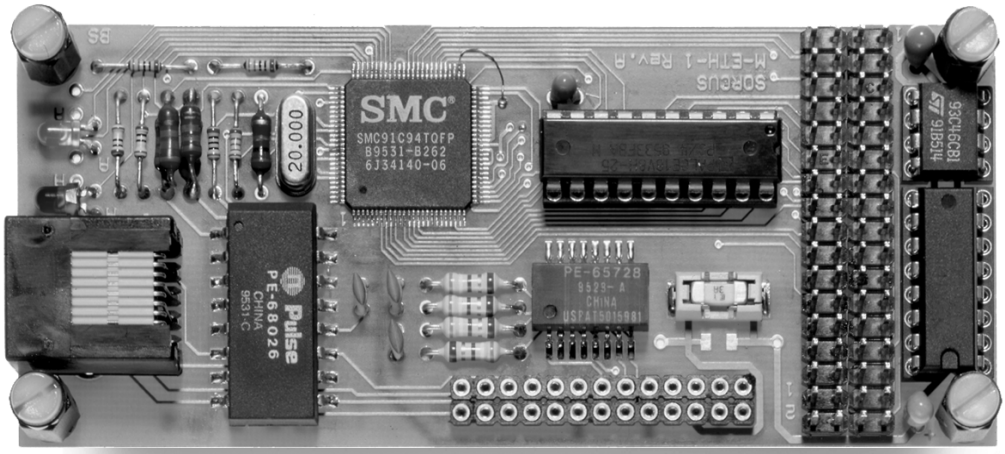
Wert	Bedeutung
0	Die Funktion wurde fehlerfrei ausgeführt.
1	Die Anzahl der empfangenen Nutzdatenbyte stimmt nicht mit den geforderten überein (Warnung).
2	Die angegebene Serviceroutine ist noch nicht installiert (Warnung).
3	Der letzte Sendewunsch für ein Message-Objekt ist noch nicht auf dem Bus abgesetzt worden (Warnung).
4	Ein Frame für dieses Message-Objekt wurde empfangen, aber noch nicht vom Interrupt behandelt (Warnung).
5	Eine empfangene Nachricht ist verloren gegangen, weil der Interrupt nicht bearbeitet werden konnte, bevor eine neue Nachricht angekommen ist (Warnung).
6	Das Message-Objekt wurde im FIFO-Puffer zwischengespeichert und konnte nicht sofort versendet werden (Warnung).
21h	Nicht genug Speicherplatz für alle Message-Objekte.
22h	Das CAN-Modul ist nicht ansprechbar.
23h	Das Treiberprogramm wurde unter einem unzulässigen Interrupt installiert.
24h	Ein EEPROM-Konfigurationsparameter liegt nicht im erlaubten Wertebereich.
25h	Ein Parameter liegt nicht im erlaubten Wertebereich.
26h	entfällt
27h	Es wurde versucht, mehr als 13 Message-Objekten die Eigenschaft *_M049_EXCL_BUFFER* zuzuweisen.
28h	Es wurde versucht, mehrere Message-Objekte mit derselben ID zu konfigurieren.
29h	Der Wert der mit m049_set_acceptance_mask gesetzten Akzeptanzmaske würde dazu führen, daß nicht mehr alle konfigurierten Nachrichten empfangen werden können.
2ah	Das übergebene Handle verweist auf ein Message-Objekt vom falschen Typ oder ist ungültig.
2bh	Der FIFO-Puffer ist voll. Das Treiberprogramm kann vorübergehend keine weiteren Sendewünsche entgegennehmen.

Wert	Bedeutung
2ch	Der Nur-Empfangspuffer konnte nicht richtig konfiguriert werden. Es wurden sowohl passive Sendeobjekte als auch passive Empfangsobjekte konfiguriert, die über diesen Puffer empfangen werden sollen (d.h. bei denen das Flag *_M049_EXCL_BUFFER* nicht gesetzt ist). Wenn Sie beide Typen verwenden, müssen Sie allen Message-Objekten eines Typs exklusive Puffer zuweisen.
2dh	Die vorgeschriebene Reihenfolge beim Aufruf der Funktionen während der Konfigurationsphase wurde nicht eingehalten.

Notizen

10. M-ETH-1

Ethernet-Controller mit AUI-Schnittstelle und 10Base-T-Anschluß



Funktionsbeschreibung

Das SP-Bus-Modul M-ETH-1 ist ein Kommunikationsmodul für das MODULAR-4 System, das eine Ankopplung an Ethernet-Netzwerke ermöglicht. Es hat folgende Eigenschaften:

- Single-Chip-Ethernet-Controller mit RAM (SCECR) SMC91C94 implementiert die Schichten PHY (Physical Layer) und MAC (Media Access Control)
- Vollständig per Software konfigurierbar
- RJ45-Buchse für Anschluß an 10Base-T-Ethernet (Twisted-Pair) direkt auf dem Modul
- AUI-Schnittstelle über Modulsteckleiste, Flachbandkabel und D-Sub-15-Buchse für alle Topologien (mit externem Transceiver)
- Übertragungsrate 10 MBit/s
- Automatische Polaritätskorrektur bei 10Base-T (bei vertauschten Adern im Kabel)
- Full Duplex Betrieb möglich
- EEPROM für Initialisierungsdaten
- 4 LEDs auf dem Modul für Diagnosezwecke
- Treiberprogramm für einfaches Senden und Empfangen von Ethernet-Paketen

Ethernet

Ethernet ist das am weitesten verbreitete LAN (Local Area Network / lokales Netzwerk). Es zeichnet sich durch geringe Kosten und hohe Flexibilität bei der Wahl der Verkabelung aus. Die verschiedenen Verkabelungsarten und das Zugriffsverfahren CSMA/CD (Carrier Sense Multiple Access / Collision Detection) sind in den internationalen Normen IEEE 802.3 und ISO 8802-3 festgelegt worden. Das ursprüngliche Ethernet (10Base-5 und später 10Base-2) basiert auf einem Koaxialkabel und wird als Bus verlegt, an den alle teilnehmenden Stationen angeschlossen werden. Das verwendete Kabel wird also möglichst nah (10Base-5) oder direkt (10Base-2) an den teilnehmenden Stationen vorbeigelegt.

Heute wird allerdings immer häufiger ein Kabel eingesetzt, das aus verdrillten Adern (Twisted Pair) besteht. Diese Art der Verkabelung (10Base-T) wird dann nicht mehr als Bus ausgeführt. Die anzuschließenden Stationen werden in diesem Fall sternförmig mit Ethernet-Hubs oder Ethernet-Switches verbunden.

Werden hierbei Kabel und Verbindungkomponenten nach Kategorie 5 (auch Cat. 5) eingesetzt, kann ohne Neuverkabelung eine Migration zu Fast Ethernet (Übertragungsrate 100 MBit/s) stattfinden.

Ethernet-Pakete

Daten werden bei Ethernet-Netzen in Form von Paketen übertragen. Diese Pakete können bis zu 1518 Byte lang sein. Ein Teil dieser Daten ist dabei üblicherweise für bestimmte Inhalte reserviert. So enthalten z.B. die ersten 6 Byte die Zieladresse, die darauf folgenden 6 Byte die Absenderadresse und die letzten 4 Byte eines Pakets eine Prüfsumme (Frame Checksum).

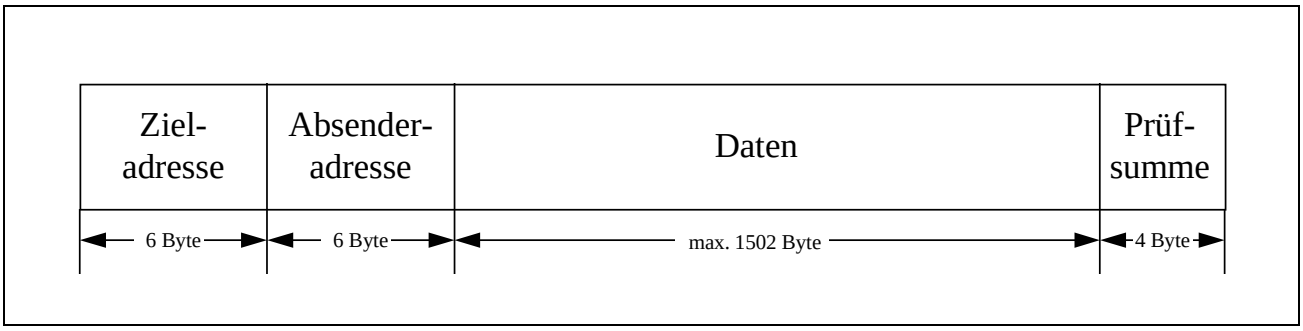


***Abb. 10-1**: Aufbau eines Ethernet-Paketes*

Jede an einem Ethernet-Netzwerk teilnehmende Station (jede Netzwerkkarte und auch jedes M-ETH-1 Modul) hat eine weltweit eindeutige, individuelle Hardware-Adresse von sechs Byte Länge.

Um mit einem MODULAR-4 System mit aufgestecktem Modul M-ETH-1 Ethernet-Pakete senden und empfangen zu können, stehen Programmbibliotheken zur Verfügung, die diese Funktionalität zur Verfügung stellen.

Zunächst kann das Modul verschieden konfiguriert werden. Folgende Konfigurationsmöglichkeiten stehen dabei zur Verfügung:

- Verwendeter Anschluß des Moduls: 10Base-T oder AUI
- Verwendete Art der I/O-Zugriffe auf das Modul: 8- oder 16-Bit
- Ständiges Prüfen der Verbindung bei 10Base-T (Link Test)
- Eigene Hardware-Adresse (wird nur zum Filtern von Empfangspaketen verwendet)
- Automatisches Anhängen der Prüfsumme an Sendepakete
- Automatisches Abschneiden der Prüfsumme von Empfangspaketen
- Automatisches Verwerfen von Paketen mit fehlerhafter Prüfsumme
- Betriebsart: Half Duplex oder Full Duplex
- Automatischer Sendeabbruch bei mangelhafter Signalqualität (Signal Quality Error)
- Aufruf einer Serviceroutine bei Sendefehler, Überlauf eines Statistikzählers oder Unterbrechung der Verbindung
- Empfang von allen Paketen oder nur solchen, die an die eigene Hardware-Adresse gerichtet sind (Promiscuous Mode)

Der Konfigurationsvorgang und die einzelnen Konfigurationsmöglichkeiten sind in den Abschnitten EEPROM-Inhalte und Programmbibliotheken genauer beschrieben.

Wird beim Installieren des Treiberprogramms die Prozedur AUTO_INIT ausgeführt, so wird das Modul automatisch entsprechend den EEPROM-Eintragungen konfiguriert und kann direkt danach verwendet werden. Die Funktionalität des Treiberprogramms kann in drei Bereiche untergliedert werden: Senden, Empfangen und Steuerung.

Zum Senden von Paketen steht eine Bibliotheksfunktion zur Verfügung, an die ein Zeiger auf einen Verbund übergeben werden muß. Durch diesen Verbund wird be-

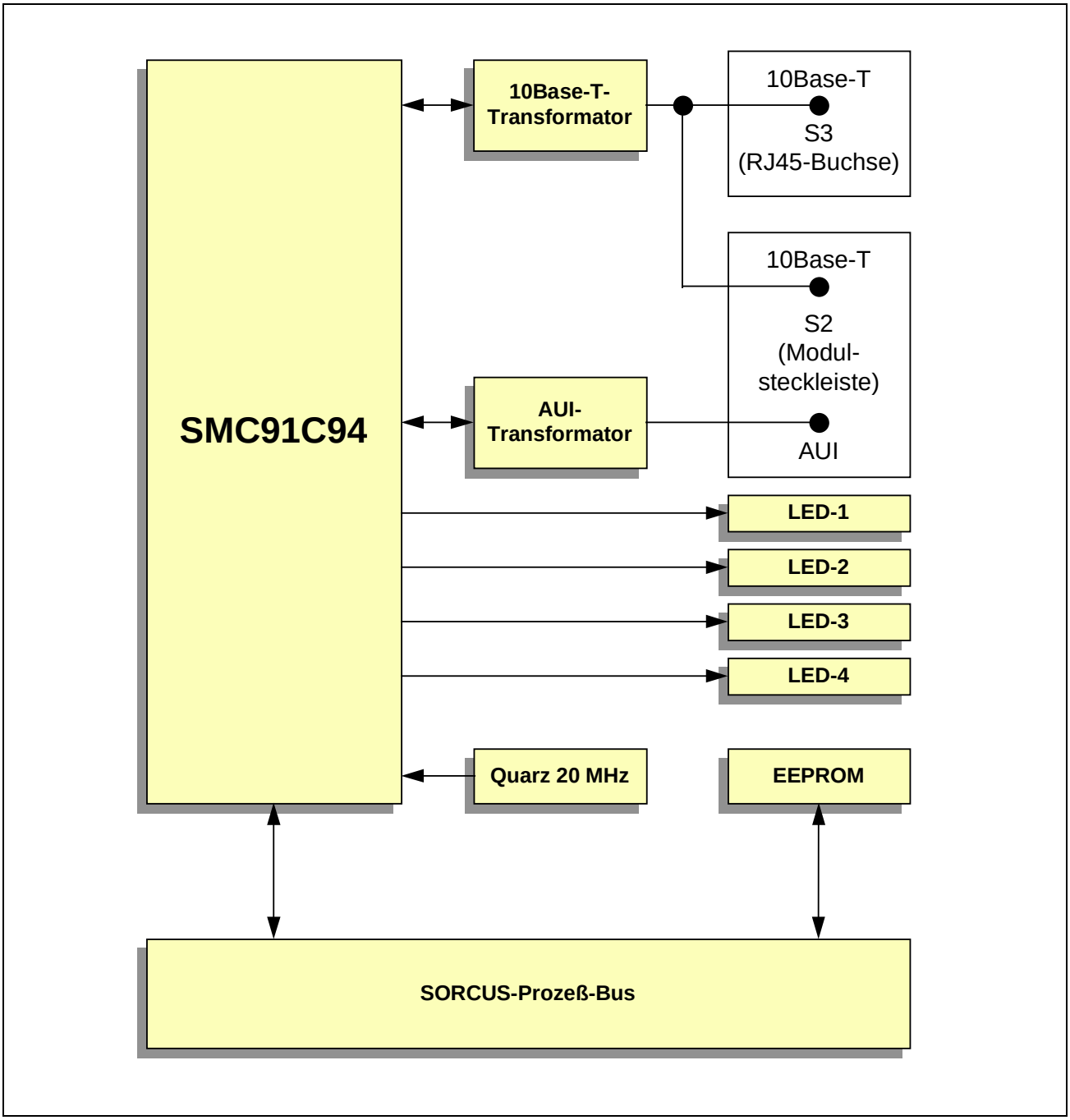
schrieben, an welchen Stellen die Abschnitte, aus denen sich das zu sendende Paket zusammensetzt (die zu sendenden Daten), im Speicher stehen.

Um Pakete empfangen zu können, muß dem Treiberprogramm die Funktion eines Echtzeitprogramms angegeben werden. Wenn kein geschwindigkeitsoptimiertes Programm benötigt wird, stellt das Treiberprogramm eine Funktion zur Verfügung, die hierfür verwendet werden kann. Wenn z.B. empfangene Ethernet-Pakete nur an den PC übertragen werden sollen, wird durch diese Funktion und die Verwendung der Programmbibliothek der Programmieraufwand erheblich reduziert, da das Erstellen von Echtzeitprogrammen entfallen kann.

Wenn mehrere Anwendungen auf der MODULAR-4 Karte Ethernet-Pakete empfangen sollen, muß diese Funktion als Teil eines Echtzeitprogramms zur Verfügung gestellt werden. Das Vorgehen und die hierbei zu beachtenden Randbedingungen sind in einer von SORCUS erhältlichen Application Note beschrieben.

Das Auftreten bestimmter Ereignisse kann das Treiberprogramm durch Aufrufen einer ebenfalls dem Treiberprogramm anzugebenden Prozedur eines Echtzeitprogramms melden. Auch für diesen Zweck steht alternativ eine Prozedur innerhalb des Treiberprogramms zur Verfügung, wenn kein Echtzeitprogramm erstellt wird. Diese Prozedur löst einen Service Request (SRQ) auf dem PC aus. Die genaue Ursache des SRQ (z.B. unterbrochene Verbindung) kann dann anhand der zugehörigen Parameterwerte ermittelt werden.

Blockschaltbild



Technische Daten

Parameter	Wert	Einheit
Anzahl der Ethernet-Kanäle	1	-
Anschlußmöglichkeiten (alternativ, per Software wählbar)	Twisted-Pair AUI	- -
Versorgungsspannungen: Für das Modul Für einen externen Transceiver	 +5 +12	 V V
Stromaufnahme: +5 V (typ., nichts angeschlossen): +12 V (begrenzt durch Sicherung):	 80 max. 500	 mA mA
Betriebstemperaturbereich Lagertemperaturbereich	0 bis 60 -55 bis 150	°C °C
Abmessungen (L x B x H)	106 x 45 x 15	mm

Lieferumfang

- Modul M-ETH-1
- 26-poliger Pfostenstecker für Flachbandkabel
- Datenträger mit Treiberprogramm M046TASK.EXE, Hochsprachenbibliotheken, Beschreibung und Software

Zubehör:

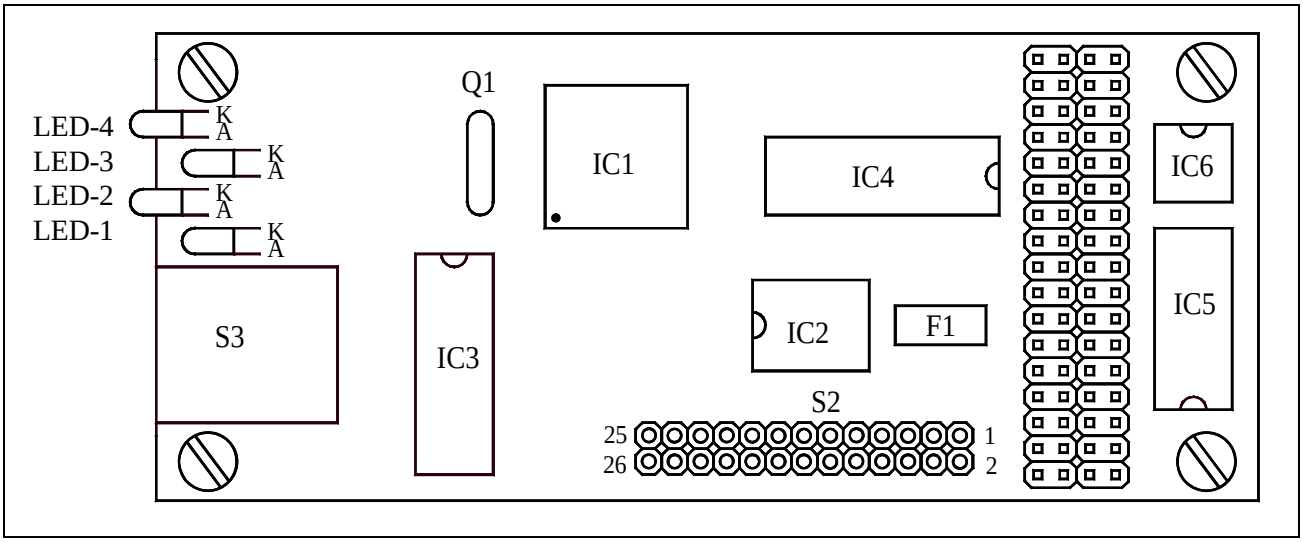
- Kabel mit Modulstecker und Slotblech mit 15-pol. D-Sub-Verbinder für AUI

Konfiguration und Einbau

Vor dem Einbau des Moduls beachten Sie bitte folgende Punkte:

- Überprüfen Sie, ob die werkseitigen Einstellungen im EEPROM für Ihre Anwendung korrekt sind. Ansonsten tragen Sie die notwendigen Änderungen in das EEPROM ein (z.B. mit SNW32).
- Es empfiehlt sich, die Einstellungen zu dokumentieren.

Lageplan



EEPROM-Inhalte

Werkseitig ist bereits eine Konfiguration im EEPROM voreingestellt:

Wort	Binär		Hex	Bedeutung (Kurzinfo)
0	0010 0001	0010 1110	212Eh	Modultyp 46, Rev. A
1	0000 0000	0000 0001	0001h	Initialisierung
2	0000 0000	0000 0000	0000h	Interrupt
3	0000 0000	0000 0000	0000h	Konfiguration
4	xxxx xxxx	xxxx xxxx	xxxxh	Individuelle Adr. Byte 1 und 0
5	xxxx xxxx	xxxx xxxx	xxxxh	Individuelle Adr. Byte 3 und 2
6	xxxx xxxx	xxxx xxxx	xxxxh	Individuelle Adr. Byte 5 und 4
7	0000 0000	0000 0000	0000h	Allgemeine Steuerinformationen
8	0000 0000	0000 0000	0000h	Steuerinformationen Sender
9	0000 0000	0000 0000	0000h	Steuerinformationen Empfänger
10	0000 0000	0000 0000	0000h	Reserviert
...	...	...	...	...
15	0000 0000	0000 0000	0000h	Reserviert

WORT-0: Typ und Version des Moduls (darf nicht geändert werden)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	1	0	0	0	0	1	0	0	1	0	1	1	1	0	WORT-0: Kennung
								0	0	1	0	1	1	1	0	Modultyp: 46 = M-ETH-1
				0	0	0	1									Revision: 1 = A, 2 = B, 3 = C, etc.
			0													Reserviert
0	0	1														Kennung

WORT-1: Initialisierung

In diesem Wort kann eingestellt werden, ob das Modul nach dem Einschalten und bei einem Hardware-Reset entsprechend den Eintragungen im EEPROM initialisiert wird (Bit-0 = 1) oder nicht (Bit-0 = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	WORT-1: Initialisierung (werks. Einst.)
																geändert am: von:
															1	Init nach Hard-Reset: 0=nein, 1=ja
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		Reserviert

Die EEPROM-Inhalte der Worte 2 bis 9 dienen zum Abspeichern einer anwenderspezifischen Modulkonfiguration. Die EEPROM-Inhalte werden nicht direkt den entsprechenden Registern des Moduls zugeordnet, sondern können vom Anwenderprogramm übernommen und zum Einstellen der Konfiguration verwendet werden.

WORT-2: Interrupt

In diesem Wort wird der Interrupt eingestellt, den das Modul auf der Basiskarte auslösen soll. Beim Ausführen der Prozedur AUTO_INIT des Treiberprogramms wird dieses Wort dazu verwendet, um zu erkennen, welches Modul zu diesem Treiberprogramm gehört. Der Interrupt, unter dem das Treiberprogramm installiert wurde, und der in diesem Wort eingestellte Interrupt müssen dabei übereinstimmen.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-2: Interrupt (werks. Einst.)
																geändert am: von:
													0	0	0	Interrupt 0 = IRQ-A 1 = IRQ-B 2 = IRQ-C 3 = IRQ-D 4 = IRQ-E 5 = IRQ-F 6, 7 = Interrupt gesperrt
0	0	0	0	0	0	0	0	0	0	0	0	0				Reserviert

WORT-3: Konfiguration

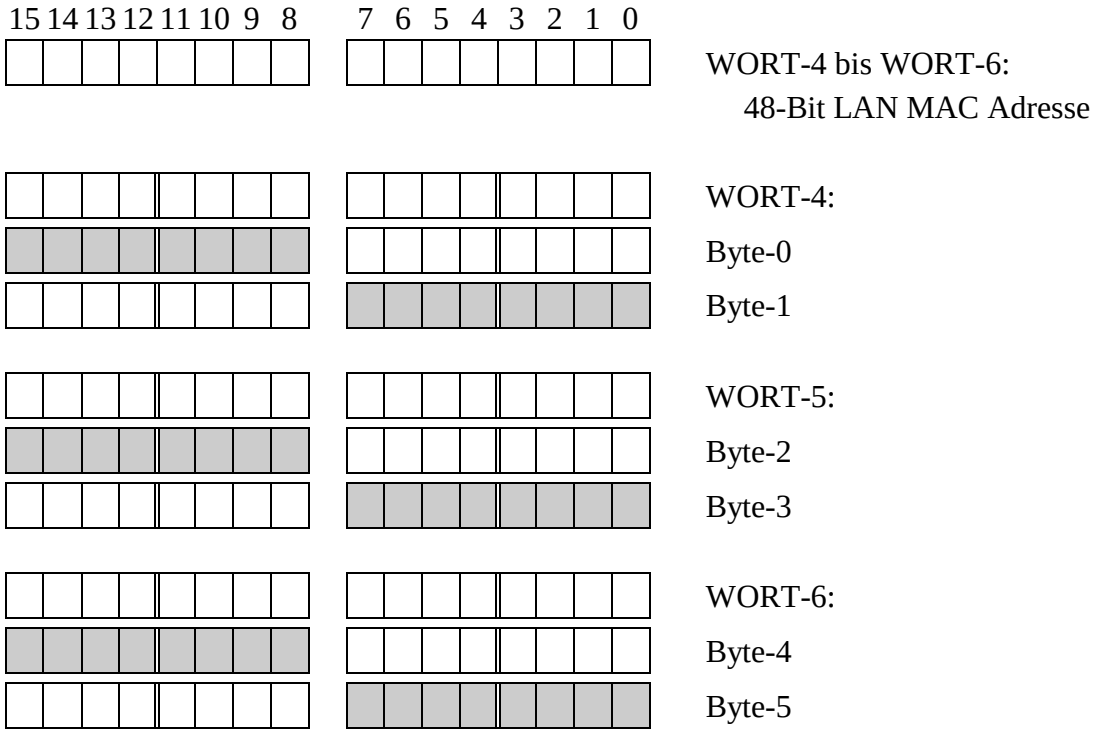
In diesem Wort kann die Hardware-Konfiguration für das Modul festgelegt werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	WORT-3: Konfiguration (werks. Einst.)
																geändert am: von:
									0							Disable Link Test 0 = 10Base-T Link-Test ein 1 = 10Base-T Link-Test aus
								1								16-Bit-Modus 0 = 8-Bit-Modus 1 = 16-Bit-Modus
							0									AUI 0 = Twisted-Pair-Anschluß aktiv 1 = AUI-Anschluß aktiv
0	0	0	0	0	0	0				0	0	0	0	0	0	Reserviert

WORT-4 bis WORT-6: Individuelle Ethernet-Adresse

In diesen Wörtern ist die individuelle Ethernet-Adresse des Moduls abgelegt. Bit-0 von Byte-0 entspricht dabei dem ersten Bit der Adresse auf dem Netzwerk.

! *Diese drei Wörter stellen die weltweit eindeutige, 48-Bit LAN (Local Area Network) MAC (Media Access Control) Adresse des Moduls M-ETH-1 dar.*
Diese werkseitige Einstellung von SORCUS darf nicht verändert werden.



WORT-7: Steuerinformationen

In diesem Wort können allgemeine Steuerinformationen festgelegt werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-7: Steuerinformationen (werks. Einst.)
																geändert am: von:
										0						Transmit Error Enable 0 = Kein SRQ bei Transmit Error 1 = Transmit Error löst SRQ aus
									0							Counter Roll over Enable 0 = Kein SRQ bei Counter Roll Over 1 = Counter Roll Over löst SRQ aus
								0								Link Error Enable 0 = Kein SRQ bei Link Error 1 = Link Error löst SRQ aus
	0															Receive bad 0 = Pakete mit fehlerhaftem CRC verwerfen 1 = Auch Pakete mit fehlerhaftem CRC empfangen
0		0	0	0	0	0	0				0	0	0	0	0	Reserviert

WORT-8: Steuerinformationen für den Sendeteil

In diesem Wort können Steuerinformationen für den Sendeteil festgelegt werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-8: Sendersteuerung
																geändert am: von:
				0												Full Duplex 0 = Half Duplex Mode 1 = Full Duplex
			0													Stop Transmission on SQET Error 0 = Übertragung bei Fehler durch Signalqualität fortsetzen 1 = Übertragung bei Fehler durch Signalqualität abbrechen
0	0	0			0	0	0	0	0	0	0	0	0	0	0	Reserviert

WORT-9: Steuerinformationen für den Empfangsteil

In diesem Wort können Steuerinformationen für den Empfangsteil festgelegt werden.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WORT-9: Empfängersteuerung (werks. Einst.)
																geändert am: von:
														0		Promiscuous Mode 0 = Standard Mode (Standard) 1 = Promiscuous Mode
0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	Reserviert

Steckerbelegung

Das Modul kann auf zwei Arten an ein Ethernet-Netzwerk angeschlossen werden. Über die 8-polige RJ45-Buchse (S3) kann es direkt an ein 10Base-T-Netzwerk angeschlossen werden.

Pin S3	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Signal	TD+	TD-	RD+	n.c.	n.c.	RD-	n.c.	n.c.

Mit dem 26-poligen (2 × 13) Steckverbinder (S2), einem entsprechenden Flachbandkabel und einer 15-poligen D-Sub-Buchse steht die Standard-AUI-Schnittstelle zur Verfügung. Zusätzlich ist auch die 10Base-T-Schnittstelle an diesem Steckverbinder verfügbar.

Pin S2	**Signal**	**Pin S2**	**Signal**
1	GND (Pin 8 / D-Sub 15)	2	n.c. (Pin 15 / D-Sub 15)
3	n.c. (Pin 7 / D-Sub 15)	4	GND (Pin 14 / D-Sub 15)
5	GND (Pin 6 / D-Sub 15)	6	+12 V (Pin 13 / D-Sub 15)
7	RECP (Pin 5 / D-Sub 15)	8	RECN (Pin 12 / D-Sub 15)
9	GND (Pin 4 / D-Sub 15)	10	GND (Pin 11 / D-Sub 15)
11	TXP (Pin 3 / D-Sub 15)	12	TXN (Pin 10 / D-Sub 15)
13	COLP (Pin 2 / D-Sub 15)	14	COLN (Pin 9 / D-Sub 15)
15	GND (Pin 1 / D-Sub 15)	16	Reserviert
17	Reserviert	18	Reserviert
19	n.c. (Pin 8 / RJ45)	20	n.c. (Pin 7 / RJ45)
21	RD- (Pin 6 / RJ45)	22	n.c. (Pin 5 / RJ45)
23	n.c. (Pin 4 / RJ45)	24	RD+ (Pin 3 / RJ45)
25	TD- (Pin 2 / RJ45)	26	TD+ (Pin 1 / RJ45)

Die 10Base-T-Signale liegen auf dem Flachbandkabel in der Reihenfolge vor, wie sie an eine RJ45-Buchse angeschlossen werden sollten (Pin 19 bis Pin 26).

Die Signale für die AUI-Schnittstelle sind auf dem Flachbandkabel so angeordnet, daß die üblicherweise hierfür verwendete 15-polige D-Sub-Buchse direkt auf die Leitungen 1 bis 15 aufgequetscht werden kann. Dabei ist darauf zu achten, daß Pin 8 der Buchse mit Pin 1 des Flachbandkabels verbunden wird.

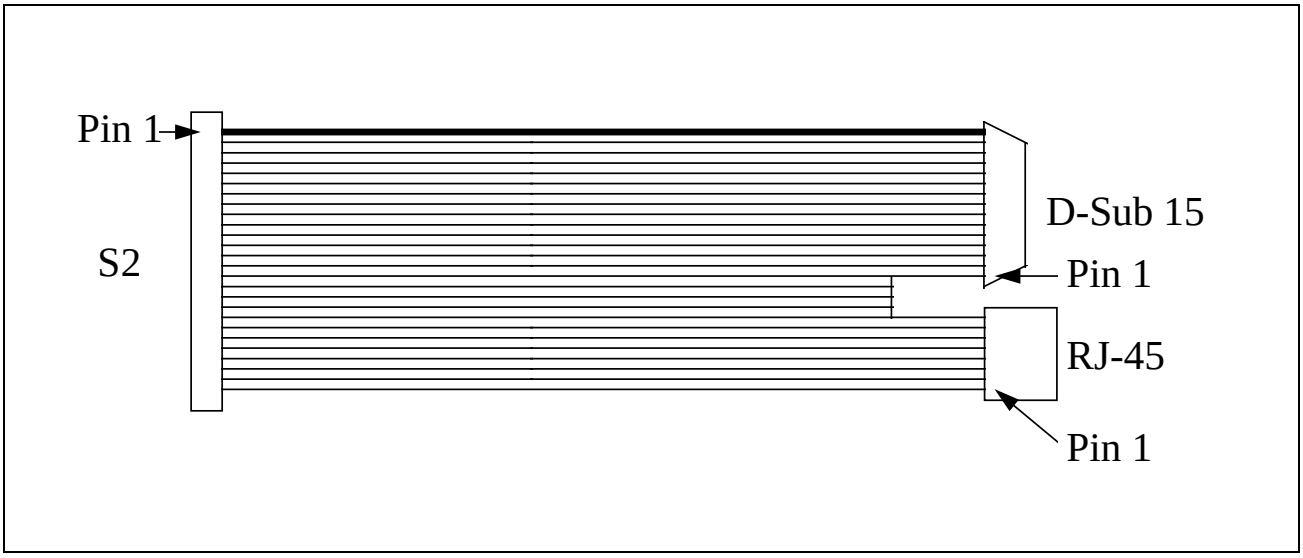


***Abb. 9-2**: Kabel für S2*

Mit der AUI-Schnittstelle kann das Modul durch Aufstecken eines externen Transceivers an alle bekannten Ethernet-Topologien, u.a. 10Base-5 (Yellow cable), 10Base-2 (Cheapernet), 10Base-T (Twisted Pair), 10Base-F (Lichtwellenleiter), angekoppelt werden.

Diagnose-LEDs

Folgende LEDs stehen auf dem Modul für Diagnosezwecke zur Verfügung:

LED-1 (rot)	Leuchtet bei Sendeaktivität
LED-2 (gelb)	Leuchtet bei Empfangsaktivität
LED-3 (grün)	Leuchtet bei intakter 10Base-T-Verbindung
LED-4 (gelb)	Leuchtet bei jedem Zugriff auf ein Register des SMC91C94. Da solche Zugriffe sehr kurz sein können, wird dieses Signal auf 125 ms ausgedehnt.

Hochsprachenbibliothek

Hinweise zum Einbinden der Bibliothek finden Sie in der Einführung im Abschnitt 'Hochsprachenbibliothek'. Der Name der Bibliothek (*libname*) lautet **M046_LIB**, Sie finden Sie im Verzeichnis (*pathname*) **MODULE**. Vor allen anderen Routinen muß die Funktion **m046_bib_startup** einmal aufgerufen werden.

Die Hochsprachenbibliothek benötigt das Treiberprogramm M046TASK.EXE, das vor der ersten Verwendung der Bibliothek für jedes aufgesteckte Modul M-ETH-1 einmal auf der MODULAR-4 Karte installiert sein muß. Zu beachten ist hierbei, daß Task- und Interrupt-Nummer nicht mehrfach verwendet werden.

Vor der Installation des Treiberprogramms sollten die EEPROM-Einträge des Moduls auf ihre Richtigkeit überprüft werden.

Das Treiberprogramm kann entweder mit Hilfe einer INS-Datei und SNW oder unter Verwendung der Funktion **ml8_transfer_and_install** aus der PC-Bibliothek für das MODULAR-4 System auf der MODULAR-4 Karte installiert werden.

Auszug aus einem Beispiel für **ml8_transfer_and_install** in C:

```
isr_task=0x10;   /* Tasknummer */
pgm=0x0601;      /* Programmnummer */
flags = 0x0989;  /* EXE not rel., II-Task, auto_init ausf. */
file = "C:\\SORCUS\\M-ETH-1\\ML8\\RT\\M8P0601\\M046TASK.EXE";
w=ml8_transfer_and_install(file,0,eth_task,pgm,IRQ_A,flags,0);
```

Das Treiberprogramm wird dadurch als II-Task unter einem bestimmten, bei der Installation festgelegten Interrupt installiert (im Beispiel: INT-A).

Beim Installieren wird die Prozedur 1 **(AUTO_INIT)** des Treiberprogramms automatisch aufgerufen. Das Treiberprogramm erkennt anhand der EEPROM-Einstellung für den Interrupt das zugehörige Modul. Dieses Modul wird dann anhand seiner EEPROM-Daten konfiguriert und das Treiberprogramm dadurch betriebsbereit.

Das Modul kann also einfach auf irgendeinen Steckplatz gesteckt werden. Solange der Interrupt beim Installieren des Treiberprogramms mit der EEPROM-Einstellung übereinstimmt, wird vom Treiberprogramm immer das richtige Modul angesprochen.

Das Treiberprogramm unterstützt keine 'Shared Interrupts'. Das heißt, daß es nicht möglich ist, mit zwei Modulen den gleichen Interrupt zu verwenden. Sollen mehrere Ethernet-Module auf einer MODULAR-4 Karte eingesetzt werden, dann müssen sie für verschiedene Interrupts konfiguriert werden, und für jedes Modul muß ein eigenes Treiberprogramm installiert werden. Die Unterscheidung zwischen mehreren

Modulen und den zugehörigen Treiberprogrammen wird dann nur durch die Tasknummer getroffen.

Bei den meisten Funktionen wird mit dem Parameter **m046task** die Tasknummer des anzusprechenden Treiberprogramms angegeben.

Parameter *m046task*: Tasknummer des Treiberprogramms

m046_bib_startup Initialisiere die Modulbibliothek

Pascal FUNCTION m046_bib_startup: WORD;

C ushort m046_bib_startup (void);

Funktion Diese Funktion initialisiert die Modulbibliothek M046_LIB und prüft, ob für jedes aufgesteckte Modul M-ETH-1 ein Treiberprogramm M046TASK.EXE auf der MODULAR-4 Karte installiert ist. Der Rückgabewert enthält die Anzahl der kommunikationsbereiten Treiberprogramme auf der MODULAR-4 Karte.

m046_set_conf_eeprom Setze EEPROM-Konfiguration

Pascal FUNCTION m046_set_conf_eeprom (m046task: WORD): WORD;

C ushort m046_set_conf_eeprom (ushort m046task);

Funktion Diese Funktion setzt die Konfiguration so, wie sie im EEPROM des Moduls angegeben ist. Die Bedeutung der EEPROM-Einträge ist im Abschnitt über die EEPROM-Inhalte beschrieben. Wenn das Setzen der Konfiguration erfolgreich war und das Treiberprogramm betriebsbereit ist, liefert die Funktion den Rückgabewert 0. Anderenfalls wird als Ergebnis 42E0h (Fehlermeldung des Treiberprogramms) zurückgeliefert. Die genaue Fehlerursache kann dann durch Aufrufen der Funktion **m046_error_diag** ermittelt werden.

Die Rückgabewerte dieser Funktion haben dann folgende Bedeutung:

Wert	Bedeutung
8002h	Es wurde kein Modul M-ETH-1 gefunden, das für den beim Installieren des Treiberprogramms angegebenen Interrupt konfiguriert ist.
8003h	Es wurde ein Modul M-ETH-1 gefunden, aber der Ethernet-Controller-Chip konnte nicht identifiziert werden. Das Modul M-ETH-1 ist vermutlich defekt.
8005h	Das Modul ist laut EEPROM für 8 Bit-Zugriffe konfiguriert. Das Treiberprogramm unterstützt z.Zt. aber nur 16 Bit-Zugriffe. Lösung: Bit-7 von WORT-3 der EEPROM-Einträge (16 Bit Modus) auf 1 setzen.

m046_init — Setze Konfiguration

Pascal FUNCTION m046_init (m046task: WORD; VAR m046_config: m046_cfg_struct): WORD;

C ushort m046_init (ushort m046task, m046_cfg_struct *m046_config);

Funktion Diese Funktion setzt die Konfiguration so, wie sie in der übergebenen Datenstruktur vom Typ **m046_cfg_struct** angegeben ist.

Die Bedeutung der Bits in den einzelnen Übergabewerten ist identisch mit den entsprechenden EEPROM-Wörtern.

Konnte die Konfiguration erfolgreich durchgeführt werden, so wird als Ergebnis 0 zurückgeliefert. Anderenfalls wird als Ergebnis 42E0h (Fehlermeldung des Treiberprogramms) zurückgeliefert. Die genaue Fehlerursache kann dann durch Aufrufen der Funktion **m046_error_diag** ermittelt werden. Die Bedeutungen der möglichen Rückgabewerte sind dieselben wie bei **m046_set_conf_eeprom**.

Parameter *m046_config*: Zeiger auf die nachfolgende Datenstruktur des Typs m046_cfg_struct.

Feldbezeichner	Typ	Beschreibung
interrupt	ushort	Interrupt
config	ushort	Konfiguration
EthID1	ushort	Hardware-Adresse (Byte 1 und Byte 0)
EthID2	ushort	Hardware-Adresse (Byte 3 und Byte 2)
EthID3	ushort	Hardware-Adresse (Byte 5 und Byte 4)
control	ushort	Allgemeine Steuerinformationen
tcr	ushort	Steuerinformationen Sendeteil
rcr	ushort	Steuerinformationen Empfangsteil

m046_toggle_promiscuous Schalte Promiscuous Mode um

Pascal PROCEDURE m046_toggle_promiscuous (m046task: WORD): WORD;

C void m046_toggle_promiscuous (ushort m046task);

Funktion Diese Prozedur schaltet den Promiscuous Mode, abhängig vom vorherigen Zustand, ein bzw. aus.

Ist der Promiscuous Mode eingeschaltet, dann werden alle Pakete (unabhängig von ihrer Zieladresse) empfangen. Ist der Promiscuous Mode ausgeschaltet, dann werden nur die Pakete empfangen, deren Zieladresse mit der individuellen Ethernet-Adresse des M-ETH-1 Moduls (üblicherweise aus den EEPROM-Wörtern 4, 5 und 6) übereinstimmt.

Broadcast-Pakete (Pakete mit der Zieladresse FF-FF-FF-FF-FF-FF) werden jedoch in jedem Fall empfangen.

Die Betriebsart des Moduls in Bezug auf den Promiscuous Mode wird in einem Byte (rel. Offset 5) im Parameterbereich des Treiberprogramms angezeigt. TRUE (ungleich 0) bedeutet, daß der Promiscuous Mode eingeschaltet ist.

m046_send_packet Sende Ethernet-Paket

Pascal FUNCTION m046_send_packet (m046task: WORD;
VAR packet_pointer): WORD;

C ushort m046_send_packet(ushort m046task, void *packet_pointer);

Funktion Diese Prozedur sendet ein Ethernet-Paket, dessen Zusammensetzung durch die Datenstruktur beschrieben wird, auf die der übergebene Zeiger **packet_pointer** zeigt. Konnte das Paket erfolgreich zum Senden übergeben werden, so wird als Ergebnis 0 zurückgeliefert. Anderenfalls wird als Ergebnis 42E0h (Fehlermeldung des Treiberprogramms) zurückgeliefert. Die genaue Fehlerursache kann dann durch Aufrufen der Funktion **m046_error_diag** ermittelt werden. Die Bedeutung der möglichen Rückgabewerte ist in folgender Tabelle beschrieben:

Wert	Bedeutung
8006h	Es wurden unzulässige Parameter übergeben.
8008h	Die Länge des zu sendenden Pakets liegt nicht innerhalb des zulässigen Bereichs (64 bis 1518 Byte).
8009h	Der Sendepuffer ist momentan voll. Es muß gewartet werden, bis mindestens ein Paket gesendet wurde. Dadurch wird wieder Platz im Sendepuffer frei.
8010h	Beim Reservieren von Speicher im Sendepuffer ist ein Fehler aufgetreten.

Parameter *packet_pointer*:

Zeiger auf eine Datenstruktur mit folgendem Aufbau:

Feldbezeichner	**Typ**	**Beschreibung**
section_count	ushort	Anzahl der Abschnitte, aus denen das zu sendende Paket besteht. Für jeden Abschnitt folgen jeweils ein Zeiger auf den Anfang des Abschnitts und die Länge des Abschnitts.
section_1_ptr	long	Zeiger auf den Anfang des ersten Abschnitts
section_1_length	ushort	Länge des ersten Abschnitts in Byte.
...	...	...
section_x_ptr section_x_length		Für jeden Abschnitt einen Zeiger auf dessen Anfang und die Länge des jeweiligen Abschnitts.
...	...	...
section_z_ptr	long	Zeiger auf den Anfang des letzten Abschnitts
section_z_length	ushort	Länge des letzten Abschnitts in Byte.

m046_register_RX_function Registriere Empfangsfunktion

Pascal
FUNCTION m046_register_RX_function (m046task: WORD;
VAR m046_rx_reg: m046_reg_rx_struct): WORD;

C
ushort m046_register_RX_function (ushort m046task,
m046_rx_reg_struct *m046_rx_reg);

Funktion
Durch diese Funktion wird eine Empfangsroutine beim Treiberprogramm angemeldet, die aufgerufen werden soll, wenn ein Ethernet-Paket empfangen wird. Übergeben werden die Task- und die Funktionsnummer und die Anzahl der Byte, die dieser Funktion zur Auswertung zur Verfügung stehen sollen, wenn sie aufgerufen wird. Konnte die Funktion fehlerlos registriert werden, so wird als Ergebnis 0 zurückgegeben. Anderenfalls wird als Ergebnis 42e0h (Fehlermeldung des Treiberprogramms) zurückgeliefert. Die genaue Fehlerursache kann dann durch Aufrufen der Funktion **m046_error_diag** ermittelt werden. Die Bedeutung der möglichen Rückgabewerte ist in folgender Tabelle beschrieben:

Wert	Bedeutung
8006h	Es wurden unzulässige Parameter übergeben.
8007h	Die Anzahl der Byte, die der Funktion beim Aufruf zur Auswertung zur Verfügung stehen sollen, ist ungültig.

Im Treiberprogramm ist eine Standard-Empfangsfunktion enthalten, die bei der Installation (durch die **AUTO_INIT**-Prozedur) als Empfangsroutine angemeldet wird.

Hinweis
Das Registrieren einer anderen Empfangsfunktion ist nur in Anwendungen notwendig, die eine eigene Empfangsfunktion zur Verfügung stellen. Diese Empfangsfunktion muß eine Taskfunktion eines Echtzeitprogramms auf der MODULAR-4 Karte sein und bestimmte Regeln einhalten. Der Aufbau einer solchen Empfangsfunktion ist in einer von SORCUS erhältlichen Application Note genauer beschrieben.

Parameter *rx_reg*: Zeiger auf eine Datenstruktur mit folgendem Aufbau:

Feldbezeichner	Typ	Beschreibung
RX_task	ushort	Tasknummer des Echtzeitprogramms, in der die zu registrierende Empfangsfunktion enthalten ist.
RX_function	ushort	Funktionsnummer der zu registrierenden Empfangsfunktion innerhalb von RX_task.
look_ahead_size	ushort	Anzahl Byte vom Anfang des Pakets, die der Empfangsfunktion bei Aufruf vorliegen sollen.

! Hinweis Das Übergeben einer ungültigen Task- oder Funktionsnummer kann zu einem späteren Programmabsturz führen.

m046_receive_packet Lies nächstes Empfangspaket

Pascal FUNCTION m046_receive_packet (m046task: WORD;
VAR packet_pointer): WORD;

C ushort m046_receive_packet (ushort m046task, void *packet_pointer);

! Hinweis Diese Funktion kann nur verwendet werden, wenn die Standard-Empfangsfunktion, die im Treiberprogramm enthalten ist, als Empfangsroutine verwendet wird. Anderenfalls wird eine Fehlermeldung erzeugt.

Funktion Diese Prozedur liest das nächste empfangene Ethernet-Paket. Bei Aufruf dieser Prozedur wird ein Zeiger auf ein Verbund erwartet, der angibt, in welche Speicherbereiche die Daten des Pakets abgelegt werden soll. Die Länge des momentan vorliegenden Empfangspakets kann durch Lesen eines 16-Bit-Wortes (rel. Offset 179) aus dem Parameterbereich des Treiberprogramms ermittelt werden.

Um weitere Pakete empfangen zu können, muß nach dem Lesen eines Pakets der von diesem Paket belegte Speicher möglichst schnell wieder freigegeben werden. Dies geschieht durch Aufrufen der Prozedur **m046_release_RX_packet**.

Konnten die Daten des Paketes fehlerlos übertragen werden, so wird als Ergebnis 0 zurückgeliefert. Anderenfalls wird als Ergebnis eine Warnung oder eine Fehlermeldung zurückgeliefert.

Die Bedeutung der möglichen Rückgabewerte ist in folgender Tabelle beschrieben:

Wert	**Bedeutung**
8011h	Es liegt kein Empfangspaket vor.
8012h	Es ist nicht die Standard-Empfangsfunktion als Empfangsroutine angemeldet.
1001h	Das Empfangspaket wurde nicht vollständig gelesen.

Parameter *packet_pointer*:

Zeiger auf eine Datenstruktur mit folgendem Aufbau:

Feldbezeichner	**Typ**	**Beschreibung**
section_count	ushort	Anzahl der Abschnitte, aus denen das zu lesende Paket besteht. Für jeden Abschnitt folgen jeweils ein Zeiger auf den Anfang des Abschnitts und die Länge des Abschnitts.
section_1_ptr	long	Zeiger auf den Anfang des ersten Abschnitts
section_1_length	ushort	Länge des ersten Abschnitts in Byte.
...	...	...
section_x_ptr section_x_length		Für jeden Abschnitt einen Zeiger auf dessen Anfang und die Länge des jeweiligen Abschnitts.
...	...	...
section_z_ptr	long	Zeiger auf den Anfang des letzten Abschnitts
section_z_length	ushort	Länge des letzten Abschnitts in Byte.

m046_release_RX_packet Lies nächstes Empfangspaket

Pascal PROCEDURE m046_release_RX_packet (m046task: WORD);

C void m046_release_RX_packet (ushort m046task);

Funktion Durch diese Porzedur wird der Empfangspuffer für neue Empfangspakete wieder freigegeben. Diese Prozedur wird nur in Verbindung mit **m046_receive_packet** verwendet.

m046_register_SRQ_procedure Registriere SRQ-Prozedur

Pascal FUNCTION m046_register_SRQ_procedure (m046task: WORD; VAR m046_srq_reg: m046_srq_reg_struct): WORD;

C ushort m046_register_SRQ_procedure (ushort m046task, m046_srq_reg_struct *m046_srq_reg);

Funktion Durch diese Funktion wird eine Prozedur beim Treiberprogramm angemeldet, die als Serviceroutine aufgerufen werden soll, wenn ein bestimmtes Ereignis aufgetreten ist. Die Datenstruktur vom Typ **m046_srq_reg_struct** enthält die Parameter zum Registrieren der Serviceroutine.

Im Treiberprogramm ist eine Standard-Serviceroutine enthalten, die bei der Installation (durch die **AUTO_INIT**-Prozedur) als Serviceroutine angemeldet wird.

Parameter *m046_srq_reg*:

Zeiger auf eine Datenstruktur mit folgendem Aufbau:

Feldbezeichner	Typ	Beschreibung
SRQ_task	ushort	Tasknummer des Echtzeitprogramms, das die zu registrierende Serviceroutine enthält.
SRQ_function	ushort	Funktionsnummer der zu registrierenden Serviceroutine innerhalb von SRQ_task.

! Hinweis Das Übergeben einer ungültigen Task- oder Funktionsnummer kann zu einem späteren Programmabsturz führen.

m046_error_diag — Ermittle Fehlerursache

Pascal FUNCTION m046_error_diag: WORD;

C ushort m046_error_diag (void);

Funktion Durch diese Funktion kann die Ursache eines Fehlers ermittelt werden, der beim Ausführen einer Funktion des Treiberprogramms aufgetreten ist.

Der Rückgabewert ist abhängig von der vorher aufgerufenen Bibliotheksfunktion und ist deshalb bei den einzelnen Funktionen beschrieben.

Notizen

Index

Index zu M-COM-2

(Index zu M-COM-2, Fortsetzung)

Index zu M-COM-2/P

Index zu M-DAS-A

Index zu M-COM-8

(Index zu M-COM-8, Fortsetzung)

Index zu M-IEC-1

Index zu M-DPM-12

(Index zu M-DPM-12, Fortsetzung)

Index zu M-DPS-12

Index zu M-CAN-1

(Index zu M-CAN-2, Fortsetzung)

Index zu M-ETH-1