VME - ISER8

Intelligentes Board
für
10 serielle Schnittstellen

Hardware-Handbuch

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft. esd übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

Sämtliche Rechte an der Dokumentation liegen bei esd. Die Weitergabe an Dritte und Vervielfältigung jeder Art, auch auszugsweise, sind nur mit schriftlicher Genehmigung durch esd gestattet.

esd electronic system design gmbh
Vahrenwalder Str. 205
D-30165 Hannover

Tel: 0511/37298-0
FAX: 0511/37298-68
E-Mail: info@esd-electronics.com
Internet: http://www.esd-electronics.com

Handbuch-Datei: ISER824.MAN **30.08.2000**

Beschriebene Platinenversion	ISER82

Änderungen in den Kapiteln

Die hier aufgeführten Änderungen im Anwenderhandbuch betreffen sowohl Änderungen in der **Firmware**, als auch reine Änderungen in der **Beschreibung** der Sachverhalte.

Kapitel	Änderungen gegenüber Handbuch-Rev. 2.3
-	Aufteilung in Hardware- und Software-Handbuch.

Weitere technische Änderungen vorbehalten.

Inhalt **Seite**

1. Hardware 3
1.1 Blockschaltbild 3
1.2 Technische Daten 4
1.2.1 VMEbus-Prozeßrechner 5
1.2.2 Serielle Prozeß-Schnittstellen 5
1.2.3 Echtzeit-Software 5
1.2.4 Allgemeines 6
1.3 Adressbelegung der ISER8 7
1.4 Konfiguration-Steckbrücken 9
1.4.1 Default-Einstellung 10
1.4.2 Basisadresse und AM (J1,J2,J7,J8) 12
1.4.2.1 Die Address-Modifier AM auf Steckbrücke J1 12
1.4.2.2 Einstellung der Basisadresse mit Steckbrücken J2, J7 und J8 16
1.4.3 Speichergröße und DTACK (J4,J11) 17
1.4.4 RESET, Interrupts, Clocks (J3,J16,J17) 19
1.4.5 Konfigurations-Feld (J12U, J12L) 20
1.4.6 Steckbrücken der 10 seriellen Kanäle (J3, J3A, J6, J9, J13, J14, J26) 21

1.5 Bestückung der SRAM- und EPROM-Bausteine 27
1.5.1 Festlegung der Speichergröße und des Speicherbereichs 27
1.5.2 Zuordnung der Zugriffszeiten der EPROMs und SRAMs zu den Taktfrequenzen der lokalen CPU . . 27
1.6 Interrupt-Verarbeitung 29
1.6.1 Belegung der lokalen Prozessorinterrupts . . . 29
1.6.2 VMEbus-Slave-Interrupt-Logik 30
1.6.2.1 Erzeugung eines VMEbus-Interrupts über den PIT 68230 30
1.6.2.2 Erzeugung eines lokalen Interrupts über den VMEbus (PAL ’GIRL’) 32
1.6.2.3 Programmierung und Rücklesen des Interrupt-Vektors 32
1.6.2.4 Auslösen eines lokalen Interrupts über das PAL GIRL 34
1.7 Watchdog-Schaltung 35
1.8 Externe Erzeugung eines lokalen RESET (VFR) oder ABORT (VFA) 35
1.9 Die seriellen Schnittstellen 37
1.9.1 Übersicht 37
1.9.2 Anschlußschema der seriellen Schnittstellen . . 39
1.9.2.1 Die RS-232 - Schnittstelle 39
1.9.2.2 Die RS-422 - Schnittstelle 40
1.9.2.3 Die RS-485 - Schnittstelle 40
1.9.2.4 Die TTY(20mA) - Schnittstelle 41

Inhalt **Seite**

2. Anhang 43
2.1 Steckerbelegungen 43
2.1.1 VMEbus P1 43
2.1.2 I/O-Stecker P2 44
2.1.3 Übergabemodul I/O-Stecker P2 Phönix FLKM64 o. FLKMS64 45
2.1.4 Belegung einer 9-poligen DSUB-Buchse mit den Signalen der seriellen Kanäle 1...8 46
2.1.5 Belegung einer 9-poligen DSUB-Buchse mit den Signalen der seriellen Kanäle 9..10 46
2.1.6 Serielle Schnittstellen Port 1 und Port 2 (A1, A2) für Terminalanschluß 47
2.2 Frontplatte 49
2.3 Stromlaufpläne 51
2.4 Datenblätter 53

Übersicht

1. Hardware

1.1 Blockschaltbild

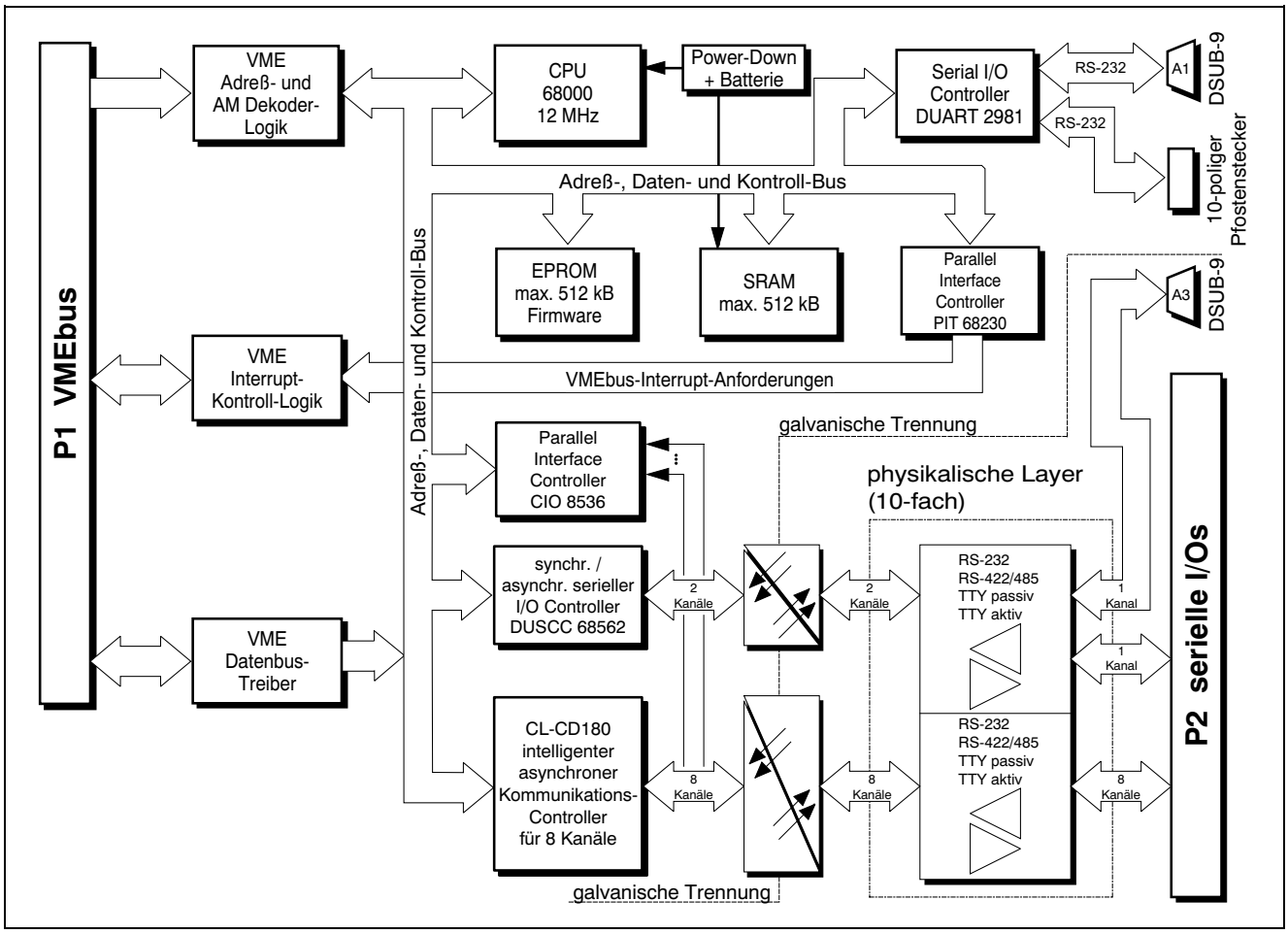


Abb. 1: Blockschaltbild der ISER8

1.2 Technische Daten

Die VMEbus-Baugruppe **VME-ISER8** ist eine **intelligente** Schnittstellen-Karte mit **10 seriellen Prozeß-Schnittstellen** und **zwei zusätzlichen** seriellen Ports für Programmierung und Service.

Die **VME-ISER8** verfügt über eine lokale **MC 68000 CPU** mit **20 MHz** und bis zu **512 kB Shared-SRAM** zur Abarbeitung komplexer Datenübertragungsprotokolle.

Für jeden seriellen Kanal ist der Übertragungsrahmen zwischen **5** und **8** Zeichen plus **optionalem Paritätsbit** (gerade, ungerade, gesetzte oder keine Parität) und **1** oder **2 Stop-Bits** frei wählbar.

Die **maximale Baudrate** bei gleichzeitiger Nutzung aller 10 seriellen Kanäle im asynchronen Betrieb beträgt **38.4 kBaud**. Für jeden Kanal kann **Software**- (XON/XOFF) oder **Hardware-Handshake** gewählt werden.

Als physikalische Layer der seriellen Übertragungsleitungen stehen für jeden Kanal **wahlweise RS-232, RS-422, RS-485 oder TTY-Stromschleife** (aktiv oder passiv) zur Verfügung.

Die Schnittstellen können wie folgt kombiniert werden:

1. **RS-232, TTY aktiv, TTY passiv**

2. **RS-232, TTY passiv, RS-485, RS-422**

Die Kombinationen TTY aktiv und RS-485 bzw. TTY aktiv und RS-422 sind nicht möglich!

Die zusätzlichen Ports für Programmierung und Service sind immer als RS-232-Schnittstellen ausgelegt.

Zwei der 10 Prozeß-Schnittstellen können auch **synchron** betrieben werden. Die Hardware unterstützt hierfür standardmäßig diverse **bitorientierte** (BOP: HDLC, SDLC, X.25, X.75) und **zeichenorientierte-Protokolle** (COP: BISYNC, DDCMP) mit **CRC-Generierung** und **-Erkennung** sowie unterschiedliche **Datencodierungsformate** (z.B. NRZ, NRZI, FM0, FM1, Manchester).

Alle Daten- und Kontrollsignale der 10 Prozeß-Schnittstellen sind über schnelle **Optokoppler** von der VMEbus-Seite **galvanisch getrennt**. Die Spannungsversorgung mit ±12 V erfolgt über DC/DC-Wandler auf der Baugruppe.

Die **Bedienung** erfolgt **über den VMEbus** oder ein **serielles Terminal**.

1.2.1 VMEbus-Prozeßrechner

- 16/32 Bit - Mikroprozessor **MC68000 20MHz**
- **128 kByte** akkugepufferts SRAM (bis zu 512 kByte) und **128 kByte** EPROM (bis zu**256 kByte EPROM**)
- **VMEbus**-Interface A32/D16 und A24/D16 nach **Rev. C**
- **WATCHDOG**-Schaltung (wird von der lokalen Software zur Zeit nicht unterstützt)
- 2 serielle Host-Schnittstellen **RS232C** / 19200 Baud
- bidirektionales, Interrupt-gesteuertes Message-Interface

1.2.2 Serielle Prozeß-Schnittstellen

- **10 asynchrone**, davon **2** auch **synchron** zu betreibende Schnittstellen:

- Physikalische Layer (als Option zu bestellen): **RS-232, RS-422, RS-485, TTY-Stromschleife** (aktiv oder passiv)

- Baudrate (asynchroner Betrieb der Schnittstellen): bis zu **38.4 KBaud** (auf CL-CD180), bis zu **115 kBaud** (nur auf DUSCC 68562) Im synchronen Betrieb des DUSCC sind auch höhere Baudraten möglich.

  Zur Vermeidung von Datenverlusten sollte die Summe der Baudraten aller aktiven Kanäle (10 Rx-Kanäle + 10 Tx-Kanäle = 20 Kanäle) ca. 800 kBaud nicht überschreiten!

- Galvanische Trennung: mit schnellen Optokopplern (tpd ≈ 50 ns)

1.2.3 Echtzeit-Software

- **Echtzeit/Multitasking**-Betriebssystem RTOS-UH

- **Firmware**

- verschiedenste **asynchrone Datenübertragungsprotokolle** wie z.B. **SAE8, TWG, ZM400, SEAB, SINEC, 3964R, MultiDrop** können angeboten werden

- durch **Shared-RAM-Schnittstelle** ist die Implementierung in **unterschiedliche Master-Betriebssysteme** wie z.B. **OS-9, UNIX, PDOS, VxWorks, VRTX32 oder RTOS-UH** einfach zu realisieren; Treiber für **OS-9** können zur Verfügung gestellt werden

1.2.4 Allgemeines

VMEbus - Interface	IEEE 1014 / C.1
Data-Transfermode	SADO32 - Slave mit A32/D16-Zugriff SD16 - Slave mit A24/D16-Zugriff
Temperaturbereich	0...50°C
Luftfeuchtigkeit	max. 90%, nicht kondensierend
Steckverbinder	P1 - DIN 41612-C96 P2 - DIN 41612-C96 A1 - DSub 9 Buchse A2 - 10pol. Pfostenstecker A3 - DSub 9 Buchse
Größe der Platine	160 mm x 233 mm
VMEbus-Einbau	6 HE hoch / 4 TE breit
Gewicht	620 g
Spannungsversorgung	VMEbus P1: 5V +-5% / 2.5A VMEbus P1: +12V +-5% / 1.0A -12V +-5% / 0.1A

Tabelle 1: Allgemeine Daten der ISER8

1.3 Adressbelegung der ISER8

Adress-Bits VME-Adresse On-Board-Adresse 31 24 16 8 0	Board-Adres.	Baustein Kurzbezeichnung	Erläuterung
eeee.eeee.sss0.1111.1111.1111.1111.1111 eeee.eeee.sss0.1100.0000.0000.0000.0000	0FFFFF 0C0000	EPROM-Bank	System-EPROM-Bereich (maximal 256KByte)
eeee.eeee.sss0.1010.xxxx.xxxx.xxxx.xxxx	0A0000	WATCHDOG	Time Out Watchdog in MAX695
eeee.eeee.sss0.1000.0111.x0xx.xxxx.xxx1	087001	VFR	**V**MEbus **F**orced **R**eset
eeee.eeee.sss0.1000.0111.x1xx.xxxx.xxx1	087401	VFA	**V**MEbus **F**orced **A**bort
eeee.eeee.sss0.1000.0110.xxxx.xx11.1111 eeee.eeee.sss0.1000.0110.xxxx.xx00.0001	08603F 086001	PIT	**P**arallel **I**nterface/**T**imer 68230
eeee.eeee.sss0.1000.0101.xxxx.xxxx.x111 eeee.eeee.sss0.1000.0101.xxxx.xxxx.x001	085007 085001	CIO	**C**ounter/Timer and Parallel **I**/**O** Unit Z8536
eeee.eeee.sss0.1000.0100.xxxx.xxx1.1111 eeee.eeee.sss0.1000.0100.xxxx.xxx0.0001	08401F 084001	DUART (Kanal A+B)	**Du**al **A**synchronous **R**eceiver/ **T**ransmitter SCN2681
eeee.eeee.sss0.1000.0011.xxxx.xxxx.xxx1	083001	RTC	**R**eal **T**ime **C**lock
eeee.eeee.sss0.1000.0010.xxxx.xxxx.xxx1	082001	IR-Vector	VMEbus-Slave-Interrupt-Logik
eeee.eeee.sss0.1000.0001.xx01.1111.1111 eeee.eeee.sss0.1000.0001.xx01.0000.0001	0811FF 081101	OCTART (Kanal 1...8)	Intelligent **Oct**al-Channel **A**synchronous **R**eceiver/ **T**ransmitter CL-CD180
eeee.eeee.sss0.1000.0001.xx00.x111.1111 eeee.eeee.sss0.1000.0001.xx00.x000.0001	08107F 081001	DUSCC (Kanal 9+10)	**D**ual **U**niversal **S**erial **C**ommuni- cations **C**ontroller SCN 68562
eeee.eeee.sss0.1000.0000.xxxx.xxxx.xxx1	080001	IRFIRE	VMEbus-Slave-Interrupt-Logik
eeee.eeee.sss0.1000.1111.1111.1111.1111 eeee.eeee.sss0.0100.0000.0000.0000.0000	07FFFF 040000	RAM2	SRAM-Bank 2 (maximaler Adressraum-256KByte)
eeee.eeee.sss0.0001.1111.1111.1111.1111 eeee.eeee.sss0.0001.0000.0000.0000.0000	01FFFF 010000	RAM2	SRAM-Bank 2 (minimaler Adressraum-64KByte)
eeee.eeee.sss0.0011.1111.1111.1111.1111 eeee.eeee.sss0.0000.0000.0000.0000.0000	03FFFF 000000	RAM1	SRAM-Bank1 für Betriebssystem (maximaler Adressraum-256KByte)

eeee.eeee.sss ....Extended-Address-Zugriff VMEbus
(A20 - A31 = CARD-ADDRESS)

sss .....Standard-Address-Zugriff VMEbus
(A20 - A23 = CARD-ADDRESS)

x .....nicht ausgewertete Adressbits

Tabelle 2: Adressbelegung der ISER8

1.4 Konfiguration-Steckbrücken

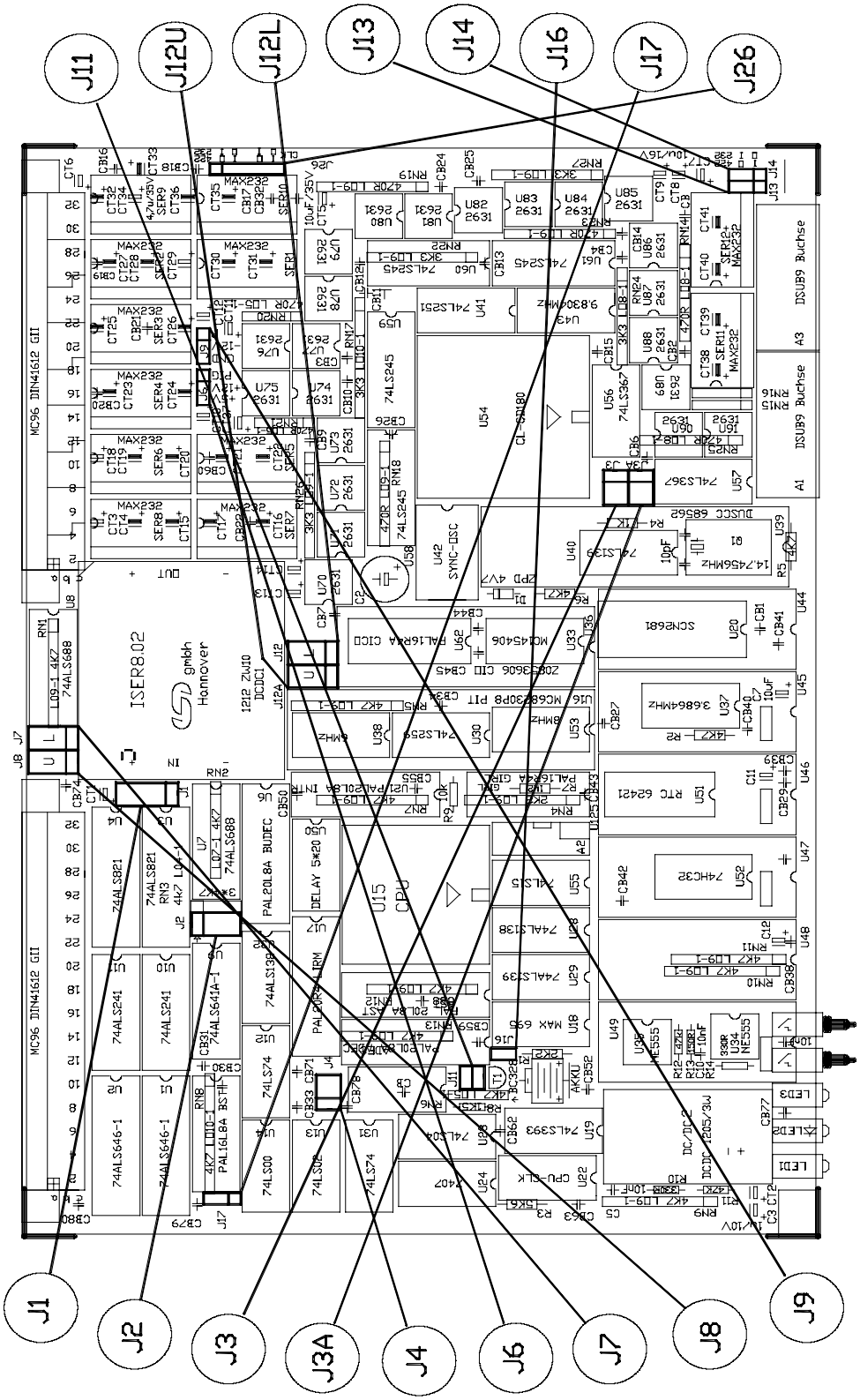


Abb. 2: Lage der Steckbrücken auf der ISER8

1.4.1 Default-Einstellung

Die jeweilige Default-Einstellung (siehe nachfolgende Tabelle) bei Auslieferung der Karte ist in die Steckbrückendarstellungen eingetragen.

Die Anordnung der Steckbrücken ist dem Bestückungsplan zu entnehmen. Eine gesetzte Steckbrücke entspricht dem '0'(Low)-Pegel eines Signals.

Die Steckbrücken sind in folgenden so dargestellt, wie sie der Anwender betrachtet, wenn er die Karte mit den VMEbus-Steckern nach oben vor sich liegen hat.

Default-Einstellung der Steckbrücken J1 bis J14:

Steckbrücke	Funktion	Einstellung
J1	Adress-Modifier AM	AM2 dont care, d.h. Zugriff mit Supervisor- und User-Mode VME-Zugriff A24/D16
J2	Adressen A20..A23	Basisadresse $xx800000
J3	Takt für Seriellen Kanal 10	offen, d.h. kein synchroner Betrieb möglich
J3A	Takt für Seriellen Kanal 9	offen, d.h. kein synchroner Betrieb möglich
J4	DTACK-Verzögerung	min. 140 ns bei VME-Zugriff
J6	Versorgung der Piggybacks	+12V DC/DC entkoppelt an Piggybacks
J7,J8	Adressen A24..A31	offen wegen A24-Mode (J1)
J9	Versorgung der Piggi-Backs	-12V DC/DC entkoppelt an Piggybacks
J11	Speicher-Größe	4 x 32 kByte SRAM (U44, U45, U47, U48) 2 x 64 kByte EPROM (U46/U49)
J12U,J12L	Konfiguration	offen, d.h. Konfig.-Byte=$FF
J13, J14	Sync-Takt Kanal 10 Input/Output	offen, d.h. kein sync.-Takt

Default-Einstellung der Steckbrücken J16 bis J26 :

Steckbrücke	Funktion	Einstellung
J16	Watchdog MAX695	offen (1.2 Sekunden)
J17	VME-RESET/SYSFAIL	kein VME-RESET oder -SYSFAIL bei CPU-Reset
J26	Sync-Takt Kanal 9 Input/Output	offen, d.h. kein sync.-Takt

Tabelle 3: Default-Einstellung der Steckbrücken

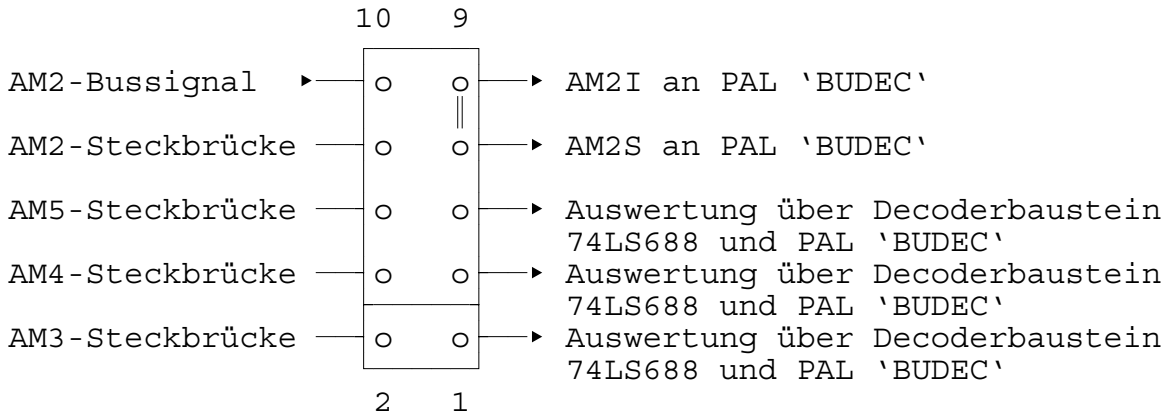
1.4.2 Basisadresse und AM (J1,J2,J7,J8)

1.4.2.1 Die Address-Modifier AM auf Steckbrücke J1:

Die ISER8 kann mit den Zugriffsarten A24/D16 (Standard) oder A32/D16 (Extended) betrieben werden.
In der gezeichneten Default-Stellung ist der Zugriff A24/D16 und AM2 wird ignoriert, so daß sowohl im User- wie auch im Supervisor-Mode adressiert werden kann.

Default-Einstellung: Standart supervisory and nonprivileged data access (A24-Mode)

J1



Die Ausdecodierung der Address-Modifier-Signale (in PAL 'BUDEC' und 74LS688) auf der Karte gestattet nicht alle willkürlichen Kombinationen.

Die Level der AM-Signale, auf die die Karte 'antwortet' sind auf der folgenden Seite aufgeführt.

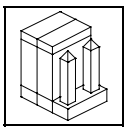
Die Address-Modifier-Signale des VMEbus werden wie folgt ausgewertet:

AM-Signal	zugelassene Level	Kodierung über J1
AM0	1	keine Auswahl möglich
AM1	0	keine Auswahl möglich
AM2	0 1 0 und 1	Steckbrücke auf 7-8 und 9-10 Steckbrücke auf 9-10 Steckbrücke auf 7-9
AM3	0 1	Steckbrücke auf 1-2 Steckbrücke 1-2 offen
AM4, AM5	0,0 1,1	die Karte 'antwortet' nur, wenn AM4 und AM5 den gleichen Level haben:
AM4 AM5	0 0	Steckbrücke auf 3-4 Steckbrücke auf 5-6
AM4 AM5	1 1	Steckbrücke 3-4 offen Steckbrücke 5-6 offen

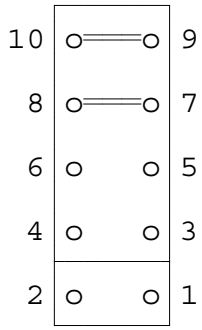
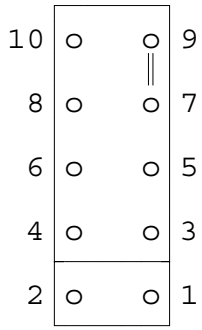
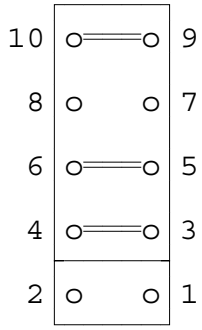
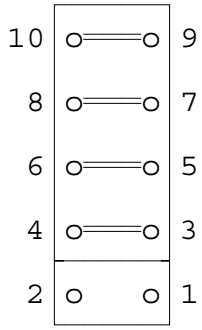
Tabelle 4: Zugelassene Level von Address-Modifier-Signalen

Aus Tabelle 4 ergeben sich folgende sinnvolle Steckbrückenkombinationen und Adressierungsarten:

J1	zugelassene AM-Codes							Adressierungsart
	AM5	AM4	AM3	AM2	AM1	AM0	HEX	
10 ○═══○ 9 8 ○ ○ 7 6 ○ ○ 5 4 ○ ○ 3 2 ○ ○ 1	1	1	1	1	0	1	3D	Standard supervisory data access (A24)



Steckbrückenbelegung

J1	zugelassene AM-Codes							Adressierungsart
	AM5	AM4	AM3	AM2	AM1	AM0	HEX	
	1	1	1	0	0	1	39	Standard nonprivileged data access (A24)
	1 1	1 1	1 1	0 1	0 0	1 1	39 3D	Standard nonprivileged data access (A24) & Standard supervisory data access (A24)
	0	0	1	1	0	1	0D	Extended supervisory data access (A32)
	0	0	1	0	0	1	09	Extended nonprivileged data access (A32)

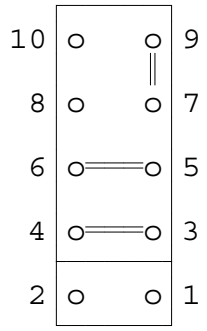
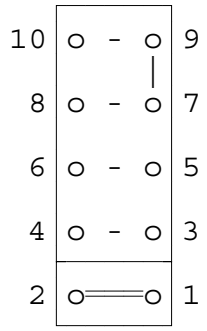
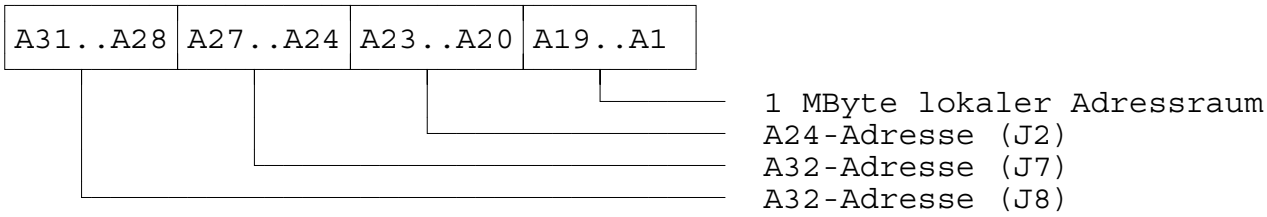
J1	zugelassene AM-Codes							Adressierungsart
	AM5	AM4	AM3	AM2	AM1	AM0	HEX	
	0	0	1	0	0	1	09	Extended nonprivileged data access (A32)
								&
	0	0	1	1	0	1	0D	Extended supervisory data access (A32)
	a	b	0	x	0	1	-	Reserved
	!							Anmerkung:
	a = b							Auf Kombinationen mit AM3 = 0 'antwortet' die Karte. Sie sind jedoch lt. VMEbus-Specification ANSI/IEEE SDT1014-1987 'reserviert'.
	a = 0,1 b = 0,1 x = 0,1							
	-...mögliche Jumper-Stellungen siehe. Tab.4							

Tabelle 5: Auswahl der Adressierungsarten über J1

1.4.2.2 Einstellung der Basisadresse mit Steckbrücken J2, J7 und J8

Die Basisadresse der ISER8 ergibt sich wie folgt:



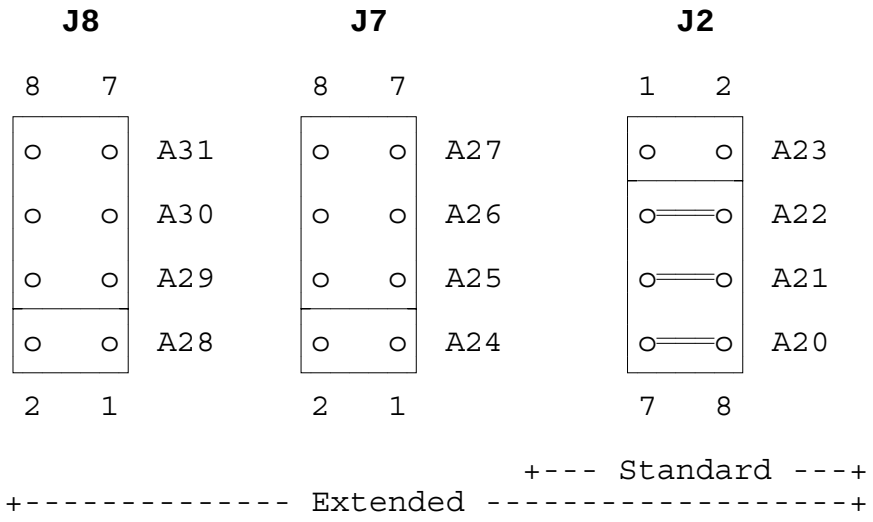
Steckbrückenfelder J7,J8 : Basisadresse für Extended-Addressing AM-Codes (A24 - A31)

Steckbrückenfeld J2 : Basisadresse für Standard-Addressing AM-Codes (A20 - A23)

Anmerkung:

Wenn mit J1 der A24-Zugriff (Standard Addressing) eingestellt ist, werden die Steckbrückenfelder J7 und J8 nicht ausdekodiert. Die Basisadresse wird von A20-A23 gebildet.

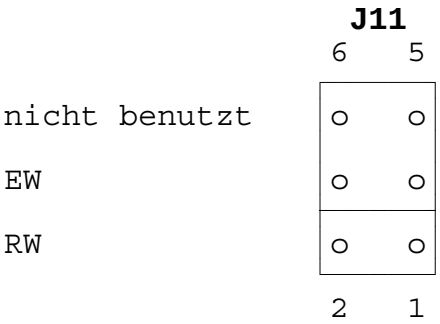
Ist mit J1 der A32-Zugriff (Extended Addressing) eingestellt, werden die Steckbrückenfelder J7, J8 <u>und</u> J2 ausgewertet (A20-A31)!



Die default-Einstellung der Basisadresse der ISER8 beträgt (A24-Mode mit J1 gesteckt) xx800000 Hex.

1.4.3 Speichergröße und DTACK (J4,J11)

Steckbrückenfeld J11: Festlegung des Speicherbereiches der RAM-Bänke 1 und 2 und der EPROM-Bank



Steckbrücke		Speicherbereich	
EW	RW	RAM-Bänke	EPROM-Bank
1	1	Bank 1: (U44, U47) $00000-$0FFFF 64 KByte z.B. 2x 51256 Bank 2: (U45, U48) $10000-$1FFFF 64 KByte z.B. 2x 51256	$C0000-$DFFFF 128 KByte z.B. 2x 27512
Default-Einstellung			
1	0	Bank 1: (U44, U47) $00000-$3FFFF 256 KByte z.B. 2x 581000 Bank 2: (U45, U48) $40000-$7FFFF 256 KByte z.B. 2x 581000	$C0000-$DFFFF 128 KByte z.B. 2x 27512
0	1	Bank 1: (U44, U47) $00000-$0FFFF 64 KByte z.B. 2x 51256 Bank 2: (U45, U48) $10000-$1FFFF 64 KByte z.B. 2x 51256	$C0000-$FFFFF 256 KByte z.B. 2x 27010
0	0	Bank 1: (U44, U47) $00000-$3FFFF 256 KByte z.B. 2x 581000 Bank 2: (U45, U48) $40000-$7FFFF 256 KByte z.B. 2x 581000	$C0000-$FFFFF 256 KByte z.B. 2x 27010

0...Steckbrücke gesetzt
1...Steckbrücke nicht gesetzt

Tabelle 6: Speicherbereichsfestlegung mit J11

Steckbrückenfeld J4 : Verzögerung des lokalen-DTACK-Signals für VMEbus-Zugriffe nach dem Chip-Select einer Baugruppe

Anmerkung:

Die effektive Verzögerung des VMEbus-DTACK-Signals nach dem Address-Strobe des VMEbus läßt sich nicht konkret angeben:

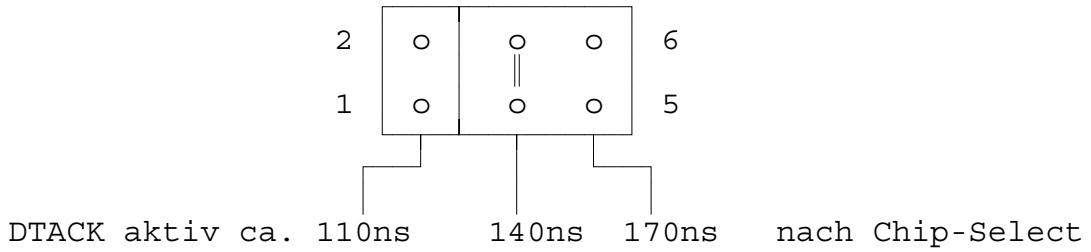
Das VMEbus-DTACK-Signal wird nur gesendet, wenn der VMEbus-Master eine Baugruppe auf der ISER8 adressiert hat. Der Zugriff des VMEbus-Masters auf eine Baugruppe kann aber nur erfolgen, wenn er die Kontrolle über den lokalen Daten/Adressbus der ISER8 besitzt.

Die lokale Busfreigabe wird nach einem Arbitrationsschema ähnlich dem des VMEbus erteilt. Das heißt, hat die lokale CPU die Buskontrolle, muß der VMEbus-Master warten bis diese ihren Zugriffszyklus beendet hat. Erst wenn dies der Fall ist, kann der VMEbus-Master seinen Zyklus einleiten, die Baugruppe selektieren und ein VMEbus-DTACK hervorrufen.

Somit hängt die Verzögerungszeit des VMEbus-DTACK nicht nur von der über J4 eingestellten Zeit ab. Wesentlich für die Dauer der Verzögerungszeit sind auch die Clock-Frequenz der lokalen CPU und der Betriebszustand, in dem sich die lokale CPU zum Zeitpunkt der VMEbus-Anforderung befindet.

Da diese Faktoren von der Bestückungsart der Platine, bzw. vom Zufall abhängen, wird hier nur die Zeit zwischen dem Chip-Select der Speicher oder Peripheribausteine und dem DTACK angegeben:

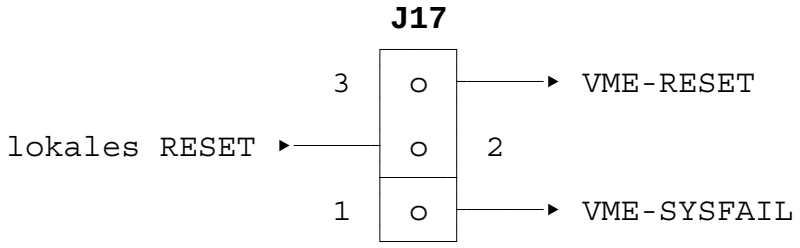
J4



Die DTACK-Verzögerung für Zugriffe der lokalen CPU ist nicht veränderbar.

1.4.4 RESET, Interrupts, Clocks (J3,J16,J17)

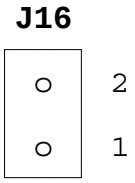
Steckbrückenfeld J17 : Zuordnung des lokalen RESET zum VME-RESET, bzw. zum VME-SYSFAIL



Standardbestückung: Keine Steckbrücke eingesetzt

Steckbrücke J16 : Einstellen der Time-Out-Zeit des Watchdog MAX 695

Jumper nicht gesetzt: RESET nach 1,2 sec
Jumper gesetzt : RESET nach 100 msec



Standardbestückung: Keine Steckbrücke eingesetzt

1.4.5 Konfigurations-Feld (J12U, J12L)

Mit den Steckbrücken J12U und J12L stehen 2 x 4 Bit für anwendungsabhängige Konfigurationen zur Verfügung. Davon werden 3 für den Systemteil (RTOS-Betriebssystem und Treiber) reserviert.

Die Signale sind auf den Port B des Pit gelegt und dort jederzeit auf der Adresse $86017 (als Byte-Zugriff) lesbar.

Eine gesetzte Steckbrücke legt das entsprechende Datenbit auf ’0’.

Steckbrücke J12U

				Datenbit	Funktion	offen	gesteckt
8	o	o	7	D7	reserviert	-	-
6	o	o	5	D6	reserviert	-	-
4	o	o	3	D5	reserviert	-	-
2	o	o	1	D4	frei	-	-

Steckbrücke J12L

				Datenbit	Funktion	offen	gesteckt
8	o	o	7	D3	frei	-	-
6	o	o	5	D2	frei	-	-
4	o	o	3	D1	frei	-	-
2	o	o	1	D0	frei	-	-

Tabelle 7: Zuordnung der Steckbrücken J12L und J12U zu den Bits des Konfigurationsbytes

1.4.6 Steckbrücken der 10 seriellen Kanäle
(J3, J3A, J6, J9, J13, J14, J26)

Steckbrücke J6 und J9: Versorgung der Piggybacks mit +12V/-12V oder mit +5V über DC/DC-Wandler

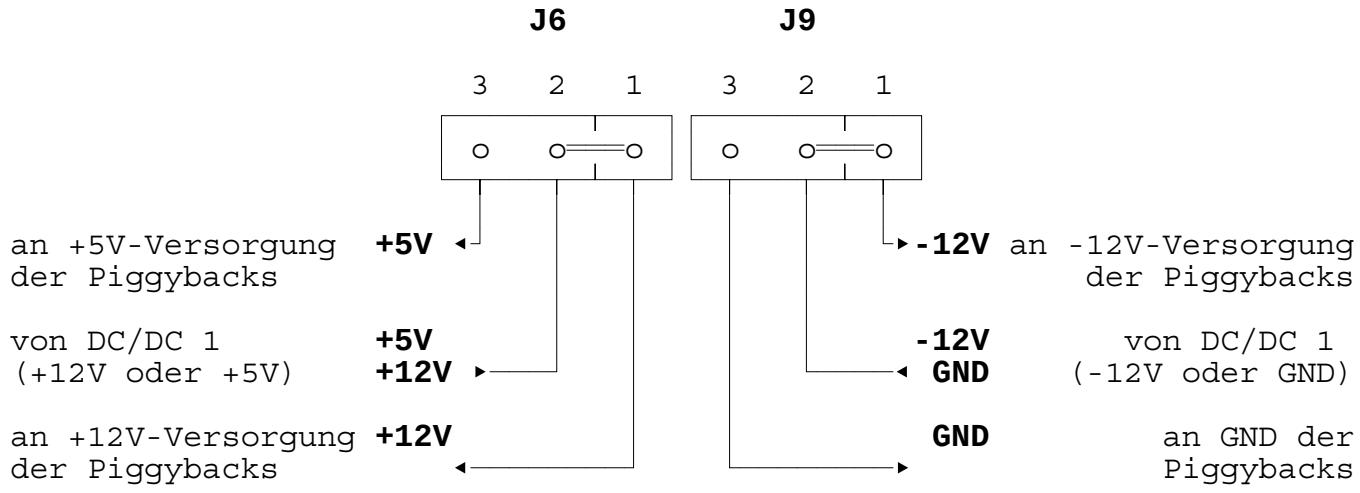
Die Bestückung der Piggybacks der seriellen Kanäle 1 bis 10 kann mit Baugruppen erfolgen, die verschiedene Versorgungsspannungen benötigen.
Aus diesem Grund läßt sich die ISER8 mit verschiedenen DC/DC-Wandlern betreiben.

Standardmäßig sind ein +-12V DC/DC-Wandler für die Piggybacks (DC/DC1) und ein +5V-DC/DC-Wandler für die Optokoppler (DC/DC2) bestückt.

Der DC/DC für die Optokoppler ist immer als +5V-Typ ausgeführt.

Der DC/DC für die Piggybacks wird je nach Wahl der Schnittstellen als +5V- oder +-12V-Typ ausgeführt.

Wird der DC/DC 1 als +5V-Typ bestückt, so ersetzt er den DC/DC 2.
In diesem Fall wird der DC/DC 2 nicht bestückt!



<u>Achtung</u> !

Werden diese Steckbrücken falsch gesetzt, so können die Piggybacks oder die DC-Wandler zerstört werden!

Bitte beachten Sie die umseitig folgende Tabelle für die Stellungen von J6 und J9!

Die Steckbrücken J6 und J9 werden je nach gewünschten Piggyback-Bestückung, also auch in Abhängigkeit von den DC/DC-Wandlern gesetzt:

Piggi-Backs	RS-232	X	X
	RS-485	nicht zulässig	X
	RS-422	nicht zulässig	X
	TTY-passiv	X	X
	TTY-aktiv	X	nicht zulässig
DC-Wandler	DC/DC1	+12V/-12V -Typ	+5V -TYP
	DC/DC2	+5V -Typ	nicht bestückt
Steckbrücken J6 und J9		**J6** 3 2 1 / **J9** 3 2 1 **Default-Stellung**	**J6** 3 2 1 / **J9** 3 2 1

Es dürfen nur diese beiden Stellungen für J6 und J9 gesteckt werden!
Die Kombinationen TTY-aktiv und RS-485 bzw. TTY-aktiv und RS-422 sind nicht möglich.

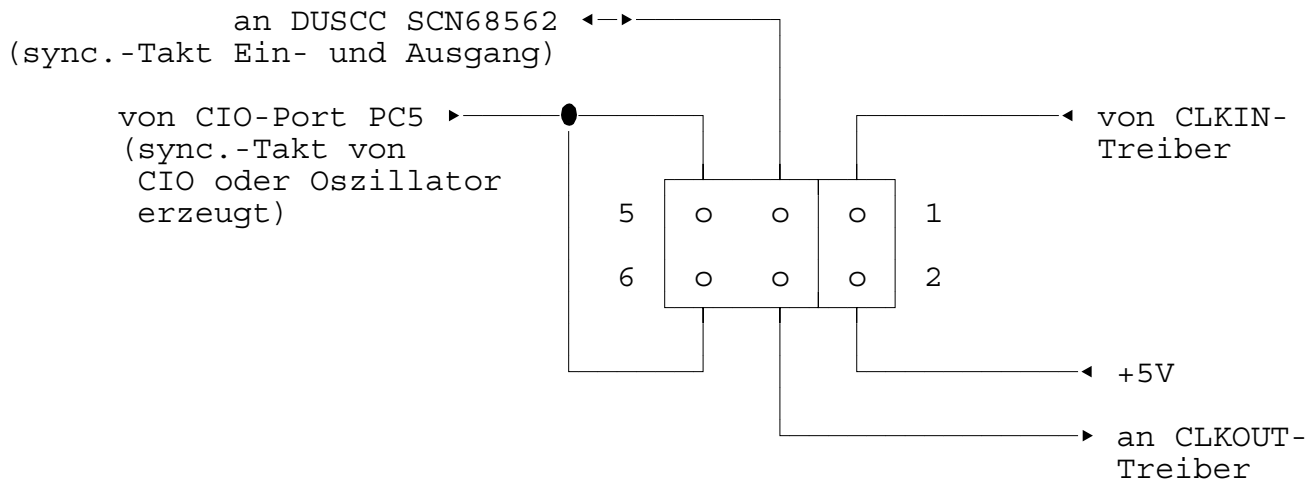
Tabelle 8: Zuordnung der Steckbrücken J6 und J8 zu den eingesetzten Piggybacks und DC/DC-Wandlern

Steckbrückenfeld J3 und J3A: Auswahl und Zuordnung des Takt-Signals für die seriellen Kanäle 9 und 10

Die Steckbrücken J3 und J3A erfüllen die gleiche Funktion.
J3 wird für den Kanal 9 gesetzt und J3A für Kanal 10.

Im synchronen Betrieb entspricht die Bitrate dem Taktsignal. Im asynchronen Betrieb entspricht die Bitrate dem Takt geteilt durch 16.

Belegung der Steckbrücken:



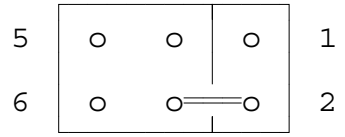
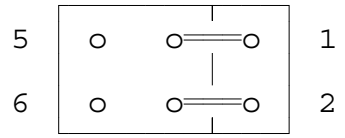
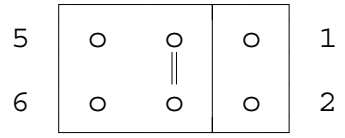
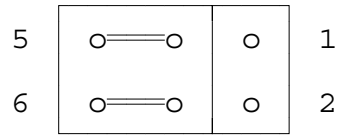
Auswahl des Takt-Signals	**J3 bzw. J3A**
kein synchroner Betrieb **default-Einstellung**	
sync.-Takt von außen zugeführt	
sync.-Takt intern erzeugt von DUSCC SCN68562	
Takt intern erzeugt von CIO Z8536	

Tabelle 9: Auswahl des Takt-Erzeugers über J3 und J3A

Steckbrückenfeld J26: Festlegung der Sync.-Takt Richtung bei den verschiedenen seriellen Übertragungsarten für Kanal 9

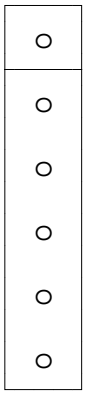
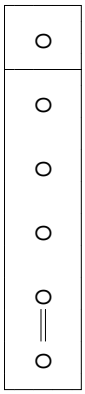
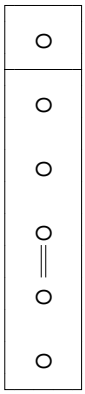
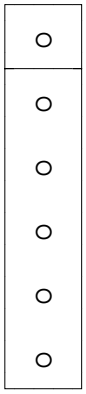
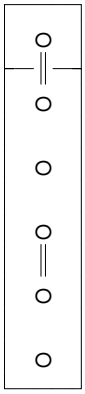
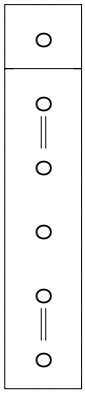
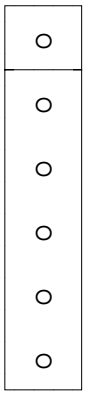
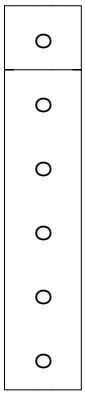
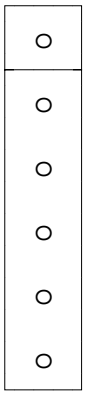
Übertra-gungs-Mode	asynchron: **(default-Einstellung)** Steckbrücke J26 ohne Jumper!	synchron	
		sync.-Takt auf ISER8 erzeugt	sync.-Takt extern erzeugt
RS-232			
RS-485 RS-422			
20mA (TTY) kein sync. Betrieb			

Tabelle 10: Festlegung der sync.-Takt Richtung für Kanal 9

Steckbrückenfelder J13, J14: Festlegung der Sync.-Takt Richtung bei den verschiedenen seriellen Übertragungsarten für Kanal 10

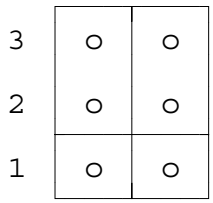
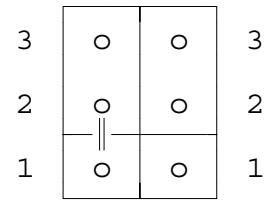
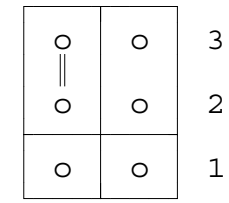
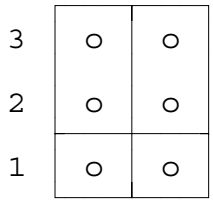
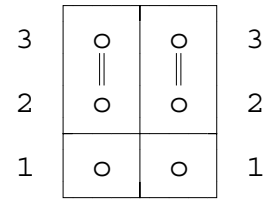
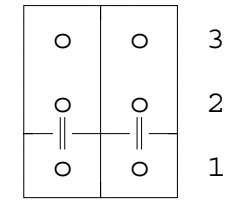
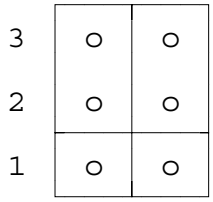
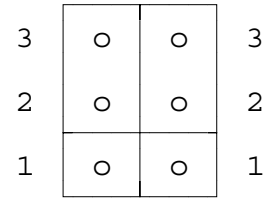
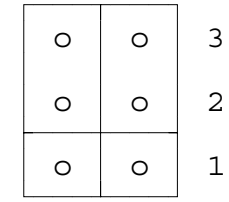
Übertragungs-Mode	asynchron: **(default-Einstelg.)** Steckbrücken J13, J14 immer ohne Jumper!	synchron: sync.-Takt auf ISER8 erzeugt	synchron: sync.-Takt extern erzeugt
RS-232			
RS-485 RS-422			
20mA (TTY) kein sync. Betrieb			

Tabelle 11: Festlegung der sync.-Takt Richtung für Kanal 10

1.5 Bestückung der SRAM- und EPROM-Bausteine

1.5.1 Festlegung der Speichergröße und des Speicherbereichs

Die Auswahl der Speichergröße wird über das Steckbrückenfeld J11 getroffen. Hier wird auch die Lage des Speicherbereiches festgelegt.

Die Beschreibung dieser Steckbrücke und der Steckbrücke J4, mit der die DTACK-Verzögerung der VMEbus-Zugriffe festgelegt wird, sind bereits in dem Abschnitt 'Speichergröße und DTACK' besprochen worden.

1.5.2 Zuordnung der Zugriffszeiten der EPROMs und SRAMs zu den Taktfrequenzen der lokalen CPU

Das lokale DTACK-Signal wird bei Zugriffen der lokalen CPU auf die EPROM- oder SRAM-Bausteine ohne (nennenswerte) Verzögerung ausgelöst. Dadurch wird die Arbeitsgeschwindigkeit der Karte wesentlich erhöht.

Die Kombination von Zugriffszeit der Speicherbausteine und eingesetzter CPU muß so gewählt werden, daß der Datentransfer fehlerfrei abläuft:

CPU/Taktfrequenz	EPROM max. Zugriffszeit	SRAM max. Zugriffszeit
68000/8MHz 68010/8MHz	200ns	150ns
68000/10MHz 68010/10MHz	150ns	100ns
68000/12MHz 68010/12MHz	150ns	100ns
68000/16MHz	120ns	50ns (80ns) 1*)
68000/20MHz	120ns	50ns (80ns) 1*)

*1) Werden für die Adress-Dekodierung schnellere PALs eingesetzt, lassen sich 20 bis 30nsec einsparen.

Tabelle 12: Zuordnung der Zugriffszeit der Speicherbausteine zu der Taktfrequenz der eingesetzten CPU

1.6 Interrupt-Verarbeitung

1.6.1 Belegung der lokalen Prozessorinterrupts

Alle in der unten angegebenen Tabelle aufgeführten Interrupts sind nach einem RESET freigegeben, d.h. liegt eine Interrupt-Bedingung vor, so wird der entsprechende Interrupt ausgelöst!

Die Interrupt-Level der lokalen CPU der ISER8 sind wie folgt belegt:

IR-Level	Baugruppe	Typ	Treiber
7	ABORT	Auto Vektor	RTOS
6	DUART-IRQ	Auto Vektor	RTOS
5	RECEIVER (OCTART)	User Vektor	Firmware
4.1	DUSCC (synchron)	User Vektor	Firmware
4.2	CIO-IRQ	User Vektor	-
3.1	TRANSMIT (OCTART)	User Vektor	Firmware
3.2	PIT-Timer-IRQ	User Vektor	-
2	GIRL-IRQ (Mailbox)	User Vektor	Firmware
1	STATUS (OCTART)	User Vektor	Firmware

Tabelle 13: Belegung der Interrupt-Level der lokalen CPU

Sollten Sie weitere Informationen zu den lokalen Interrupts benötigen, so bitten wir Sie, sich mit uns in Verbindung zu setzen.

1.6.2 VMEbus-Slave-Interrupt-Logik

Die VMEbus-Slave-Interrupt-Logik bietet die Möglichkeit, von der ISER8 (PIT) aus einen Interrupt auf dem VMEbus auszulösen. Außerdem kann vom VMEbus aus ein lokaler Interrupt über das PAL 'GIRL' erzeugt werden.

1.6.2.1 Erzeugung eines VMEbus-Interrupts über den PIT 68230

Neben dem Timer-Interrupt des PIT (TIRQ*) kann der PIT auch auf Port PC5IRQ einen Interrupt auslösen. Dieser Interrupt löst einen VME-IRQ aus und ist zusätzlich auf den CIO-Pin PB3 geführt.
Die Interrupt wird über die PIT-Eingänge H1...H4 ausgelöst.
Bei einem Flankenwechsel dieser Eingangssignale generiert der PIT einen Interrupt und gibt in der Interruptroutine einen entsprechenden Vektor aus. Das weitere Interrupt-Handling des PIT ist im Datenblatt des PIT 68230 näher beschrieben.

Über die PIT-Ports PC0, PC1 und PC2 wird der Level des Interrupts auf dem VMEbus festgelegt:

PIT-Port			VMEbus Interrupt Level
PC2	PC1	PC0	
0	0	0	no IRQ
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

Tabelle 14: VMEbus-Interrupt-Level von PIT-Port C0 bis C2 vorgegeben

Der Zugriff (starten des Interrupts) auf die Signale H1...H4 erfolgt durch das Anlegen entsprechender Adressen:

Signal	Funktion
IRFIRE*	'Chip-Select'
A1	A1=1 -▸ Hx=1 A1=0 -▸ Hx=0 (x = 1...4)
A2,A3,A4	Auswahl von H1...H4

Tabelle 15: Zugriff auf die PIT-Eingänge H1...H4 über Adressbits

Die Eingänge H1...H4 werden bei einem RESET auf '0' gesetzt. In der folgenden Tabelle sind die Adressen aufgeführt, die die entsprechenden H-Signale auf '1' oder '0' setzen.

Adresse	Interrupt-Signal	Pegel
$080011	H4	-▸ 0
$080013	H4	-▸ 1
$080015	H3	-▸ 0
$080017	H3	-▸ 1
$080019	H2	-▸ 0
$08001B	H2	-▸ 1
$08001D	H1	-▸ 0
$08001F	H1	-▸ 1

Tabelle 16: Setzen der PIT-Eingänge H1...H4 über Adressbits

1.6.2.2 Erzeugung eines lokalen Interrupts über den VMEbus (PAL 'GIRL')

Über das PAL GIRL können, gestartet durch das Anlegen entsprechender Adressen, vier verschiedene Interrupts ausgelöst werden. Diese Interrupt-Signale (F0...F3) können, zusammen mit vorhergehend programmierten Datenbits, 4 x 16 verschiedene Interrupt-Vektoren auf den lokalen Datenbus legen. Vom PAL wird beim 'feuern' eines der Interrupt-Eingänge ein lokaler Interrupt auf dem Level 2 der CPU erzeugt.

Im folgenden wird in Form von Tabellen zuerst die Programmierung des Interrupt-Vektors und danach das Starten der Interrupts beschrieben.

1.6.2.3 Programmierung und Rücklesen des Interrupt-Vektors

Adresse des Interrupt-Vektors: **$082001**

Schreiben:

Es können die Datenbits D0 bis D3 frei programmiert werden.

Lesen:

Die Datenbits D0 und D1 geben codiert den *aktuell* anliegenden Interrupt mit der höchsten Priorität an (F0...F3).
Die Datenbits D2 und D3 werden immer als '1' gelesen.
Die Bits D4 bis D7 geben die vorher einprogrammierten Bits D0 bis D3 wieder.

Ablaufbeschreibung:

1. Zustand des Registers nach der fallenden Flanke des Select-Signals IRV* - **Schreiben**

Datenbits

7	6	5	4	3	2	1	0
D3 stored	D2 stored	D1 stored	D0 stored	**D3 input**	**D2 input**	**D1 input**	**D0 input**
don't care	don't care	don't care	don't care	**input**	**input**	**input**	**input**

2. Zustand des Registers nach der steigenden Flanke des Select-Signals IRV* - **Schreiben**

Datenbits

7	6	5	4	3	2	1	0
D3 stored	D2 stored	D1 stored	D0 stored	**D3 input**	**D2 input**	**D1 input**	**D0 input**

shift D0...D3 -▸ D4...D7 and store

3. Zustand des Registers beim Lesen im Interupt-Acknowledge-Zyklus - **Lesen**

Datenbits

7	6	5	4	3	2	1	0
D3 stored	D2 stored	D1 stored	D0 stored	1	1	Fcy transp	Fcx transp

aktueller Interrupt der höchsten Prio:

immer auf '1'

programmierte Datenbits für Interrupt-Vektor

Fcy	Fcx	Interrupt
0	0	F0
0	1	F1
1	0	F2
1	1	F3

1.6.2.4 Auslösen eines lokalen Interrupts über das PAL GIRL

Der lokale Interrupt des GIRL-PALs wird ähnlich dem VMEbus-Interrupt des PIT ausgelöst. Durch das Anlegen von Adressen werden die vier Interrupt-Signale (F0...F3) auf low (‘0‘) oder auf high (‘1‘) gesetzt. Im Gegensatz zum PIT löst das GIRL-PAL jedoch nicht bei einem Flankenwechsel der Interruptsignale einen IRQ aus, sondern bei dem High-Pegel der Signale.

Der Zugriff auf die Signale F0...F3 erfolgt durch das Anlegen entsprechender Adressen:

Signal	Funktion
IRFIRE*	‘Chip-Select‘
A1	A1=1 -▸ Fx=1 A1=0 -▸ Fx=0 (x = 0...3)
A2,A3,A4	Auswahl von F0...F3

Tabelle 17: Zugriff auf die GIRL-PAL-Eingänge F0...F3 über Adressbits

Die Eingänge F0...F3 werden bei einem RESET auf ‘0‘ gesetzt. In der folgenden Tabelle sind die Adressen aufgeführt, die die entsprechenden F-Signale auf ‘1‘ oder ‘0‘ setzen.

Adresse	Interrupt-Signal	Pegel
$080001	F0 **reserviert**	-▸ 0
$080003	F0 **reserviert**	-▸ 1
$080005	F1 **reserviert**	-▸ 0
$080007	F1 **reserviert**	-▸ 1
$080009	F2	-▸ 0
$08000B	F2	-▸ 1
$08000D	F3	-▸ 0
$08000F	F3	-▸ 1

Tabelle 18: Setzen der GIRL-PAL-Eingänge F0...F3 über Adressbits

1.7 Watchdog-Schaltung

Der verwendete MAX695 hat folgende Aufgaben:

1. Erzeugung des RESET-Signals bei POWER-UP

2. Erzeugung des RESET-Signals bei fehlendem Watchdog-Trigger. Der Watchdog muß vorher eingeschaltet werden, falls ein RESET gewünscht wird (siehe ‘RESET, Interrupts, Clocks‘).
   Lokal ist die Funktion immer wirksam.

3. Erzeugung eines RESET-Signals bei Unterschreitung der Betriebsspannungs-Toleranz.

Wird der Watchdog einmal getriggert, muß er innerhalb einer festen Zeit nachgetriggert werden. Sie beträgt bei nicht gestecktem J16 ca. 1.2 Sekunden, bei gesteckter Brücke 0.1 Sekunden.
Nachgetriggert wird der Watchdog durch Schreiben oder Lesen auf das WATCHDOG-Register $A0000 (siehe Mapping: ‘Address-Belegung der ISER8‘).

1.8 Externe Erzeugung eines lokalen RESET (VFR) oder ABORT (VFA)

Der Anwender hat die Möglichkeit, über den VMEbus auf der ISER8 einen Hardware- ABORT oder -RESET zu erzeugen.
Die RESET-Leitung der lokalen CPU wird hierbei über die Hardware auf ‘low‘ (0V) gezogen. Analog dazu wird bei dem ABORT über die Hardware ein Interrupt auf Level 7 der lokalen CPU erzeugt.

Die RESET- und die ABORT-Leitung werden durch das Schreiben eines ‘Dummy-Bytes‘ auf folgende Adressen (relativ zur Basisadresse der ISER8) aktiviert:

RESET: $87001

ABORT: $87401

1.9 Die seriellen Schnittstellen

1.9.1 Übersicht

Die ISER8 besitzt 10 serielle Prozeß-Schnittstellen. Davon werden 8 von dem OCTART CL-CD180 und 2 von dem DUSCC SCN68562 bedient.

(Zwei weitere Ports für Programmierung und Service werden von dem DUART 2681 bedient und sind immer als RS-232-Schnittstellen ausgelegt.)

Die maximale Baudrate bei gleichzeitiger Nutzung aller 10 seriellen Kanäle beträgt 38.4 kBaud. Für jeden Kanal kann Software- (XON/XOFF) oder Hardware-Handshake gewählt werden.

Jeder der 10 Kanäle kann wahlweise als RS-232, RS-422, RS-485 oder TTY-Stromschleife (aktiv oder passiv) betrieben werden. Die verschiedenen Übertragungsarten werden mit Hilfe des RS-232-Treibers MAX232A/MAX243 oder Piggybacks realisiert.

Die Schnittstellen können wie folgt kombiniert werden:

1. **RS-232, TTY aktiv, TTY passiv**

   oder

2. **RS-232, TTY passiv, RS-485, RS-422**

(Die Kombinationen TTY aktiv und RS-485 bzw. TTY aktiv und RS-422 sind nicht möglich.)

Im ersten Fall beträgt die Versorgungsspannung der Treiber +12V/-12V. Im zweiten Fall werden die Bausteine mit +5V betrieben.

Die ISER8 wird standardmäßig mit einem +/-12V-DC/DC-Wandler ausgestattet.

Sollte die Bestückung der ISER8 mit Piggybacks später einmal verändern werden, so darf dies nur im Rahmen der oben angegebenen Variationsmöglichkeiten geschehen. Ansonsten sind Fehlfunktionen oder Schäden auf der ISER8 und den Piggybacks nicht auszuschließen! (Siehe auch Abschn.: Steckbrücken der seriellen Schnittstellen)

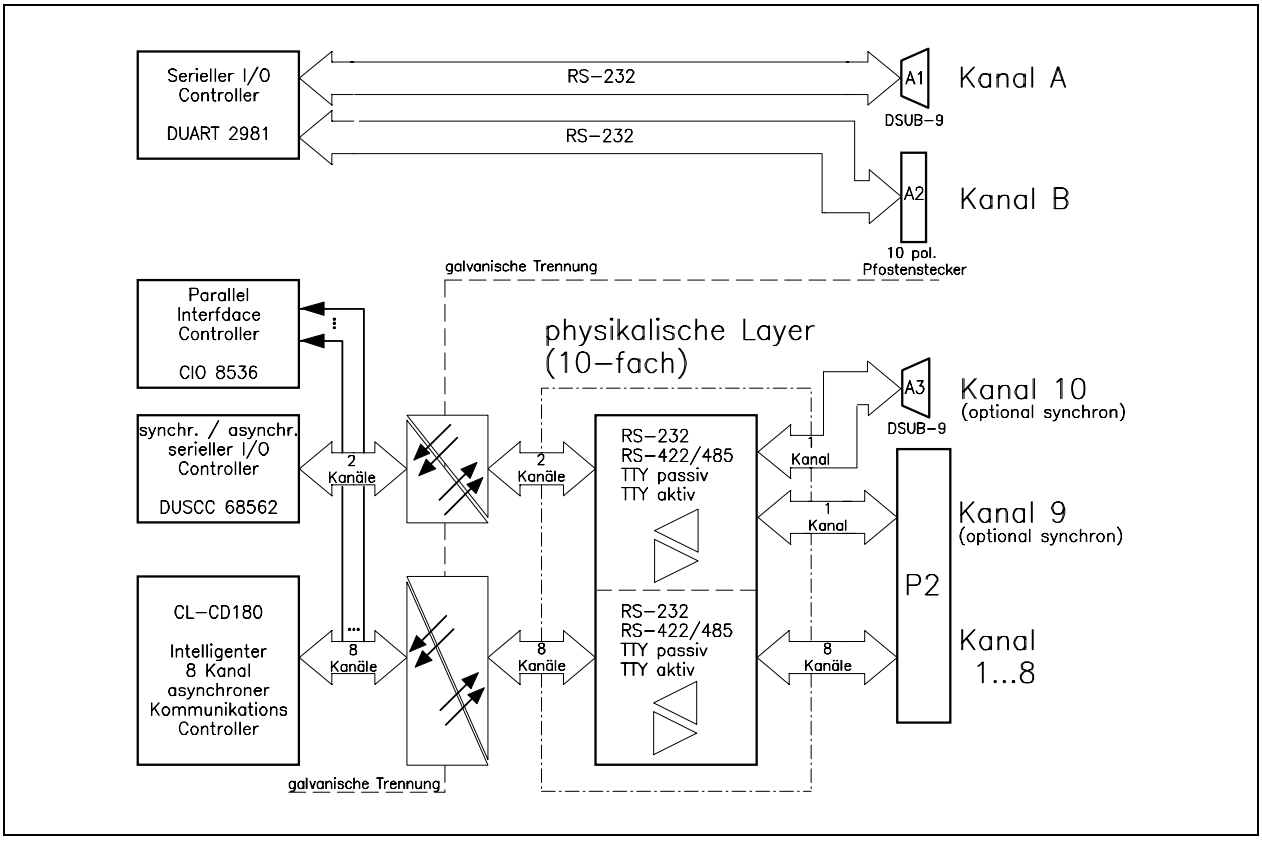


Abb. 3: Zuordnung der seriellen Kanäle zu den Controllern und Steckern

1.9.2 Anschlußschema der seriellen Schnittstellen

Im folgenden wird die Verdrahtung der seriellen Schnittstellen in Bezug auf die Datenrichtung dargestellt. Die Abbildungen sollen die im Anhang (Steckerbelegung) verwendeten Kurzbezeichnungen der Signale erläutern. Im Anhang (Stromlaufpläne) sind außerdem die Schaltpläne der verschiedenen lieferbaren Piggybacks zu finden.

1.9.2.1 Die RS-232 - Schnittstelle

Die Kanäle 1 bis 10 können, mit den Treiberbausteinen MAX232A, MAX243 oder gleichwertigen bestückt, als RS-232-Schnittstellen betrieben werden.

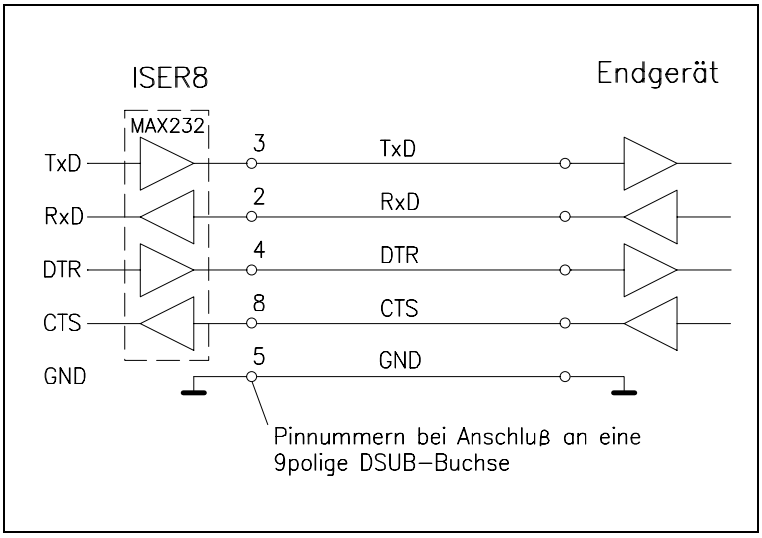


Abb. 4: Anschlußschema für RS-232-Betrieb

1.9.2.2 Die RS-422 - Schnittstelle

Die Kanäle 1 bis 10 können, mit den entsprechenden Piggybacks bestückt, als RS-422-Schnittstellen betrieben werden.

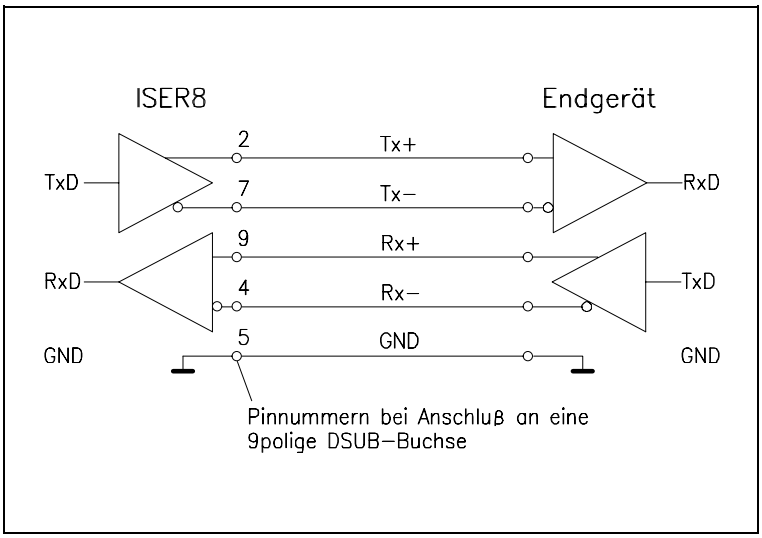


Abb. 5: Anschlußschema für RS-422-Betrieb

1.9.2.3 Die RS-485 - Schnittstelle

Die Kanäle 1 bis 10 können, mit den entsprechenden Piggybacks bestückt, als RS-485-Schnittstellen betrieben werden. Auf dem Piggyback befindet sich ein Abschlußwiderstandsnetzwerk, das durch Setzen von Brücken (z.B. im DSUB-Stecker) aktiviert werden kann.

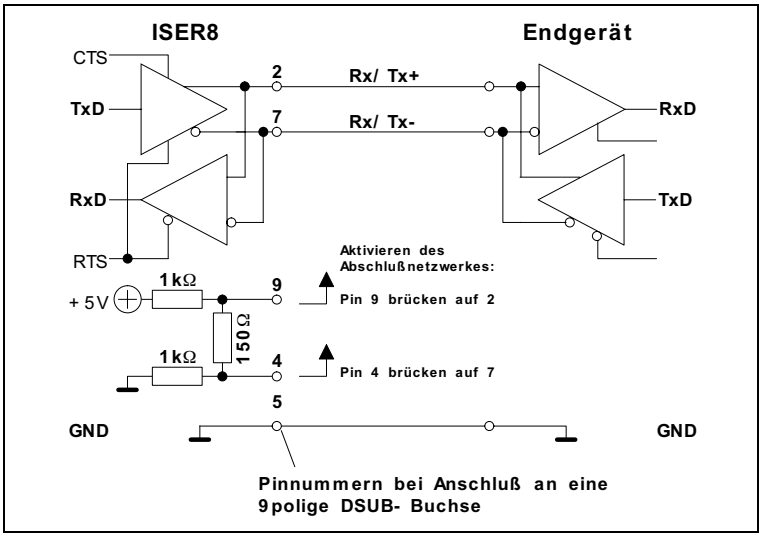


Abb. 6: Anschlußschema für RS-485-Betrieb

1.9.2.4 Die TTY(20mA) - Schnittstelle

Die Kanäle 1 bis 10 können, mit den entsprechenden Piggybacks bestückt, als 'passive' oder auch als 'aktive' TTY-Schnittstellen betrieben werden.

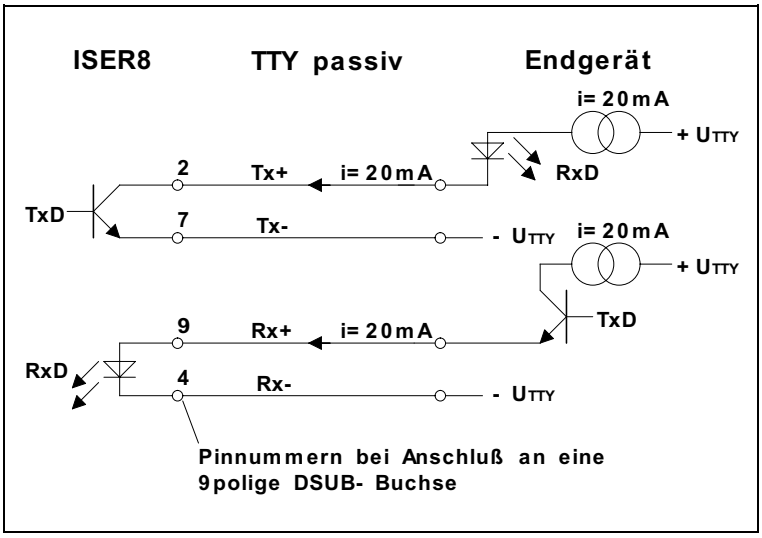


Abb. 7: Anschlußschema für TTY-Betrieb (passiv)

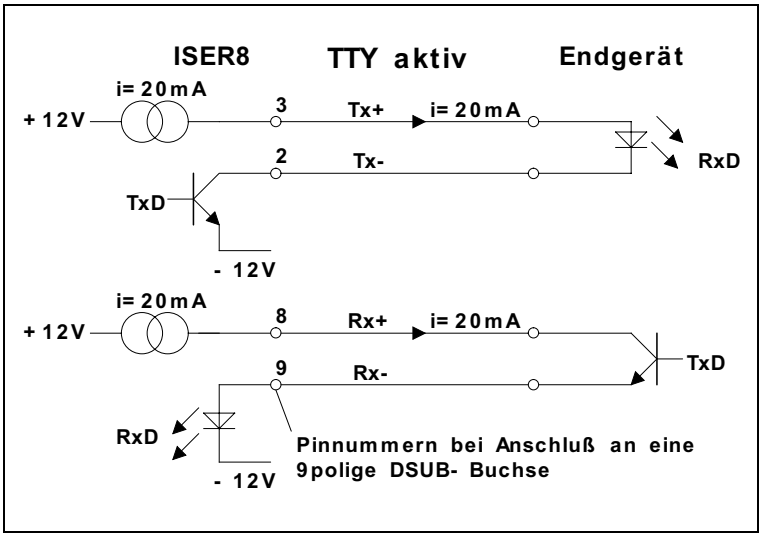


Abb. 8: Anschlußschema für TTY-Betrieb (aktiv)

2. Anhang

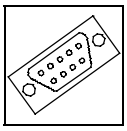
2.1 Steckerbelegungen

2.1.1 VMEbus P1

Pin	Reihe a	Reihe b	Reihe c
1	DATA 0	BBSY*	DATA 8
2	DATA 1	BCLR*	DATA 9
3	DATA 2	ACFAIL*	DATA 10
4	DATA 3	BG0 IN*	DATA 11
5	DATA 4	BG0 OUT*	DATA 12
6	DATA 5	BG1 IN*	DATA 13
7	DATA 6	BG1 OUT*	DATA 14
8	DATA 7	BG2 IN*	DATA 15
9	GND	BG2 OUT*	GND
10	SYSCLOCK	BG3 IN*	SYSFAIL*
11	GND	BG3 OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	ADDR 23
16	DTACK*	AM0	ADDR 22
17	GND	AM1	ADDR 21
18	AS*	AM2	ADDR 20
19	GND	AM3	ADDR 19
20	IACK*	GND	ADDR 18
21	IACKIN*	SERCLOCK	ADDR 17
22	IACKOUT*	SERDAT	ADDR 16
23	AM4	GND	ADDR 15
24	ADDR 7	IRQ7*	ADDR 14
25	ADDR 6	IRQ6*	ADDR 13
26	ADDR 5	IRQ5*	ADDR 12
27	ADDR 4	IRQ4*	ADDR 11
28	ADDR 3	IRQ3*	ADDR 10
29	ADDR 2	IRQ2*	ADDR 9
30	ADDR 1	IRQ1*	ADDR 8
31	- 12V	-	+ 12V
32	+ 5V	+ 5V	+ 5V

Messerleiste nach DIN 41 612 Bauform C 96 / a+b+c

Signal with * : Active Low
Imax per Pin : 1.0 A



Steckerbelegung

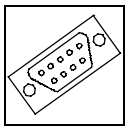
2.1.2 I/O-Stecker P2

Messerleiste nach DIN 41612 Bauform C96 a+c

Pin	Reihe a						Reihe c					Pin
	RS-232	RS-485	RS-422	20mA passiv	20mA aktiv		RS-232	RS-485	RS-422	20mA passiv	20mA aktiv	
1	-	R/Tx-8	Tx-8	Tx-8	(-12V)		RxD8	R/Tx+8	Tx+8	Tx+8	Tx-8	1
2	CTS8	GND	GND	(I2+8)	Rx+8	KANAL 8	TxD8	-	-	(I1+8)	Tx+8	2
3	-	AΩ8+	Rx+8	Rx+8	Rx-8		DTR8	AΩ8-	Rx-8	Rx-8	(-12V)	3
4	RxD7	R/Tx+7	Tx+7	Tx+7	Tx-7		GND8	GND8	GND8	GND8	GND8	4
5	TxD7	-	-	(I1+7)	Tx+7		-	R/Tx-7	Tx-7	Tx-7	(-12V)	5
6	DTR7	AΩ7-	Rx-7	Rx-7	(-12V)	KANAL 7	CTS7	GND	GND	(I2+7)	Rx+7	6
7	GND7	GND7	GND7	GND7	GND7		-	AΩ7+	Rx+7	Rx+7	Rx-7	7
8	-	R/Tx-6	Tx-6	Tx-6	(-12V)		RxD6	R/Tx+6	Tx+6	Tx+6	Tx-6	8
9	CTS6	GND	GND	(I2+6)	Rx+6	KANAL 6	TxD6	-	-	(I1+6)	Tx+6	9
10	-	AΩ6+	Rx+6	Rx+6	Rx-6		DTR6	AΩ6-	Rx-6	Rx-6	(-12V)	10
11	RxD5	R/Tx+5	Tx+5	Tx+5	Tx-5		GND6	GND6	GND6	GND6	GND6	11
12	TxD5	-	-	(I1+5)	Tx+5		-	R/Tx-5	Tx-5	Tx-5	(-12V)	12
13	DTR5	AΩ5-	Rx-5	Rx-5	(-12V)	KANAL 5	CTS5	GND	GND	(I2+5)	Rx+5	13
14	GND5	GND5	GND5	GND5	GND5		-	AΩ5+	Rx+5	Rx+5	Rx-5	14
15	-	R/Tx-4	Tx-4	Tx-4	(-12V)		RxD4	R/Tx+4	Tx+4	Tx+4	Tx-4	15
16	CTS4	GND	GND	(I2+4)	Rx4+	KANAL 4	TxD4	-	-	(I1+4)	Tx+4	16
17	-	AΩ4+	Rx+4	Rx+4	Rx4-		DTR4	AΩ4-	Rx-4	Rx-4	(-12V)	17
18	RxD3	R/Tx+	Tx+3	Tx+3	Tx-3		GND4	GND4	GND4	GND4	GND4	18
19	TxD3	-	-	(I1+3)	Tx+3		-	R/Tx-3	Tx-3	Tx-3	(-12V)	19
20	DTR3	AΩ3-	Rx-3	Rx-3	(-12V)	KANAL 3	CTS3	GND	GND	(I2+3)	Rx+3	20
21	GND3	GND3	GND3	GND3	GND3		-	AΩ3+	Rx+3	Rx+3	Rx-3	21
22	-	R/Tx-2	Tx-2	Tx-2	(-12V)		RxD2	R/Tx+2	Tx+2	Tx+2	Tx-2	22
23	CTS2	GND	GND	(I2+2)	Rx2-	KANAL 2	TxD2	-	-	(I1+2)	Tx+2	23
24	-	AΩ2+	Rx+2	Rx+2	Rx2+		DTR2	AΩ2-	Rx-2	Rx-2	(-12V)	24
25	RxD1	R/Tx+1	Tx+1	Tx+1	Tx-1		GND2	GND2	GND2	GND2	GND2	25
26	TxD1	-	-	(I1+1)	Tx+1		-	R/Tx-1	Tx-1	Tx-1	(-12V)	26
27	DTR1	AΩ1-	Rx-1	Rx-1	(-12V)	KANAL 1	CTS1	GND	GND	(I2+1)	Rx+1	27
28	GND1	GND1	GND1	GND1	GND1		-	AΩ1+	Rx+1	Rx+1	Rx-1	28
29	RxD9	R/Tx+9	Tx+9	Tx+9	Tx-9		CLKI/O	CLK9	CLK9	-	-	29
30	TxD9	-	-	(I1+9)	Tx+9		-	R/Tx-9	Tx-9	Tx-9	(-12V)	30
31	DTR9	AΩ9-	Rx-9	Rx-9	(-12V)	KANAL 9	CTS9	GND	GND	(I2+9)	Rx+9	31
32	GND9	GND9	GND9	GND9	GND9		-	AΩ9+	Rx+9	Rx+9	Rx-9	32

()...Die in Klammern gesetzten Signale sind an die DSUB-Buchse angeschlossen, werden aber für den entsprechenden Betriebs-Mode nicht benötigt.

RS485: Um das Abschlußwiderstandsnetzwerk einer Schnittstelle zu aktivieren, ist jeweils das Signal AΩy+ mit R/Tx+y zu verbinden und AΩy- mit R/Tx-y zu verbinden (y = 1, 2,...9).



Steckerbelegung

2.1.3 Übergabemodul I/O-Stecker P2 Phönix FLKM64 o. FLKMS64

Pin	Reihe a						Reihe c					Pin
	RS-232	RS-485	RS-422	20mA passiv	20mA aktiv		RS-232	RS-485	RS-422	20mA passiv	20mA aktiv	
2	-	R/Tx-8	Tx-8	Tx-8	(-12V)		RxD8	R/Tx+8	Tx+8	Tx+8	Tx-8	1
4	CTS8	GND	GND	(I2+8)	Rx+8	KANAL 8	TxD8	-	-	(I1+8)	Tx+8	3
6	-	AΩ8+	Rx+8	Rx+8	Rx-8		DTR8	AΩ8-	Rx-8	Rx-8	(-12V)	5
8	RxD7	R/Tx+7	Tx+7	Tx+7	Tx-7		GND8	GND8	GND8	GND8	GND8	7
10	TxD7	-	-	(I1+7)	Tx+7		-	R/Tx-7	Tx-7	Tx-7	(-12V)	9
12	DTR7	AΩ7-	Rx-7	Rx-7	(-12V)	KANAL 7	CTS7	GND	GND	(I2+7)	Rx+7	11
14	GND7	GND7	GND7	GND7	GND7		-	AΩ7+	Rx+7	Rx+7	Rx-7	13
16	-	R/Tx-6	Tx-6	Tx-6	(-12V)		RxD6	R/Tx+6	Tx+6	Tx+6	Tx-6	15
18	CTS6	GND	GND	(I2+6)	Rx+6	KANAL 6	TxD6	-	-	(I1+6)	Tx+6	17
20	-	AΩ6+	Rx+6	Rx+6	Rx-6		DTR6	AΩ6-	Rx-6	Rx-6	(-12V)	19
22	RxD5	R/Tx+5	Tx+5	Tx+5	Tx-5		GND6	GND6	GND6	GND6	GND6	21
24	TxD5	-	-	(I1+5)	Tx+5		-	R/Tx-5	Tx-5	Tx-5	(-12V)	23
26	DTR5	AΩ5-	Rx-5	Rx-5	(-12V)	KANAL 5	CTS5	GND	GND	(I2+5)	Rx+5	25
28	GND5	GND5	GND5	GND5	GND5		-	AΩ5+	Rx+5	Rx+5	Rx-5	27
30	-	R/Tx-4	Tx-4	Tx-4	(-12V)		RxD4	R/Tx+4	Tx+4	Tx+4	Tx-4	29
32	CTS4	GND	GND	(I2+4)	Rx4+	KANAL 4	TxD4	-	-	(I1+4)	Tx+4	31
34	-	AΩ4+	Rx+4	Rx+4	Rx4-		DTR4	AΩ4-	Rx-4	Rx-4	(-12V)	33
36	RxD3	R/Tx+	Tx+3	Tx+3	Tx-3		GND4	GND4	GND4	GND4	GND4	35
38	TxD3	-	-	(I1+3)	Tx+3		-	R/Tx-3	Tx-3	Tx-3	(-12V)	37
40	DTR3	AΩ3-	Rx-3	Rx-3	(-12V)	KANAL 3	CTS3	GND	GND	(I2+3)	Rx+3	39
42	GND3	GND3	GND3	GND3	GND3		-	AΩ3+	Rx+3	Rx+3	Rx-3	41
44	-	R/Tx-2	Tx-2	Tx-2	(-12V)		RxD2	R/Tx+2	Tx+2	Tx+2	Tx-2	43
46	CTS2	GND	GND	(I2+2)	Rx2-	KANAL 2	TxD2	-	-	(I1+2)	Tx+2	45
48	-	AΩ2+	Rx+2	Rx+2	Rx2+		DTR2	AΩ2-	Rx-2	Rx-2	(-12V)	47
50	RxD1	R/Tx+1	Tx+1	Tx+1	Tx-1		GND2	GND2	GND2	GND2	GND2	49
52	TxD1	-	-	(I1+1)	Tx+1		-	R/Tx-1	Tx-1	Tx-1	(-12V)	51
54	DTR1	AΩ1-	Rx-1	Rx-1	(-12V)	KANAL 1	CTS1	GND	GND	(I2+1)	Rx+1	53
56	GND1	GND1	GND1	GND1	GND1		-	AΩ1+	Rx+1	Rx+1	Rx-1	55
58	RxD9	R/Tx+9	Tx+9	Tx+9	Tx-9		CLKI/O	CLK9	CLK9	-	-	57
60	TxD9	-	-	(I1+9)	Tx+9		-	R/Tx-9	Tx-9	Tx-9	(-12V)	59
62	DTR9	AΩ9-	Rx-9	Rx-9	(-12V)	KANAL 9	CTS9	GND	GND	(I2+9)	Rx+9	61
64	GND9	GND9	GND9	GND9	GND9		-	AΩ9+	Rx+9	Rx+9	Rx-9	63

()...Die in Klammern gesetzten Signale sind an die DSUB-Buchse angeschlossen, werden aber für den entsprechenden Betriebs-Mode nicht benötigt.

RS485: Um das Abschlußwiderstandsnetzwerk einer Schnittstelle zu aktivieren, ist jeweils das Signal AΩy+ mit R/Tx+y zu verbinden und AΩy- mit R/Tx-y zu verbinden (y = 1, 2,...9).

2.1.4 Belegung einer 9-poligen DSUB-Buchse mit den Signalen der seriellen Kanäle 1...8

Signale					Pins	Signale				
					9-pol DSUB					
RS-232	RS-485	RS-422	20mA passiv	20mA aktiv		20mA aktiv	20mA passiv	RS-422	RS-485	RS-232
-	-	-	-	-	1	-	-	-	-	-
RxD	R/Tx+	Tx+	Tx+	Tx-	2	(-12V)	Tx-	Tx-	R/Tx-	-
TxD	-	-	(I1+)	Tx+	3	Rx+	(I2+)	GND	GND	CTS
DTR	AΩ-	Rx-	Rx-	(-12V)	4	Rx-	Rx+	Rx+	AΩ+	-
GND	GND	GND	GND	GND	5					
					6					
					7					
					8					
					9					

()...Die in Klammern gesetzten Signale sind an die DSUB-Buchse angeschlossen, werden aber für den entsprechenden Betriebs-Mode nicht benötigt.

RS485: Um das Abschlußwiderstandsnetzwerk einer Schnittstelle zu aktivieren, ist jeweils das Signal AΩy+ mit R/Tx+y zu verbinden und AΩy- mit R/Tx-y zu verbinden (y = 1, 2,...9).

2.1.5 Belegung einer 9-poligen DSUB-Buchse mit den Signalen der seriellen Kanäle 9..10 (optional Synchron-Mode)

Signale					Pins	Signale				
					9-pol DSUB					
RS-232	RS-485	RS-422	20mA passiv	20mA aktiv		20mA aktiv	20mA passiv	RS-422	RS-485	RS-232
-	-	-	-	-	1	-	-	CLK	CLK	CLKI/O
RxD	R/Tx+	Tx+	Tx+	Tx-	2	(-12V)	Tx-	Tx-	R/Tx-	-
TxD	-	-	(I1+)	Tx+	3	Rx+	(I2+)	GND	GND	CTS
DTR	AΩ-	Rx-	Rx-	(-12V)	4	Rx-	Rx+	Rx+	AΩ+	-
GND	GND	GND	GND	GND	5					
					6					
					7					
					8					
					9					

()...Die in Klammern gesetzten Signale sind an die DSUB-Buchse angeschlossen, werden aber für den entsprechenden Betriebs-Mode nicht benötigt.

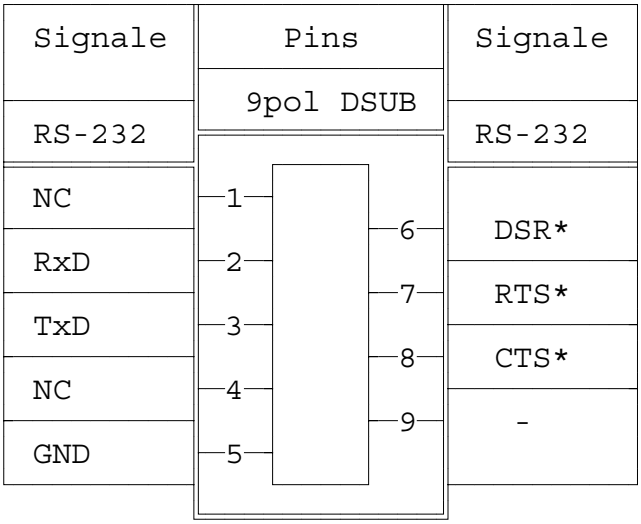
RS485: Um das Abschlußwiderstandsnetzwerk einer Schnittstelle zu aktivieren, ist jeweils das Signal AΩy+ mit R/Tx+y zu verbinden und AΩy- mit R/Tx-y zu verbinden (y = 1, 2,...9).

2.1.6 Serielle Schnittstellen Port 1 und Port 2 (A1, A2) für Terminalanschluß

Die beiden seriellen Schnittstellen entsprechen RS232C. Die Baud-Rate beträgt 9600 Baud.

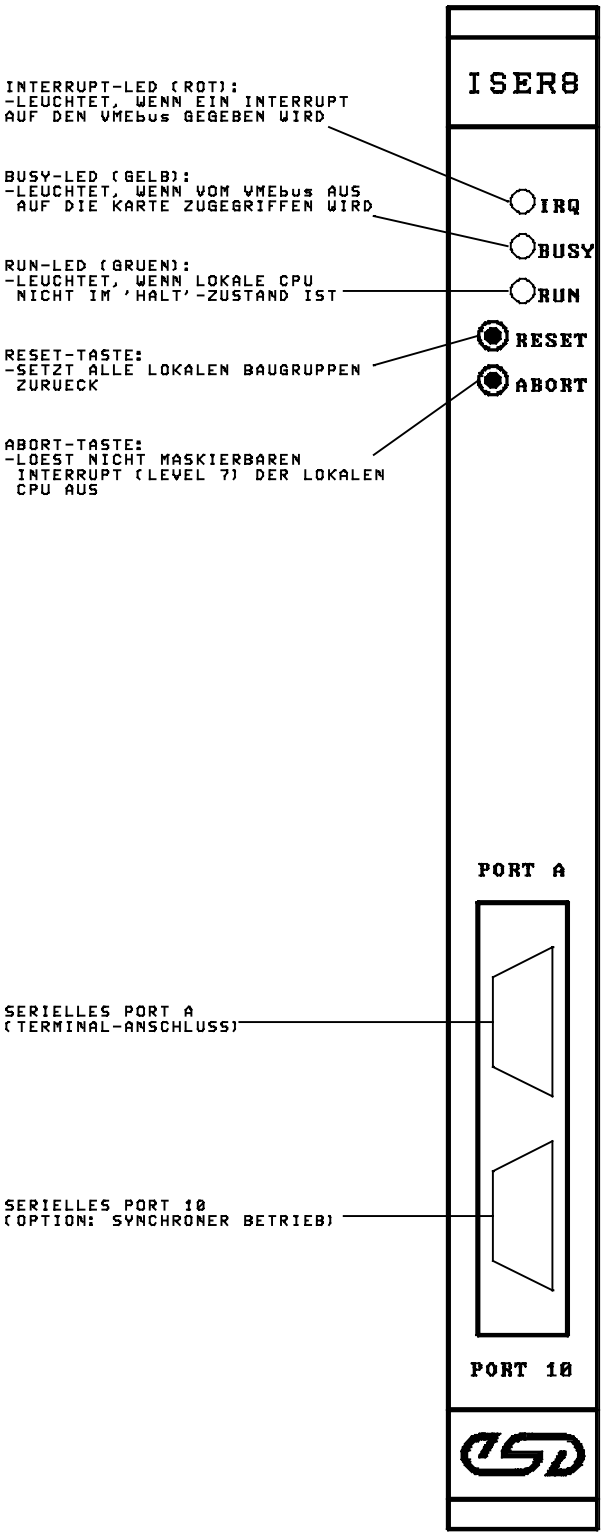
Port 1 ist auf eine 9polige DSub-Buchse geführt (A1), und ist für den Terminal-Anschluß vorgesehen. Die DSUB-Buchse ist in der Frontplatte angebracht (obere DSUB-Buchse).

Port 2 ist nur mit Rxd, TxD und GND belegt und wird auf einen 10poligen Pfostenstecker (A2) geführt, der sich direkt mit einem Flachbandkabel 1:1 wie Port 1 anschließen läßt.



NC...Nicht angeschlossen
*....Diese Signale werden nur von Port 1 (A1) bedient.

2.2 Frontplatte



2.3 Stromlaufpläne

2.4 Datenblätter

VME - ISER8

Intelligentes Board für 10 serielle Schnittstellen

Software-Handbuch

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft. **esd** übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

Sämtliche Rechte an der Dokumentation liegen bei **esd**. Die Weitergabe an Dritte und Vervielfältigung jeder Art, auch auszugsweise, sind nur mit schriftlicher Genehmigung durch **esd** gestattet.

esd electronic system design gmbh
Vahrenwalder Str. 205
30165 Hannover

Tel.: 0511/372 98-0
FAX : 0511/372 98-68
E-Mail: info@esd.electronics.com
Internet: www.esd-electronics.com

Dokument-Datei:	I:\texte\Doku\MANUALS\VME\ISER8\ISER8_01s.ma6
Datum der Druckvorlagen-herstellung:	30.08.2000

Beschriebene Firmware-Version:	isers50b

Änderungen in den Kapiteln

Die hier aufgeführten Änderungen im Anwenderhandbuch betreffen sowohl Änderungen in der **Hardware** als auch reine Änderungen in der **Beschreibung** der Sachverhalte.

Kapitel	Änderungen gegenüber vorhergehender Handbuch-Version
-	Aufteilung in Hardware- und Software-Handbuch.

Weitere technische Änderungen vorbehalten.

Inhalt

1. Einführung 3
1.1 Allgemeines 3
1.2 Kanalübersicht 4
1.2.1 Kanaltypen 4
1.2.2 Aufgaben des VME-Master-Servers 4
1.3 Initialisierung des Systems 5
1.4 Die Kanalstruktur 6
1.4.1 Verkettung der Kanäle 6
1.4.2 Beschreibung der einzelnen Kanalzellen 8
1.5 Verwaltung der Datenkanäle 12
1.5.1 Allgemeines 12
1.5.2 Übersicht der Kanäle mit Verkettung über Pointer 12
1.6 Buffer-Zuteilung 15
1.6.1 Speicherzuteilung über Semaphore 15
1.6.2 Beispiel einer Buffer-Zuteilung 15

2. Kanalbeschreibungen 17
2.1 Beschreibung der Datenkanäle 17
2.2 Beschreibung des Parameter-Kanals 19
2.2.1 Struktur des Parameter-Kanals 19
2.2.2 Beschreibung der Parameter 20
2.2.3 Kommandoübergabe durch den Parameter-Kanal 25
2.3 Beschreibung des Interrupter-Kanals 26
2.3.1 Struktur des Interrupter-Kanals 26
2.3.2 Beschreibung der Zellen des Interrupter-Kanals 28

3. Der lokale VME-ISER-Server 31
3.1 Funktionsweise des lokalen VME-ISER-Servers 31
3.1.1 Output-Kanäle 31
3.1.2 Input-Kanäle 31
3.1.3 Interrupt-Betrieb 32
3.1.4 Time-Out 32
3.1.5 Receive-Error-Mode 34
3.2 Beispiele zum VME-ISER-Server 35
3.2.1 Beispiel: Initialisierung des VMEbus-Masters 35
3.2.2 Beispiel: Data-Output auf Schnittstelle 2 ohne IRQ 36
3.2.3 Beispiel: Data-Input von Schnittstelle 8 37
3.2.4 Beispiel: Parametrierung von Schnittstelle 1 38
3.3 Anwenderprotokolle 39
3.3.1 Funktionsbeschreibung 39
3.3.2 Bedingungen für den Einsatz anwenderspezifischer Rx-Protokolle/Filter 39
3.3.3 Register- und Strukturvereinbarungen 40
3.3.4 Protokolleinbindung für Rx-Betrieb 43

1. Einführung

1.1 Allgemeines

Diese Handbuch ist für die seriellen VMEbus-Schnittstellenkarten **VME-ISER8** und **VME-ISER12** gültig. Die Funktion der beiden Karten ist in weiten Bereichen identisch. Bei Abweichungen ist an den entsprechenden Stellen darauf hingewiesen.
Im den folgenden, für beide Karten zutreffenden Beschreibungen, wird statt der Auflistung beider Namen nur der Begriff 'VME-ISER' verwendet.

Die VME-ISER8 ist eine intelligente Schnittstellenkarte für den VMEbus, die lokal 8 asynchrone und 2 wahlweise synchrone-/asynchrone serielle Schnittstellen verwaltet.

Die VME-ISER12 besitzt die gleiche Anzahl Schnittstellen wie die VME-ISER8. Optional können an die VME-ISER12 zwei Transition-Module des Typs ESP360 angeschlossen werden. In Zusammenhang mit diesen Modulen bietet die VME-ISER12 10 asynchrone und 2 synchrone-/asynchrone serielle Schnittstellen.

Der Anwender arbeitet auf einen linear organisierten Speicher und wird von I/O-Betreuungsaufgaben durch die lokale CPU entlastet.

Der dem Anwender zugängliche Speicher ist in sogenannten Kanälen organisiert, welche aus einem Header und einem Datenbereich bestehen. Die Länge eines Kanals beträgt 256 Byte (128 Byte Netto-Daten), bzw. 1024+128 Byte*(1K Netto-Daten).

Der Aufbau der Header ist für alle vorkommenden Kanaltypen identisch, die verschiedenen Kanäle unterscheiden sich durch entsprechende Einträge in dem Header des Kanals.

Der Status und die Parametrierung der seriellen Schnittstellen ist transparent lesbar, eine Umparametrierung erfolgt synchron zu dem I/O-Transfer.

1.2 Kanalübersicht

1.2.1 Kanaltypen

Das System besteht aus folgenden Kanaltypen:

- den Parameterkanälen	1 Kanal pro serielle Schnittstelle
- den Datenkanälen	1 Receive-Kanal (1KByte)
	1 Transmit-Kanal (1KByte)
	26 Transmit-Kanäle (je 128 Byte)
- dem Interrupter-Kanal	1 Kanal pro Karte

Kanäle sind Softwarestrukturen, die untereinander durch Pointer verkettet sind. Der 'ROOT-Pointer', sowie eine 'Card-Id' befinden sich auf festen Adressen.

1.2.2 Aufgaben des VME-Master-Servers

Der VME-Master-Server für die seriellen Schnittstellen muß im wesentlichen folgende Aufgaben erfüllen:

- Suchen eines freien Kanals und den Kanal belegen

- Eintragen des Übertragungsmodes

- (Datentransfer zum VME-ISER-Speicher für Sendebetrieb)

- Aktivierung des Slave-Servers (lokale Interrupt-Auslösung)

- Polling auf 'ready' oder Reaktivierung durch VME-Interrupt

- (Datentransfer vom VME-ISER-Speicher für Empfangsbetrieb)

- Kanal freigeben

1.3 Initialisierung des Systems

Alle im weiteren aufgeführten Adressen sind relativ zur Karten-Basisadresse angegeben und müssen von der VME-Master-CPU entsprechend adressiert werden.

Nach einem System_Reset initialisiert die lokale-CPU ihren lokalen Speicher und baut die Kanal-Kette neu auf. Dies kann je nach Speichergröße bis zu 2 sec in Anspruch nehmen. Die Master-CPU sollte nach einem Neustart folgende Einträge überprüfen:

- Lesezugriff auf die Basisadresse der Slave-Karte.
  Antwortet die Karte mit einem 'DTACK-Signal', so ist die Karte physikalisch auf der entsprechenden Adresse vorhanden; erfolgt ein BUSERROR (z.B. über Time-Out) so ist die Karte nicht vorhanden >> Abbruch der Initialisierung.

- Überprüfen der Adresse ***CPUID*** = `$0998` auf: `$49534552.L`
  Die lokale CPU muß hier in ASCII: `"ISER" ==($49,$53,$45,$52)` eingetragen haben.

- Überprüfen der Adresse ***ANCHOR*** = `$099C` auf ungleich `$0`
  Die lokale CPU trägt hier den ROOT-Pointer auf die Buffer-Struktur ein (Default: `$00008000.L`)

Die lokale CPU hat nun die im weiteren beschrieben Buffer-Struktur aufgebaut, mit der jetzt eine Kommunikation mit der Master-CPU erfolgen kann.

1.4 Die Kanalstruktur

1.4.1 Verkettung der Kanäle

Alle Kanäle sind durch Pointer miteinander verkettet, wobei zwischen einer reinen Speicherverkettung und einer Suchverkettung unterschieden werden muß.
Die Speicherverkettung verbindet alle vorkommenden Kanäle miteinander, während die Suchverkettung nur die der entsprechenden Schnittstelle zugeordneten Kanäle verbindet.

Speicherverkettung:

Sequentielle Verkettung
Der Root-Pointer auf den ersten vorhandenen Kanal steht als Langwort auf der Adresse ***ANCHOR*** =`$0099C`, der Pointer auf den nächsten Kanal (forpointer) steht als Langwort jeweils in der Zelle ***iofor*** des Kanal-Headers. Der Forpointer des letzten Kanals zeigt wieder zurück auf den ersten Kanal.
Als Default wird ***ANCHOR*** mit `$00008000` versorgt. Alle in den Tabellen angegebenen Adresse beziehen sich auf diese Basis, sind jedoch frei verschieblich.
Die Länge eines Kanals beträgt defaultmäßig 256 Byte und teilt sich auf zu 128 Byte Header und 128 Byte Datenbereich.

Sternförmige Verkettung (ab Software-Rev. iser 50b)
Die sternförmige Verkettung soll die Ermittlung der Adressen der einzelnen Kanäle beschleunigen. Im Interrupt-Kanal sind aufeinanderfolgend die Adressen aller Parameter-Kanäle zu finden. In jedem Parameter-Kanal sind die Adressen des zugehörigen Tx- und des Rx-Kanals abgelegt.

Anmerkung:
Die sequentielle Verkettung und die sternförmige Verkettung stehen beide zur Verfügung und können alternativ genutzt werden.

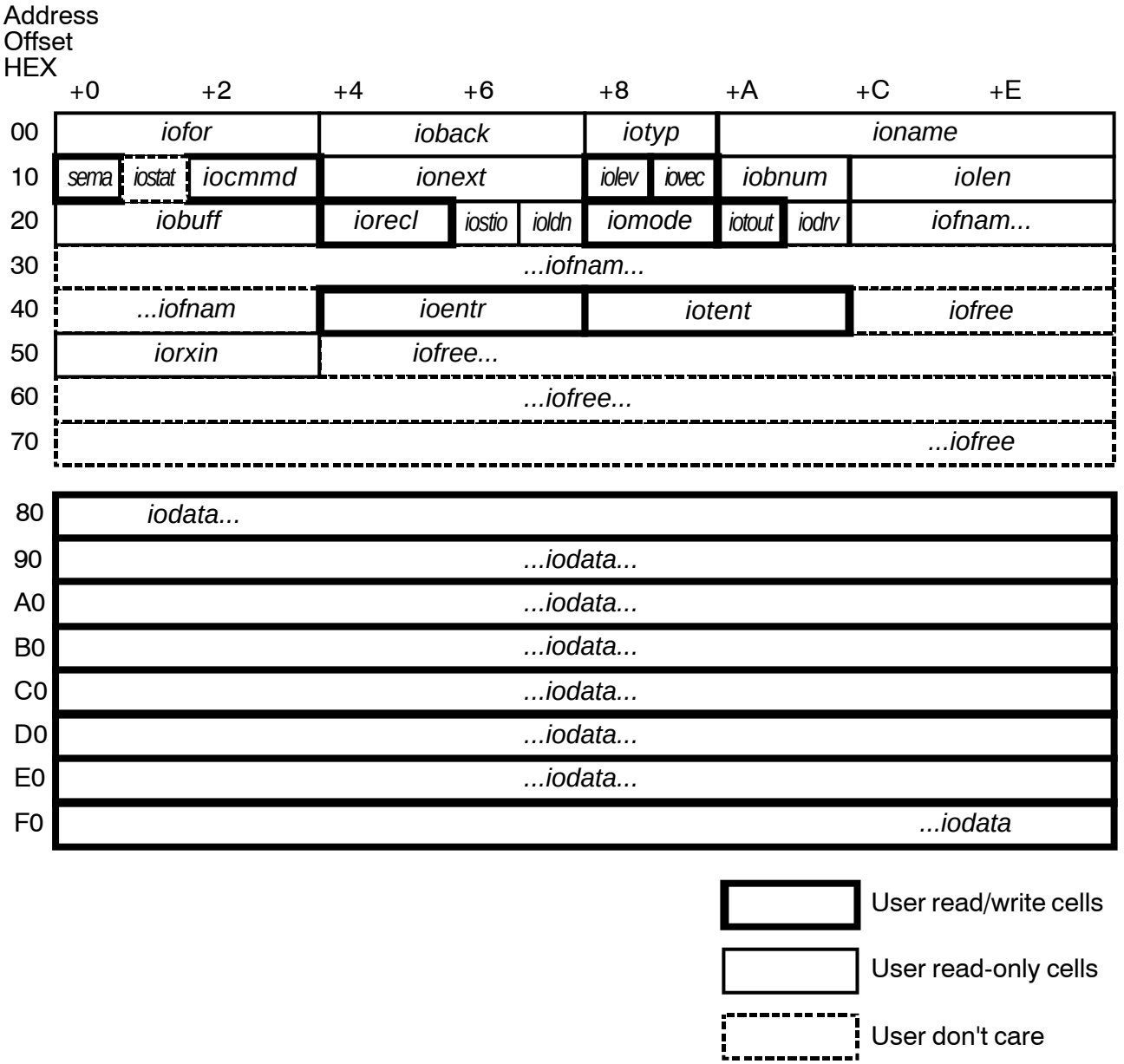


Tabelle 1.4.1: Interne Kanalstruktur mit READ/WRITE-Zuordnung der Zellen

1.4.2 Beschreibung der einzelnen Kanalzellen

Zusammenfassung der Kanalzellen

Name	Offset [HEX]	Organisation	Kurzbeschreibung	Default/Preset
iofor	00	Langwort	Pointer auf nächsten Kanal	
ioback	04	Langwort	nicht benutzt,	$0000000
iotyp	08	Wort	Kanaltyp (s.u.)	
ioname	0A	6 Byte ASCII	Kanal-Identifier als Text	
iosema	10	Byte	Kanalsemaphore,	Preset: $00
iostat	11	Byte	Kanalstatus,	Preset: $00
iocmmd	12	Wort	Kanalkommando,	Preset: $0000
ionext	14	Langwort	Such-Pointer auf nächsten Kanal	
ioilev	18	Byte	VME-Irq-Level für Slave-Irq	
ioivec	19	Byte	VME-Irq-Vektor für Slave-Irq	
iobnum	1A	Wort	Nummer des jeweiligen Kanaltyps	
iolen	1C	Langwort	Länge des Datenbereichs	
iobuff	20	Langwort	Pointer auf Datenbereich	
iorecl	24	Wort	Anzahl Daten im Datenbereich	
iostio	26	Byte	I/O-Status	Default: $00
ioldn	27	Byte	Schnittstellen-Nr. 1 ... 10	
iomode	28	Wort	Sende-/Empfangsmode	
iotout	2A	Byte	Time-Out	
iodrv	2B	Byte	reserviert	
iofnam	2C ... 43	ASCII	reserviert	Default: $0000
ioentr	44	Langwort	Pointer auf User-Protokoll (nur Parameter-Kanal)	
iotent	48	Langwort	reserviert für Tx-Server	
iofree	4C ... 7F		reserviert	Default: $0000
iorxln	50	Wort	Anzahl empfangener Daten	
iodata	80 ... FF	Byte	Datenbereich (128 Byte-Kanäle)	
	80 ... 47F	Byte	Datenbereich (1 KByte-Kanäle)	

Tabelle 1.4.2: Kurzbeschreibung der Kanalzellen

Erläuterung der einzelnen Zellen

iofor dient zur Speicher-Verkettung der Kanäle. *iofor* zeigt immer auf die Anfangsadresse des nächsten Kanals, *iofor* des letzten Kanals zeigt wieder auf den ersten Kanal.

ioback zeigt auf die Anfangsadresse des vorangegangenen Kanals.

iotyp ist der Kanal-Identifier und unterscheidet folgende Kanaltypen:
- `$FFFF` Interrupter-Kanal
- `$000C` Parameter-Kanal
- `$0014` Default-Kanal (not used)
- `$0018` Buffer
- `$001C` Buffer-Kanal (not used)
- `$0114` Tx-Buffer Long
- `$0214` Rx-Buffer Long

ioname enthält den Kanal-Identifier als 6 Byte-ASCII-String und eine fortlaufende Nummerierung:

- *Irch*	Interrupter-Kanal		
- *PARAxy*	Parameter-Kanal	mit xy = 01, 02, ... 09, 0A	
- *TBUFxy*	Transmit-Buffer_Long	mit xy = 01, ... 0A	
- *RBUFxy*	Receive-Buffer_Long	mit xy = 01,... 0A	
- *Buffxz*	Transmit-Buffer (128 Byte)	mit x = 1, ... A	z = a, b,...z

iosema ist mit der Kanalsemaphore und dem Kanal-Statusbit belegt:

- Bit 7	Semaphore:	'0' -- Kanal ist frei
		'1' -- Kanal ist belegt
- Bit 6 - 1	reserviert, Default: '0'	
- Bit 0	Kanalstatus:	'0' -- Kanal ist busy
		'1' -- Kanal ist ready

iostat wird bisher nicht versorgt und ist mit $0 vorbesetzt.

iocmmd ist das Kanalkommando und wird nur bei der Schnittstellenparametrierung benötigt (siehe Beschreibung des Parameterkanals ab Seite 19).

ionext ist der Pointer auf den nächsten Daten-Kanal. Wird nur Datenkanälen zugeordnet, ansonsten 0.

ioilev und ***ioivec***
bestimmen das Slave-Interruptverhalten. Sind *ioilev* und *ioivec* = 0, dann erfolgt keine Interrupt-Erzeugung des Slaves bei Beendigung des dem Kanal zugeordneten Auftrags, sondern es wird nur *iosema* entsprechend gesetzt. Andernfalls wird ein IRQ auf dem VME-Bus mit dem IRQ-Level *ioilev* (1..7) und dem IRQ-Vektor *ioivec* (`$00` .. `$FF`) erzeugt.

iobnum enthält die fortlaufende Numerierung der Kanäle.
Für den Interrupter-Kanal hat *iobnum* den Wert von 0.

In ***iolen*** steht die verfügbare Länge des Datenbereichs. Falls sich der Datenbereich innerhalb der Kanalstruktur befindet (default), so ist *iolen* = `$00080` == 128 Byte, bzw. `$400`. Ausgelagerte Datenbereiche können jedoch eine beliebige Länge haben.

iobuff ist der Pointer auf den dem Kanal zugeordneten Datenbereich. Als Default zeigt *iobuff* auf die Zelle *iodata* in der Kanalstruktur. Bei ausgelagerten Datenbereichen kann *iobuff* jedoch auf beliebige lokale Adressen zeigen, so daß bei der Adressierung des Datenbereichs **unbedingt** der aktuelle Inhalt von *iobuff* zu verwenden ist !

iorecl bestimmt die Anzahl der gültigen Daten im Datenbereich. (Anzahl der zu sendenden bzw. der empfangenen Daten).

Ist *iorecl* negativ, d.h. ist das oberste Bit gesetzt, so ist die Übertragung mit Fehler beendet worden!

Fehlercodes:
`$8007` - Time-Out
`$801E` - Framing-Error
`$801F` - Overrun-Error
`$8020` - Parity-Error
`$8046` - Break erkannt

iostio wird bisher nicht versorgt und ist mit $00 vorbesetzt.

ioldn enthält die Kanal-Server No. (1,...,10)

iomode dient zur Einstellung der Datenrichtung (Sende-/Empfangsbetrieb) sowie zur Parametrierung des Empfangsprotokolls :

Bit-No	Mnemo	Bedeutung	
15	*MODBWA*	0	Nach dem Senden aller Daten wird **kein** IRQ ausgelöst, der belegte Kanal wird vom Slave automatisch wieder frei gegeben.
		1	Nach dem Senden aller Daten wird *ready* in *iosema* gesetzt, bzw. der angegebene IRQ ausgelöst. Der belegte Kanal wird **nicht** wieder vom Slave freigegeben.
14	*MODBOU*	0	Kennung: Empfangskanal
		1	Kennung: Sendekanal
13	*MODBOU*	1	Bei Erkennen eines <cr> ($0D) wird der Empfang in diesen Kanal abgeschlossen.
12	*MODBLF*	1	Bei Erkennen eines <lf> ($0A) wird der Empfang in diesen Kanal abgeschlossen.
11	*MODBEO*	1	Bei Erkennen eines <eot> ($04) wird der Empfang in diesen Kanal abgeschlossen.
10	*MODBSC*	-	suppress_command: wird momentan nicht unterstützt
9	*MODBNE*	-	no_echo: wird momentan nicht unterstützt
8	*MODBIN*	0	kein Binärtransfer
		1	Binärtransfer: kein End-Check, kein Software-Handshake

Tabelle 1.4.3: Bits von *iomode*

Bit 7-0 von *iomode* ist als Mode-Erweiterungs-Byte reserviert. Bisher belegt sind folgende Kombinationen:

- `$00` Normaler I/O-Transfer (Default)
- `$08` Nur für Empfangsbetrieb:
  Alle bisher eingetroffenen Zeichen im lokalen Buffer werden gelöscht.

iotout Time-Out Wert.
Ist das oberste Bit (Bit 7) gesetzt, so erfolgt eine Time-Out-Überwachung des Kanals.
Erfolgt `kein` Transfer auf einem aktiven Kanal, so wird nach einer Zeit *T_Out* der Kanal wieder freigegeben und der Status *Time_Out* zurückgegeben! (über *iorecl*)

iofnam ist reserviert für ASCII-Einträge (bis zu 24 Byte).
Steht in den ersten 4 Byte von *iofnam* der ASCII-String `SCAN`, so gelten für einen Empfangskanal folgende Rückgabebedingungen:

1.) Rückgabe des Buffers, wenn <*iorecl*> Daten empfangen wurden
2.) Rückgabe des Buffers, wenn eine der in <*iomode*> spezifizierte Endebedingung erfüllt ist.
3.) Rückgabe des Buffers, wenn keine weiteren Daten im lokalen Interrupt-Buffer verfügbar sind, d.h., falls z.B. der Interrupt-Buffer leer ist, so wird der Empfangskanal sofort mit <*iorecl*>=0 zurückgegeben.

Für alle anderen Einträge in *iofnam* gelten nur die Endebedingungen 1.) und 2.). Mit dem Eintrag `PROT` erfolgt der Empfang über ein spezielles User-Protokoll.

ioentr dient zur Einbindung eines anwenderspezifischen Empfangsprotokolls (Nur Parameter-Kanal). Die Einsprungadresse des in einen freien Speicherbereich geladenen Protokolls wird hier eingetragen.

iotent ist reserviert zur Einbindung eines anwenderspezifischen Sendeprotokolls (Nur Parameter-Kanal).

iorxln bestimmt die Anzahl gültiger empfangener Daten, speziell auch im Fehlerfall.

iofree wird zur Zeit nicht belegt und wird mit $00 vorbesetzt.

iodata ist der Default-Datenbereich eines Kanals und hat eine Länge von 128 Byte, bzw. 1KByte (*TBUFxy, RBUFxy*).
Ein Überschreiben der Grenzen des Datenbereichs hat eine Zerstörung der gesamten I/O-Struktur zur Folge!

1.5 Verwaltung der Datenkanäle

1.5.1 Allgemeines

Die Kanäle sind, wie oben bereits erwähnt, in Parameter-, Buffer-, Default- und Interrupter-Kanäle aufgeteilt. Jeder seriellen Schnittstelle ist ein Parameter- und ein Default Tx-Buffer, ein Rx-Buffer, sowie eine Anzahl Buffer aus dem Buffer-Pool zugeteilt.

Der Parameter-, der Tx-Buffer und der Rx-Buffer sind **ausschließlich** der jeweiligen Schnittstelle zugeordnet. Die Buffer können prinzipiell von jedem Kanal in Anspruch genommen werden, aufgrund der Zeigerverkettung ergibt sich jedoch eine vorrangige Zuordnung der Buffer zu den jeweiligen Schnittstellenkanälen.

Die Verkettung des Tx-Buffer und der Buffer-Kanäle ist in den folgenden Tabellen zu erkennen. Über den Such-Pointer *ionext* ist dem jeweiligen Tx-Buffer-Kanal ein Buffer-Bereich zugeordnet. Der *ionext*-Pointer des letzten Buffers zeigt wieder auf den Tx-Buffer.

Diese Kanalaufteilung wurde gewählt, um ein hohes Maß an Flexibilität bei der Speicherzuteilung zu haben, während der Algorithmus des Speichersuchverfahrens schnell und einfach bleibt.

1.5.2 Übersicht der Kanäle mit Verkettung über Pointer

Kanal-Root-Pointer		
Adresse [HEX]	Inhalt [HEX]	Bemerkungen
0099C	08000	Startadresse des Buffer-Bereiches

Tabelle 1.5.1: Kanal-Root-Pointer auf Adresse *ANCHOR*

Buffer-Nummer [DEZ]	Adresse [HEX]	Kanal-Header						Bemerkungen
		iofor [HEX]	iobnum [HEX]	ioldn	ionext [HEX]	iolen [HEX]	ionam	
0	08000	08100	0	0	0	80	Irch__	Interrupter-Kanal
1	08100	08200	1	1	0	80	PARA01	Parameter-Kanal 1
2	08200	08300	2	2	0	80	PARA02	Parameter-Kanal 2
3	08300	08400	3	3	0	80	PARA03	Parameter-Kanal 3
4	08400	08500	4	4	0	80	PARA04	Parameter-Kanal 4
5	08500	08600	5	5	0	80	PARA05	Parameter-Kanal 5
6	08600	08700	6	6	0	80	PARA06	Parameter-Kanal 6
7	08700	08800	7	7	0	80	PARA07	Parameter-Kanal 7
8	08800	08900	8	8	0	80	PARA08	Parameter-Kanal 8
9	08900	08A00	9	9	0	80	PARA09	Parameter-Kanal 9
10	08A00	08B00	A	10	0	80	PARA0A	Parameter-Kanal 10

Tabelle 1.5.2: Interrupter-Kanal und Parameter-Kanäle

Buffer-Nummer [DEZ]	Adresse [HEX]	Kanal-Header						Bemerkungen
		iofor [HEX]	iobnum [HEX]	ioldn	ionext [HEX]	iolen [HEX]	ionam	
11	08B00	08F80	B	1	0E500	400	TBUF01	Transmit-Buffer 01
12	08F80	09400	C	1	08F80	400	RBUF01	Receive-Buffer 01
13	09400	09880	D	2	0FF00	400	TBUF02	Transmit-Buffer 02
14	09880	09D00	E	2	09880	400	RBUF02	Receive-Buffer 02
15	09D00	0A180	F	3	11900	400	TBUF03	Transmit-Buffer 03
16	0A180	0A600	10	3	0A180	400	RBUF03	Receive-Buffer 03
17	0A600	0AA80	11	4	13300	400	TBUF04	Transmit-Buffer 04
18	0AA80	0AF00	12	4	0AA80	400	RBUF04	Receive-Buffer 04
19	0AF00	0B380	13	5	14B00	400	TBUF05	Transmit-Buffer 05
20	0B380	0B800	14	5	0B380	400	RBUF05	Receive-Buffer 05
21	0B800	0BC80	15	6	16700	400	TBUF06	Transmit-Buffer 06
22	0BC80	0C100	16	6	0BC80	400	RBUF06	Receive-Buffer 06
23	0C100	0C580	17	7	18100	400	TBUF07	Transmit-Buffer 07
24	0C580	0CA00	18	7	0C580	400	RBUF07	Receive-Buffer 07
25	0CA00	0CE80	19	8	19B00	400	TBUF08	Transmit-Buffer 08
26	0CE80	0D300	1A	8	0CE80	400	RBUF08	Receive-Buffer 08
27	0D300	0D780	1B	9	1B500	400	TBUF09	Transmit-Buffer 09
28	0D780	0DC00	1C	9	0D780	400	RBUF09	Receive-Buffer 09
29	0DC00	0E080	1D	10	1CF00	400	TBUF0A	Transmit-Buffer 0A
30	0E080	0E500	1E	10	0E080	400	RBUF0A	Receive-Buffer 0A

Tabelle 1.5.3: Transmit-, und Receive-Buffer

Buffer-Nummer [DEZ]	Adresse [HEX]	Kanal-Header						Bemerkungen
		iofor [HEX]	iobnum [HEX]	ioldn	ionext [HEX]	iolen [HEX]	ionam	
31	0E500	0E600	1F	1	0E600	80	BUFF1a	26 Buffer für Kanal 1
32	0E600	0E700	20	1	0E70	80	BUFF1b	
:	:	:	:	:	:	:	:	
55	0FD00	0FE00	37	1	0FE00	80	BUFF1y	
56	0FE00	0FF00	38	1	08800	80	BUFF1z	
57	0FF00	10000	39	2	10000	80	BUFF2a	26 Buffer für Kanal 2
:	:	:	:	:	:	:	:	
82	11800	11900	52	2	09400	80	BUFF2z	
83	11900	11A00	53	3	11A00	80	BUFF3a	26 Buffer für Kanal 3
:	:	:	:	:	:	:	:	
108	13200	13300	6C	3	09D00	80	BUFF3z	
109	13300	13400	6D	4	13400	80	BUFF4a	26 Buffer für Kanal 4
:	:	:	:	:	:	:	:	
134	14C00	14D00	86	4	0A600	80	BUFF4z	
135	14D00	14E00	87	5	14E00	80	BUFF5a	26 Buffer für Kanal 5
:	:	:	:	:	:	:	:	
160	16600	16700	A0	5	0AF00	80	BUFF5z	
161	16700	16800	A1	6	16800	80	BUFF6a	26 Buffer für Kanal 6
:	:	:	:	:	:	:	:	
186	18000	18100	BA	6	0B800	80	BUFF6z	
187	18100	18200	BB	7	18200	80	BUFF7a	26 Buffer für Kanal 7
:	:	:	:	:	:	:	:	
212	19A00	19B00	D4	7	0C100	80	BUFF7z	
213	19B00	19C00	D5	8	19C00	80	BUFF8a	26 Buffer für Kanal 8
:	:	:	:	:	:	:	:	
238	1B400	1B500	EE	8	0CA00	80	BUFF8z	
239	1B500	1B600	EF	9	1B600	80	BUFF9a	26 Buffer für Kanal 9
:	:	:	:	:	:	:	:	
264	1CE00	1CF00	108	9	03D00	80	BUFF9z	
265	1CF00	1D000	109	10	1D000	80	BUFFAa	26 Buffer für Kanal 10
:	:	:	:	:	:	:	:	
289	1E700	1E800	121	10	1E800	80	BUFFAy	
290	1E800	08000	122	10	0DC00	80	BUFFAz	

Tabelle 1.5.4: Buffer-Kanäle 1 bis 10

1.6 Buffer-Zuteilung

1.6.1 Speicherzuteilung über Semaphore

Um die Speicherverwaltung Multitasking- und Multiuser-fest zu gestalten, erfolgt die Speicherzuteilung über eine Semaphore, auf die mit dem unteilbaren Assemblerbefehl `TAS` zugegriffen werden muß.

Beginnend mit der Adresse des entsprechenden Default-Kanals wird die Semaphore des Kanals belegt.

War der Zugriff erfolgreich, so ist der entsprechende Kanal jetzt belegt, war der Zugriff vergeblich, so ist die nächste Adresse aus *ionext* zu bestimmen. Abbruch- und Wartebedingung können eine gewisse Anzahl von vergeblichen Zugriffen sein, bzw. eine Erkennung auf 'wrap-around' (neu_pointer < alt_pointer).

Nach Beendigung des I/O-Auftrags gibt entweder der Slave-Server den Kanal durch Rücksetzen der Semaphore wieder frei, bzw. der Master muß entscheiden, ab wann er den Kanal wieder zur Verfügung stellt.

1.6.2 Beispiel einer Buffer-Zuteilung

```
* Speicher auf der ISER-8 allozieren
*
            MOVEA.L crdadr,A0            ;Basisadresse ISER-8
            MOVE.L  dfltbf,D0            ;kartenrelative Adresse
                                         ;des Defaultbuffers
            BSR     srchbuff             ;Buffer suchen
            BNE     no_success           ;Kein Buffer frei
*           sonst:  in A0 aktuelle  absolute Adresse des Kanals
*                   in D0 die kartenrelative Adresse des Kanals
            ....
            ....
         -- Transfer --
            END

srchbuff MOVE.L  D0,D1                   ;Einstiegsadresse als
                                         ;Endebedingung
srch1       TAS     iosema(A0,D0.L)      ;Semaphorenzugriff
            BEQ.S   srchex               ;Semaph. war frei
                                         ;kartenrel. Adr.in D0
            MOVE.L  ionext(A0,D0.L),D0   ;nächster Kanal
            CMP.L   D0,D1                ;Endebedingung ?
            BGT.S   srch1                ;Nein, weitersuchen
            TST.L   D0                   ;Flag 'NE'
srchex      LEA     0(A0,D0.L),A0        ;absolute Adresse in A0
            RTS
```

2. Kanalbeschreibungen

2.1 Beschreibung der Datenkanäle

Datenkanäle dienen zum Transfer von Sende-/Empfangsdaten und sind vom Typ Default- bzw. Buffer-Kanal. Vor Beginn eines Sende-/Empfangstransfers wird ein Daten-Kanal nach oben beschriebenem Beispiel alloziert. Danach wird der Header des Kanals mit den entsprechenden Parametern versorgt, gegebenenfalls Daten eingetragen und der lokalen CPU übergeben.

	Address Offset HEX	+0	+2	+4	+6	+8	+A	+C	+E
HEADER	00	iofor		ioback		iotyp	ioname		
	10	sema / iostat	iocmmd	ionext		iolev / iovec	iobnum	iolen	
	20	iobuff		iorecl	iostio / ioldn	iomode	iotout / iodrv	iofnam...	
	30	...iofnam...							
	40	...iofnam		ioentr		iotent		iofree	
	50	iorxlen*		iofree...					
	60	...iofree...							
	70								...iofree
DATA AREA	80	iodata...							
	90	...iodata...							
	A0	...iodata...							
	B0	...iodata...							
	C0	...iodata...							
	D0	...iodata...							
	E0	...iodata...							
	F0	...iodata							

* Nur für Rx-Buffer

Tabelle 2.1.1: Interne Kanalstruktur (für alle Kanaltypen gültig)

Address Offset HEX	+0		+2		+4		+6		+8		+A		+C		+E	
8B00	00	00	**8F**	**80**	00	00	**8A**	**00**	**01**	**14**	**'TBUF01'**					
8B10	00	00	00	00	00	00	**E5**	**00**	00	00	00	**0B**	00	00	**04**	**00**
8B20	00	00	**8B**	**80**	00	00	00	**01**	00	00	00	00	00	00	00	00
8B30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8B40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8B50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8B60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8B70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

8B80	*iodata...*
:	*...iodata...*

8F70	*...iodata*

Tabelle 2.1.2: Default-Kanäle (Bsp.: `TBUF01`)

2.2 Beschreibung des Parameter-Kanals

2.2.1 Struktur des Parameter-Kanals

Jedem seriellen Schnittstellenkanal ist ein sogenannter Parameter-Kanal fest zugeordnet. Im Datenbereich dieses Parameter-Kanals ist der aktuelle Status der Schnittstelle abgelegt, der völlig transparent vom VMEbus-Master gelesen werden kann.

Der Parameter-Kanal wird ebenso zur Parametrierung der Schnittstelle benötigt. Dazu werden die aktuellen Parameter an den entsprechenden Stellen des Parameter-Kanals eingetragen, und der Parameter-Kanal wird als Sendekanal dem VME-ISER-Server übergeben. (siehe auch Kap. ‘Der lokale ISER-Server’ ab Seite 31). Dadurch wird eine Synchronisation mit laufenden Sende- und Empfangs-aufträgen erreicht.

Der Parameter-Kanal ist in 2 verschiedene Bereiche aufgeteilt:
- vom Anwender beschreibbare Parameter (Offset: `$80 – $BF`)
- vom Anwender **nur lesbare** Parameter (Offset: `$C0 – $FF`)

Die Parameter *txb...hnd* sind als Byte formatiert und sind als Identifier für die physikalische Parametrierung zu interpretieren.

Address Offset HEX	+0		+2		+4		+6		+8		+A		+C		+E	
8100	00	00	**82**	**00**	00	00	00	00	00	**0C**	***'PARA01'***					
8110	00	00	*'iocmmd'*		00	00	00	00	00	00	00	**01**	00	00	00	**80**
8120	00	00	**81**	**80**	00	00	00	00	00	00	00	00	00	00	00	00
8130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8140	00	00	00	00	*Protokoll*				*Protokoll*				00	00	00	00
8150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8170	*tx_buffer1*				*rx_buffer1*				00	00	00	00	00	00	00	00
8180	*txbs*	*rxbs*	*chrls*	*stpls*	*parts*	*hnds*	*rxtime0*		*rxtime1*		*ttimes*		*txclkmods*	*rxclkmods*	reserved	
8190	*txbvs*				*rxbvs*				*protoks*	*encodes*	00	00	00	00	00	00
81A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
81B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
81C0	*txb*	*rxb*	*chrl*	*stpl*	*part*	*hnd*	*rtime0*		*rtime1*		*ttime*		*txclkmod*	*rxclkmod*	reserved	
81D0	*txbv*				*rxbv*				*protok*	*encode*	*endpar= FFFF*		00	00	00	00
81E0	*rxfifo*	*rxtout*	*resrv*	*spchr1*	*spchr2*	*spchr3*	*spchr4*	00	00	00	00	00	00	00	00	00
81F0	*txstat*	*rxstat*	*errlog*	00	00	00	00	00	00	00	00	00	00	00	00	00

Tabelle 2.2.1: Parameter-Kanäle (Bsp.: Parameter-Kanal 1)

2.2.2 Beschreibung der Parameter

Schreibzugriffe auf die Parameter können nur erfolgen , wenn in der Zelle *iocmmd* das Kommando *paraxy* (`$0000`) eingetragen ist. Lesezugriffe auf die Parameter sind, unabhängig von *iocmmd*, immer möglich. (siehe auch 'Kommandoübergabe über den Parameter-Kanal' ab Seite 25).

Schreib- und lesbare Parameter:

txbs	Index Sollwert *baud*:	Transmitter-Baudrate
rxbs	Index Sollwert *baud*:	Receiver-Baudrate
chrls	Index Sollwert *chri*:	Bits/Char
stpls	Index Sollwert *stpi*:	Anzahl Stop-Bits
parts	Index Sollwert *pari*:	Parity-Typ
hnds	Index Sollwert *hndi*:	Handshake-Mode

Zuordnung der Index-Sollwerte der Parameter:

```
Bedeutung der Baudraten baud:
0  --   Baudrate = 38400
1  --   Baudrate = 19200
2  --   Baudrate =  9600
3  --   Baudrate =  4800
4  --   Baudrate =  2400
5  --   Baudrate =  1200
6  --   Baudrate =   600
7  --   Baudrate =   300
8  --   Baudrate =   150
9  --   Baudrate =   110
10 --   Baudrate =    75
11 --   Baudrate =    50
$FF  --   Baudrate = variabel über txbv, rxbv

nur für Kanal 9 und 10:
12  --   Baudrate =  76800
13  --   Baudrate = 115200

Bedeutung des Index chri:
0  --   8 Bits per Character
1  --   7 Bits per Character
2  --   6 Bits per Character
3  --   5 Bits per Character

Bedeutung des Index stpi:
0  --   1 Stopbit
1  --   2 Stopbits

Bedeutung des Index pari:
0  --   Keine Rx-, keine Tx-Parity
1  --   Rx-/Tx-Parity ODD
2  --   Rx-/Tx-Parity EVEN

Bedeutung des Index hndi:
0  --   Hardware Handshake DTR/CTS
1  --   Software Handshake XON/XOFF
2  --   Modem-Betrieb RTS, CTS Handshake
3  --   kein Handshake
4  --   RS-485-Betrieb, kein Handshake
5  --   RS-422-Betrieb, XON/XOFF Handshake
```

rtime0s* Receive Time-Out für das erste Zeichen in msec
0: Receive Time-Out disabled

rtime1s* Receive 'character-to-character' Time-Out in msec
0: kein 'charakter-to-charakter Time-Out

ttimes* Transmit Time-Out in msec
0: Transmit Time-Out disabled
* siehe auch Abschn.'Time-Out' auf Seite 32

rxclkmods ***txclkmods*** Takt-Betriebsart der DUSCC-Kanäle für Receive und Transmit getrennt anzugeben:

rxclkmods ***txclkmods***	*txbvs*	Betriebsart	Funktion des Pins RxTxCLK	Takt
x	0	Kanal ausgeschaltet	ohne Bedeutung	ohne Bedeutung
0	≠ 0	Async-Mode	ohne Bedeutung	16x Baudrate
1	≠ 0	Synch-Mode	Pin RxTxCLK = OUT	1x Baudrate
2	≠ 0	Synch-Mode	Pin RxTxCLK = OUT	16x Baudrate
-1	≠ 0	Synch-Mode	Pin RxTxCLK = IN	1x Baudrate
-2	≠ 0	Synch-Mode	Pin RxTxCLK = IN	16x Baudrate

Tabelle 2.2.2: Auswertung von *rxclkmods* und *txclkmods*

Pin RxTxCLK = DUSCC-Pin 39 (J3A-Pin 3) für Kanal 9,
bzw. DUSCC-Pin 10 (J3-Pin 3) für Kanal 10

txbvs ***rxbvs*** Baudrate absolut, Wertebereich `50...`∞ (asynchron),
Einheit Baud
In *txbvs* und *rxbvs* wird die aktuelle Baudrate als absolute Zahl angegeben. Wird eine Baudrate gewünscht, die von den über *txb*, bzw. *rxb* auswählbaren Baudraten abweicht, so kann über *txbvs*, bzw. *rxbvs* die Baudraten absolut übergeben werden (*txbs*, bzw. *rxbs* auf `$FF` gesetzt).
Die Schnittstelle wird mit der am nächsten gelegenen möglichen Baudrate programmiert und der wirkliche Wert der eingestellten Baudrate wird in *txbv* und *rxbv* zurückgegeben.

Beispiel: Parametrierung mit Tx-Baudrate 115.000 Baud auf der VME-ISER8

Eingabe : $FF --> *txbs*
Eingabe : 115000 --> *txbvs*
Rückgabe: -->> *txbv* = 115200
(wirkliche Baudrate = 115200 Baud!)

Anmerkung: Die Hardware der VME-ISER12 bietet gegenüber der VME-ISER8 eine verbesserte Auflösung für die Einstellung der absoluten Baudrate, weil hier eine weitere Grundfrequenz zur Erzeugung der Baudrate zur Verfügung steht.

protoks Protokoll-Mode der Kanäle 9 und 10

protoks	Protokoll-Mode
0	UART-Mode (alle Parameter der Parameter-Kanäle 9 und 10 sind relevant)
1	HDLC-Mode (nur die für synchrone Übertragungen not wendigen Parameter der Kanäle 9 und 10 müssen berücksichtigt werden: *rtime02, txclkmods, rxclkmods. txbvs, encode*)

Tabelle 2.2.3: Protokoll-Mode

encodes Signalkodierung der seriellen Schnittstellen
Zur Zeit wird nur das Format NRZ (No Return To Zero) unterstützt (*encodes* = 0).

Nur lesbare Parameter:

Als Statusinformation dienen folgende Parameter:
(**nicht** vom Anwender zu Beschreiben !!)

txb	Index Istwert *baud*:	Transmitterbaudrate
rxb	Index Istwert *baud*:	Receiverbaudrate
chrl	Index Istwert *chri*:	Bits/Character
stpl	Index Istwert *stpi*:	Anzahl Stopbits
part	Index Istwert *pari*:	Parity-Typ
hnd	Index Istwert *hndi*:	Handshake-Mode

(Zuordnung der Index-Sollwerte siehe Seite 20.)

rtime0* Receive Time-Out für das erste Zeichen in msec
rtime1* Receive 'character-to-character' Time-Out in msec
ttime * Transmit Time-Out in msec
* siehe auch Abschn.'Time-Out' auf Seite 32

txclkmod, Rücklesen der Parameter der Takt-Betriebsart der DUSCC-Kanäle
rxclkmod (Bedeutung der Parameter siehe Tabelle auf Seite 21)

txbv, Baudrate absolut, Wertebereich 50...38400, Einheit Baud
rxbv In *txbv* und *rxbv* wird die aktuelle Baudrate als absolute Zahl angegeben.
(siehe auch txbvs, rxbvs auf Seite 21)

protok Protokoll-Mode der Kanäle 9 und 10
$00 - UART-Mode
$01 - HDLC-Mode
(siehe auch *protoks* auf Seite 22)

encode Signalkodierung der seriellen Schnittstellen
Zur Zeit wird nur das Format NRZ (No Return To Zero) unterstützt (*encode* = 0).

rxfifo interne FiFo-Schwelle für Rx-Interrupt (lokal !!)
rxtout Zeit für Rx-Timeout in 5 msec Units (lokal !!)
resrv reserviert
spchr1-
spchr4 interne Controller-Commands

txstat Status des Transmitters

Bit 7 : nicht benutzt
Bit 6 : nicht benutzt
Bit 5 : nicht benutzt
Bit 4 : nicht benutzt
Bit 3 : '1' - Tx-Time-Out aufgetreten
'0' - kein Tx-Time-Out aufgetreten
Bit 2 : '1' - Tx-Queue voll
'0' - Tx-Queue bereit
Bit 1 : '1' - Transmitter durch Handshake gesperrt
'0' - Transmitter durch Handshake freigegeben
Bit 0 : '1' - Transmitter gesperrt
'0' - Transmitter frei

rxstat Status des Receivers

Bit 7 : '1' - Break erkannt
'0' - kein Break erkannt
Bit 6 : '1' - Parity-Fehler erkannt
'0' - kein Parity-Fehler erkannt
Bit 5 : '1' - Framing-Fehler erkannt
'0' - kein Framing-Fehler erkannt
Bit 4 : '1' - Receiver-Overrun erkannt (Datenverlust!)
'0' - kein Receiver-Overrun erkannt
Bit 3 : '1' - Rx-Time-Out aufgetreten
'0' - kein Rx-Time-Out aufgetreten
Bit 2 : '1' - Zeichen im lokalen Interrupt-Buffer
'0' - kein Zeichen im lokalen Interrupt-Buffer
Bit 1 : '1' - Receiver hat Handshake auf 'gesperrt' gesetzt
'0' - Receiver hat Handshake auf 'frei' gesetzt
Bit 0 : '1' - Receiver gesperrt
'0' - Receiver freigegeben

errlog Enable/Disable Rx-Error Funktion, Read Only.

errlog = $00 - keine Rx-Error Funktion
errlog = $FF - Rx-Error Funktion freigegeben

errlog wird gesetzt durch das Kommando *receive-errlog*.
errlog wird zurückgesetzt durch *receive-on* und *receive-off*.

2.2.3 Kommandoübergabe durch den Parameter-Kanal

Über die Parameter im Datenbereich hinaus können über den Parameter-Kanal auch Kommandos an den Schnittstellenkanal übergeben werden. Der Parameter-Kanal wird dazu in die Queue des Tx-Servers eingetragen und somit synchron abgearbeitet.

Eine Ausnahme bilden die Befehle 'clear' und 'reset', die schon vor dem Eintragen in die Queue abgewickelt werden.

Das entsprechende Kommando wird in die Zelle **iocmmd** im Header des Parameter-Kanals eingetragen.

Bisher implementierte Kommandos:

`$0000`	**paraxy**
`$000C`	**clear**
`$000D`	**reset**
`$000E`	**reset-Status**
`$0050`	**receive-Off**
`$0051`	**receive-On**
`$0052`	**receive-Errlog**
`$FFFF`	**sync**

Beschreibung der Kommandos:

paraxy dient zum Ändern der Schnittstellenparameter, wie z.B. Baudrate, Handshake, usw.

clear löscht die lokal gespeicherten RX-Daten;
setzt die Output-Queue zurück, ändert keine Schnittstellenparameter

reset Grundinitialisierung des Kanals mit Defaultparametern

reset-stat setzt die *error*-Flags in *txstat* und *rxstat* zurück

receive-off dient dem expliziten Abschalten des Empfängers

receive-on dient dem expliziten Einschalten des Empfängers (kein 'End-by-error')

receive-errlog schaltet den Empfängers ein, gibt die Funktion 'End-by-error' frei

sync einlinken des Parameter-Kanals als Output ohne Daten, kein Datentransfer, keine Änderung des Schnittstellenstatus.

Bei stark belastetem Sendebetrieb ohne 'wait for ready' (*MODMWA* in *iomode*=0) wird der Zustand 'Output Queue full' sehr schnell erreicht, so daß der Master im Polling-Betrieb auf 'Output Queue ready' abfragen muß.
Nach einem weiteren Transfer ist jedoch die Queue wieder gefüllt.
In diesem Zustand bietet es sich an, einen 'Dummy-Transfer' mit 'wait for ready' und aktiviertem Interrupt-Mode durchzuführen, so daß nach der kompletten Abarbeitung der Queue dem Master wieder die gesamte Speichertiefe zur Verfügung steht.

2.3 Beschreibung des Interrupter-Kanals

2.3.1 Struktur des Interrupter-Kanals

Der Interrupter-Kanal hat die Aufgabe, eine Verbindung zwischen dem VME-Master-Programm und dem lokalen Server herzustellen.

Das Master-Programm muß, nachdem es einen Daten-Kanal alloziert hat und die Parameter im Header des Kanals eingetragen hat, den Kanal dem lokalen Server übergeben. Dafür stellt der Interrupter-Kanal in seinem Datenbereich die Zellen *TCHACH1-TCHACHA* und *RCHACH1-RCHACHA* zur Verfügung.

Das Master-Programm trägt in diesen Zellen die **kartenrelative** Adresse des zu bearbeitenden Kanals (D0 in dem o.g. Beispiel) ein, und aktiviert durch Triggern eines lokalen Interrupts den VME-ISER-Server, der anhand des Eintrags im Interrupter-Kanal den Datenkanal identifiziert und somit bearbeiten kann.

Für jede der 10 Schnittstellen stellt der Interrupter-Kanal je einen Eintrag für Sende- und Empfangsbetrieb zur Verfügung.

Die Zellen *TCHACHx/RCHACHx* dienen ebenso als Statuszellen:

Ist der Inhalt der Zelle *CHACHx* ungleich `$00000000.L`, so wurde der entsprechende Datenkanal noch nicht in die Queue des VME-ISER-Servers aufgenommen, und es darf kein neuer Eintrag erfolgen.

Sobald der Datenkanal in die Verwaltung des Servers aufgenommen wurde, wird der Eintrag im Interrupter-Kanal auf `$00000000.L` gesetzt. Dies gibt jedoch noch keinen weiteren Aufschluß über den Status des betreffenden Kanals. Dieser kann nur dem Zustand der *iosema*-Zelle im Header des Datenkanals entnommen werden!

Address Offset HEX	+0		+2		+4		+6		+8		+A		+C		+E	
8000	00	00	**81**	**00**	00	00	00	00	**FF**	**FF**	***'Irch__'***					
8010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**80**
8020	00	00	**80**	**80**	00	00	00	00	00	00	00	00	00	00	00	00
8030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8040	*addr_para1*				*addr_para2*				*addr_para3*				*addr_para4*			
8050	*addr_para5*				*addr_para6*				*addr_para7*				*addr_para8*			
8060	*addr_para9*				*addr_paraA*				00	00	00	00	00	00	00	00
8070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8080	*TCHACH1*				*TCHACH2*				*TCHACH3*				*TCHACH4*			
8090	*TCHACH5*				*TCHACH6*				*TCHACH7*				*TCHACH8*			
80A0	*TCHACH9*				*TCHACHA*				00	00	00	00	00	00	00	00
80B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80C0	*RCHACH1*				*RCHACH2*				*RCHACH3*				*RCHACH4*			
80D0	*RCHACH5*				*RCHACH6*				*RCHACH7*				*RCHACH8*			
80E0	*RCHACH9*				*RCHACHA*				00	00	00	00	00	00	00	00
80F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Tabelle 2.3.1: Interrupter-Kanal

2.3.2 Beschreibung der Zellen des Interrupter-Kanals

addr_para1... Startadressen der Parameterkanäle 1 bis 10
adr_paraA

TCHACH1...
TCHACHA Einträge für den **Tx-Server**:

Zelle	Offset [HEX] relativ zu *iodata*	Eintrag Kanal für Tx-Server
TCHACH1	00	1
TCHACH2	04	2
TCHACH3	08	3
TCHACH4	0C	4
TCHACH5	10	5
TCHACH6	14	6
TCHACH7	18	7
TCHACH8	1C	8
TCHACH9	20	9
TCHACHA	24	10

Tabelle 2.3.2: Einträge für den Tx-Server

Triggern des lokalen VME-ISER-Tx-Irq's:

Um die VME-ISER-Tx-Server-Task zu aktivieren, die die Einträge im Interrupter-Kanal abarbeitet, muß ein Zugriff auf die lokale IRQ-Trigger-Adresse erfolgen.
Dieser Zugriff muß als 'write-Word' auf die **kartenrelative** Adresse

tirtrig = `$080002` erfolgen.

RCHACH1...
RCHACHA Einträge für den **Rx-Server**:

Zelle	Offset [HEX] relativ zu *iodata*	Eintrag Kanal für Rx-Server
RCHACH1	40	1
RCHACH2	44	2
RCHACH3	48	3
RCHACH4	4C	4
RCHACH5	50	5
RCHACH6	54	6
RCHACH7	58	7
RCHACH8	5C	8
RCHACH9	60	9
RCHACHA	64	10

Tabelle 2.3.3: Einträge für den Rx-Server

Triggern des lokalen VME-ISER-Tx-Irq's:

Um die VME-ISER-Rx-Server-Task zu aktivieren, die die Einträge im Interrupter-Kanal abarbeitet, muß ein Zugriff auf die lokale IRQ-Trigger-Adresse erfolgen.
Dieser Zugriff muß als 'write-Word' auf die **kartenrelative** Adresse

rirtrig = `$080006` erfolgen.

3. Der lokale VME-ISER-Server

3.1 Funktionsweise des lokalen VME-ISER-Servers

Der lokale VME-ISER-Server verwaltet alle Kanäle, die vom VME-Master-Programm zur VME-ISER übergeben wurden. Grundlegend unterscheidet der Server zwischen **Input-** und **Output-** Kanälen. Die Abarbeitung eines Parameter-Kanals ist eine Sonderform eines Output-Kanals.

3.1.1 Output-Kanäle

Der VME-ISER-Server hat für jede Schnittstelle eine lokale Abarbeitungsqueue. Diese ist als Default 32-Einträge tief. Ein per Interrupt-Kanal eingelinkter Output-Datenkanal wird in die Queue eingetragen und der für die Schnittstelle zuständige **TX-Server** entnimmt sich den jeweiligen Kanal aus der Queue und gibt den Eintrag nach der kompletten Abarbeitung wieder frei.

Ein Überlaufen der Queue wird durch den Handshake mit den Zellen *TCHACHx* verhindert: Ist die Queue gefüllt, so wird zwar der Eintrag des entsprechenden Datenkanals übernommen, die Zelle *TCHACHx* wird jedoch noch nicht wieder freigegeben. Dies erfolgt erst dann, wenn Platz für mindestens einen weiteren Eintrag in der Queue ist.

Erkennt der TX-Server den aktuellen Output-Kanal als Parameter-Kanal, so erfolgt keine Ausgabe, sondern eine Abarbeitung des Kommandos **iocmmd**.

3.1.2 Input-Kanäle

Jedem der 10 seriellen Schnittstellen ist als Default ein sogenannter **Interrupt-Buffer** zugeteilt, zu dem der Anwender jedoch keinen direkten Zugriff hat.

Werden Daten auf der Schnittstelle empfangen, und liegt dem Input-Server **kein** Input-Buffer vor, so werden die einkommenden Daten im Interrupt-Buffer zwischengespeichert.

Solange noch Daten im Interrupt-Buffer sind, wird ein vom VME-Master eingelinkter Input-Kanal mit diesen Daten gefüllt, andernfalls werden einkommende Daten direkt in den Input-Kanal übertragen.

Ausnahmen:

- Wird ein Input-Kanal mit *iomode* = `$xx08` bearbeitet, so werden alle bisher eingetroffenen Daten im Interrupt-Buffer gelöscht und es werden nur die ab diesem Zeitpunkt eintreffenden Daten in dem nächsten READ-Aufruf übergeben.

- Ist *iofnam* auf ASCII `'SCAN'` gesetzt, so werden Daten aus dem Interrupt-Buffer bis zum Erreichen der angegebenen Endebedingung übergeben. Ist der Interrupt-Buffer geleert, so wird ebenso die Endebedingung gesetzt.

- Ist *iofnam* auf ASCII `'PROT'` gesetzt, so wird das eingetragene Protokoll ausgeführt.

Der Interrupt-Buffer hat als Default eine Länge von 1k-Byte. Der Receive-Handshake wird entsprechend dem Füllgrad des Interrupt-Buffer verwaltet:

Ist für die Schnittstelle ein Handshake vorgesehen, so wird bei einem Füllgrad von etwa 90% der Handshake auf **sperren** gesetzt.
Unterschreitet der Füllgrad des Interrupt-Buffers etwa 30%, so wird der Handshake auf **freigeben** gesetzt.

3.1.3 Interrupt-Betrieb

Wünscht der Anwender einen VME-Interrupt von der VME-ISER nach Beendigung eines Auftrags (Input-Kanal gefüllt, oder Output-Kanal übertragen mit *MODMWA* = '1' in *iomode*), so muß in die Zellen *ioilev* und *ioivec* des entsprechenden Datenkanals der gewünschte VME-Interrupt-Level, sowie der Interrupt-Vektor eingetragen werden. Die VME-ISER löst dann den so spezifizierten Interrupt aus. Wird keine Interrupt-Erzeugung gewünscht, so ist *iolev* auf 0 zu setzen.
Der Anwender muß in seiner Interrupt-Routine den Interrupt bestätigen. Die Bestätigung des Interrupts erfolgt folgendermaßen:

Die niederwertigsten 2 Bit des Interrupt-Vektors bestimmen die Bit-position in dem Interrupt-Ackn.-Register. Dieses Bit ist als Acknowledge auf '1' zu setzen. Die **kartenrelative** Adresse dieses Registers ***iack*** ist `$08601B`.

z.B.:

```
---   Interrupt-Entry  ---
            MOVE.B  #ioivec,D0          ;aktueller Int-Vektor
            ANDI.B  #$03,D0             ;Maskieren Bit2-7
            BSET    D0,iack+iserbase    ;Setzen des Bits auf VME-ISER
---   weitere Interrupt-Routine --
```

Das Setzen des IACK-Bits sollte so bald wie möglich erfolgen, da auf der VME-ISER die Auslösung eines neuen IRQ's so lange verhindert wird, bis der aktuelle Interrupt bestätigt wurde!!

3.1.4 Time-Out

Es besteht die Option, Sende- bzw. Empfangsaufträge nach einer vorgegebenen Zeit *T-Out* abzubrechen. Die Vorgabe der Zeit erfolgt über den Kanal-Parameter *iotout*, bzw. über die Parameter *rtime0*, *rtime1* und *ttime* im Parameter-Kanal.
Der Wert in *iotout* bezieht sich auf den in Abarbeitung befindlichen Kanal, während *rtime0*, *rtime1* und *ttime* sich global auf die Schnittstelle beziehen.
Die Angabe von *iotout* übersteuert die Angaben im Parameter-Kanal.

iotout Ist Bit 7 von *iotout* = 0, so ist ein Time-Out über <*iotout*> disabled.
Ist Bit 7 = 1, so ergibt der Wert in den anderen 7 Bits die Time-Out Zeit in 10 msec-Einheiten.
z.B.:
iotout = `$0x` - kein Time-Out
iotout = `$85` - Time-Out nach 50 msec
iotout = `$FF` - Time-Out nach 1,2 sec

Über den Parameter-Kanal ist es möglich, ein schnittstellenglobales Time-Out aufzusetzen, wobei zwischen Sende- und Empfangsbetrieb unterschieden werden kann.
Der Wertebereich umfaßt `0....32767`, die Einheit ist 1msec.
Ist *rtime0*=0, bzw. *ttime*=0, so ist die jeweilige Time-Out Funktion blockiert!

ttime - Time-Out für Sendebetrieb
rtime0 - Time-Out für Empfangsbetrieb für das erste Zeichen
rtime1 - Time-Out für Empfangsbetrieb für alle weiteren Zeichen

Die Time-Out Funktion ist retriggerbar, d.h. erfolgt eine Sende-/Empfangsoperation, so wird der entsprechende Counter zurückgesetzt. Der zeitliche Abstand dieser Operationen ist variabel (FIFO-Betrieb) und entspricht der Dauer von mindestens einer, maximal jedoch 8 Character-Zeiten.
(z.B. 1200 Baud: 1 Char.Time . (1+8+1)/1200 = 8,3 msec
8 Char.Time . 66,6 msec,
d.h. ein Time-Out Wert unter 67 msec ist nicht zu empfehlen!)

Im Empfangsbetrieb wird darüber hinaus noch zwischen erstem und 'character-to-character' Time-Out unterschieden, d.h. die Zeit zwischen Auftragseingang und Eintreffen des ersten Zeichens darf größer sein als die character-to-character Zeit bei einer laufenden Übertragung.

Aktionen bei erfolgtem Time-Out:

Ist bei einer Übertragung ein Time-Out aufgetreten, so erfolgen prinzipiell folgende Aktionen:

1. Im entsprechenden Kanal wird die Time-Out Marke gesetzt:
   `$8007` --> *iorecl*

2. Im Parameter-Kanal wird das Time-Out Bit in *rxstat*, bzw. *txstat* gesetzt.

Das Rücksetzen dieser Bits erfolgt über das Kommando **reset-sta**t im Parameter-Kanal oder bei einer Neuinitialisierung des Kanals. Es wird **nicht** durch eine erfolgreiche Ein-/Ausgabe zurückgesetzt!

Der in Arbeit befindliche Kanal wird wieder freigegeben, d.h. bei einem Sendekanal ohne 'wait' wird der Kanal 'verschrottet'. Die Semaphore *iosema* wird zurückgesetzt und der nächste Sendekanal der Queue entnommen.

Bei Sendebetrieb mit 'wait', bzw. bei einem Empfangskanal wird der Master entsprechend benachrichtigt. Der Kanalstatus wird auf 'ready' gesetzt und, falls verlangt, wird ein Interrupt ausgelöst.

3.1.5 Receive-Error-Mode

Treten im Rx-Betrieb Fehler auf, so werden diese in *rxstat* protokolliert. Ein Zurücksetzen des Rx-Status erfolgt durch die Kommandos

reset-stat, **reset** oder **receive-errlog**.

Detektierbare Fehler sind Break, Parity-, Framing- und Overrun-Fehler.

Wird eine Auswertung dieser Fehler gewünscht, so muß über das Kommando **receive-errlog** der Receiver-Error-Mode aktiviert werden.

Tritt im aktiven Mode einer der o.g. Fehler auf, und liegt kein Receive-Auftrag vor, so werden alle bisher im Interrupt-Buffer eingetroffenen Zeichen gelöscht. Liegt ein Rx-Auftrag vor, so wird der Auftrag abgebrochen und es wird eine Fehlernummer über *iorecl* zurückgegeben.

Bei gleichzeitigem Auftreten mehrerer Fehler wird folgende Priorität eingehalten: Break/Parity-Error/Framing-Error/Overrun-Error.

Fehlernummern in *iorecl*:

`$8007` - Time-Out
`$801E` - Framing-Error
`$801F` - Overrun-Error
`$8020` - Parity-Error
`$8046` - Break erkannt

Die Fehlerbedingung Time-Out ist vom Zustand *errlog* unabhängig, und wird nur über die beschriebenen Time-Out Zellen freigegeben.

3.2 Beispiele zum VME-ISER-Server

3.2.1 Beispiel: Initialisierung des VMEbus-Masters

Es wird empfohlen, daß die Initialisierungsroutine des Masters folgende Adressen **einmalig** bestimmt und auf Master-lokalen Zellen ablegt:

CRDADR -- VME-Bus Basisadresse der VME-ISER
TxBUFF -- VME-ISER-relative Adresse der Tx-Kanäle 1-10
RxBUFF -- VME-ISER-relative Adresse der Rx-Kanäle 1-10
PARAn -- VME-ISER-relative Adresse der Parameter-Kanäle 1-10
IRCH -- VME-ISER-relative Adresse auf den Datenbereich des Interrupter-Kanals (*iobuff* (IRCH))
IACK -- Interrupt-Ack-Adresse absolut
TIRTRIG -- Transmit-Interrupt-Trigger_Adresse absolut
RIRTRIG -- Receive-Interrupt-Trigger_Adresse absolut

Der Master sollte die Kanäle der VME-ISER, beginnend mit der Adresse aus ***ANCHOR*** scannen und entweder auf den entsprechenden ASCII-String (*TBUFxy*, *RBUFxy*, *PARAxy* und *Irch*) überprüfen, bzw. über die Zellen *iotyp* und *ioldn* den Kanal bestimmen. Als Next-Pointer ist *iofor* zu verwenden.

3.2.2 Beispiel: Data-Output auf Schnittstelle 2 ohne IRQ

```
TCHACH1   EQU   (1-1)*4           ;Offset Server 1
TCHACH2   EQU   (2-1)*4           ;Offset Server 2
..
TCHACH9   EQU   (9-1)*4           ;Offset Server 9
TCHACHA   EQU   (10-1)*4          ;Offset Server A


          MOVEA.L CRDADR,A0       ;Basisadresse
          MOVE.L  TXBUF2,D0       ;erster Kanal
          BSR     srchbf          ;freien Kanal suchen (s.o.)
          BNE     wait            ;Kein Kanal frei, warten !?
*         In A0 steht jetzt die absolute Adresse des aktuellen
*         Kanals, in D0 die kartenrelative Adresse
          MOVEA.L iobuff(A0),A1   ;rel Adresse Datenbereich
          ADDA.L  CRDADR,A1       ;absolute Adresse
          MOVE.W  #anzdata,D1      ;Anzahl der Datenbytes
          MOVE.W  D1,iorecl(A0)    ;in Header eintragen
          SUBQ    #1,D1           ;wg. DBxx
          MOVEA.L source,A2       ;Pointer auf Sendedaten
loop      MOVE.B  (A2)+,(A1)+     ;Transfer zur VME-ISER
          DBF     D1,loop         ;
          MOVE.W  #0,ioilev(A0)    ;ioilev,ioivec == $0
          MOVE.L  #0,iofnam(A0)    ;Clear fname
          MOVE.W  #$4600,iomode(A0);Output, kein wait
*         VME-ISER-Server aktivieren
          MOVEA.L IRCH,A2         ;Pointer auf Daten-Interrup.
          ADDA.L  CRDADR,A2       ;absolut
          TST.L   TCHACH2(A2)      ;Eintrag frei ?
          BNE     wait            ;Nein, warten ?
          MOVE.L  D0,TCHACH2(A2)   ;rel. Adr. des Kanals eintr.
          MOVE.W  D0,TIRTRIG      ;Write 'any' als Trigger
* ----    fertig    ---
```

3.2.3 Beispiel: Data-Input von Schnittstelle 8

```
RCHACH1   EQU   (1-1)*4+$40            ;Offset Server 1
RCHACH2   EQU   (2-1)*4+$40            ;Offset Server 2
..
RCHACH9   EQU   (9-1)*4+$40            ;Offset Server 9
RCHACHA   EQU   (10-1)*4+$40           ;Offset Server 10

          MOVEA.L CRDADR,A0            ;Basisadresse
          MOVE.L  RXBUF,D0             ;erster Kanal
          TAS    iosema(A0,D0.L)       ;freien Kanal suchen (s.o.)
          BNE    wait                  ;Kein Kanal frei, warten !?
          LEA     0(A0,D0.L),A0
*         In A0 steht jetzt die absolute Adresse des aktuellen
*         Kanals, in D0 die kartenrelative Adresse
          MOVE.W  #anzdata,D1          ;maximale Anzahl der zu
*                                      lesenden Datenbytes
          MOVE.W  D1,iorecl(A0)        ;in Header eintragen
          MOVE.B  #05,ioilev(A0)       ;IRQ-Level = 5
          MOVE.B  #$60,ioivec(A0)      ;IRQ-Vector =$60
          MOVE.W  #$2600,iomode(A0)    ;Input, Ende bei <cr>
          MOVE.L  #0,iofnam(A0)        ;normaler Input
*         VME-ISER-Server aktivieren
          MOVEA.L IRCH,A2              ;Pointer auf Daten-Interrup.
          ADDA.L  CRDADR,A2            ;absolut
          TST.L   RCHACH8(A2)          ;Eintrag frei ?
          BNE     wait                 ;Nein, warten ?
          MOVE.L  D0,RCHACH2(A2)       ;rel. Adr. des Kanals eintr.
          MOVE.W  D0,RIRTRIG           ;Write 'any' als Trigger
*
* ----    Warten bis zum Eintreffen des spez. IRQ's
              MOVE.W  iorecl(A0),D1     ;Anzahl der empf. Daten
              BEQ     exit             ;keine Daten empfangen
              BMI     error
              SUBQ    #1,D1            ;wg. DBxx
              MOVEA.L destin,A2        ;Ziel der Daten
              MOVEA.L iobuff(A0),A1     ;Quelle der Daten, relativ
              ADDA.L  CRDADR,A1        ;Adresse absolut
loop1     MOVE.B  (A1)+,(A2)+          ;Transfer Datenbytes
              DBF     D1,loop1         ;
              MOVE.B  =0,iosema(A0)     ;Kanal freigeben !!
*  ----   fertig --
*
error     ANDI.W  =$7FFF,D1            ;mask Error-Number
              .
              . (Fehlerroutine)
              .
```

3.2.4 Beispiel: Parametrierung von Schnittstelle 1

```
TCHACH1   EQU   (1-1)*4           ;Offset Server 1
TCHACH2   EQU   (2-1)*4           ;Offset Server 2
..
TCHACHA   EQU   (10-1)*4          ;Offset Server 10
txbs      EQU   0                 ;Sollwert Tx_Baud
rxbs      EQU   txbs+1
chrls     EQU   rxbs+1
stpls     EQU   chrls+1
parts     EQU   stpls+1
hnds      EQU   parts+1
txb       EQU   $40               ;Istwert Tx_Baud
rxb       EQU   txb+1
chrl      EQU   rxb+1
stpl      EQU   chrl+1
part      EQU   stpl+1
hnd       EQU   part+1

          MOVEA.L CRDADR,A0       ;Basisadresse
          MOVE.L  PARA1,D0        ;Parameter-Kanal, relativ
          ADDA.L  D0,A0            ;absolute Adresse
          ADDA.L  CRDADR,A0        ;absolute Adresse
          MOVEA.L iobuff(A0),A1    ;Datenbereich Parameter
          ADDA.L  CRDADR,A1        ;absolute Adresse

*         z.B.:
*         Tx-Baudrate auf 300 Baud setzen
*         Rx-Baudrate auf 600 Baud setzen
*         Handshake auf XON/XOFF setzen
          MOVE.B  #7,txbs(A1)     ;Tx-Baud = 300
          MOVE.B  #6,rxbs(A1)      ;Rx-Baud = 600
          MOVE.B  #1,hnds(A1)      ;XON/XOFF Handshake
*         Alle weiteren Parameter unverändert
          MOVE.W  #$4700,iomode(A0);Output-Mode
          MOVE.W  #0,ioilev(A0)    ;Kein IRQ
          MOVE.W  #0,iocmmd(A0)    ;Mode: Init-Parameter
*         Parameter-Kanal in Server-Queue eintragen
          MOVEA.L IRCH,A2         ;Pointer auf Daten-Interrup.
          ADDA.L  CRDADR,A2       ;absolut
          TST.L   TCHACH1(A2)      ;Eintrag frei ?
          BNE     wait            ;Nein, warten ?
          MOVE.L  D0,TCHACH1(A2)   ;rel. Adr. des Kanals eintr.
          MOVE.W  D0,TIRTRIG      ;Write 'any' als Trigger
*  ----   fertig
```

3.3 Anwenderprotokolle

3.3.1 Funktionsbeschreibung

Der Anwender hat die Möglichkeit, für jeden Kanal ein gesondertes Rx-Protokoll oder Rx-Filter zu implementieren. Dazu muß das Protokoll-Programm in einem freien RAM-Bereich der VME-ISER geladen werden (z.B. `$20000... $3FFFF`) und die Einsprungadresse des Anwenderprogramms muß dem lokalen ISER-Server bekannt gemacht werden. Dies geschieht durch Eintragen der Einsprungadresse in die Zelle *ioentr* im Parameter-Kanal des jeweiligen Kanals.

Wird nun vom VME-Master ein Rx-Element mit *iofnam* = `PROT` angefordert, so wird nach dem Speichern der empfangenen Zeichen im Interrupt-Buffer das eingetragene Protokoll ausgeführt, welches die gepufferte Zeichenkette untersuchen, und gegebenenfalls in den geforderten Kanal übertragen kann.

Ist *iofnam* des angeforderten Kanals ungleich `PROT`, so erfolgt ein normaler Datentransfer durch den Standard-VME-ISER-Server.
Ist *iofnam* des angeforderten Kanals gleich `PROT` und ist der Protokoll-Entry *ioentr* nicht versorgt, so wird die Rx-Anforderung ignoriert.

Es ist auf jeden Fall darauf zu achten, das die Grundparametrierung des Kanals über den Parameter-Kanal nicht zu Konflikten mit dem geforderten Protokoll führt (z.B.: Software-Handshake bei binären Protokollen)!

3.3.2 Bedingungen für den Einsatz anwenderspezifischer Rx-Protokolle/Filter

- Anwenderprogramm muß im freien Speicherbereich zwischen `$20000` und `$3FFFE` installiert werden
- die Einsprungadresse der Server-Routine muß in dem jeweiligen Parameter-Kanal auf der Zelle *IOENTR* eingetragen werden
- die Einsprungadresse muß gerade sein
- die letzten 4 Byte vor der Einsprungadresse müssen die ASCII-Id '`PROT`' beinhalten
- wiedereintrittsfester, frei verschieblicher 68000-Code
  Keine Befehle für 68020/30/40!
- keine Software-Traps
- Einschränkungen bei der Verwendung der Register:
  Das Register A1 enthält den Pointer auf die Variablen des jeweiligen Kanals (z.B. *irwp*, *ceaddr*,...).Das Register A3 enthält die Rücksprungadresse. Im Register D0 wird der Status des Protokolls zurückgegeben:
  '0' - Prot. noch nicht beendet
  > 0 - Anzahl der Bytes
  < 0 - z.B. Anzahl der Bytes + Bit 15 gesetzt: CRC-Fehler
  Die Datenregister A2, A4, D1, D2, D4 können verwendet werden. A1 und A3 dürfen nicht verändert werden!

Der Eintritt in das Protokoll erfolgt im Supervisory-Mode, auf Interrupt-Level 5, bzw. Interrupt-Level 7.

3.3.3 Register- und Strukturvereinbarungen

Register	
`A1.L`	Pointer auf die Struktur *irbuf*
`A3.L`	Rückkehradresse
`A2.L/A4.L`	frei
`D0.L/D1.L/D2.L/D4.L`	frei

Bei Austritt aus dem Protokoll über 'JMP(A3)' muß D0.W mit dem Rückgabewert versorgt sein und die entsprechenden Flags im Statusregister müssen gesetzt sein:

Rückgabewerte in D0.W:

D0	Flags	
`'0'`	*'eq'*	Protokoll noch nicht erfüllt, keine weitere Aktion des ISER-Servers
`'$0001','m'`	*'ne','pl'*	Protokoll ohne Fehler beendet, es wurden *m* Zeichen in den Rx-Buffer übertragen: Der VME-ISER-Server gibt den Rx-Buffer an den VME-Master zurück.
`'$8000','m+$8000'`	*'ne','mi'*	Protokoll mit Fehler beendet, es wurden *m* Zeichen in den Rx-Buffer übertragen: Der VME-ISER-Server gibt den Rx-Buffer an den VME-Master zurück.

Datenstruktur *irbuf* (Interrupt-Buffer)

Jedem VME-ISER-Kanal ist eine Struktur *irbuf* zugeordnet, über die der Tx- und der Rx-Transfer abgearbeitet wird. In dieser Struktur werden z.B. die empfangenen Daten abgelegt. Sie besteht im wesentlichen aus 4 Bereichen:

- Pointer und Zähler für den Tx-Betrieb
- Queue für den Tx-Betrieb (32 Einträge)
- Pointer und Zähler für den Rx-Betrieb
- Ring-Puffer für Rx-Betrieb (1024 Byte)

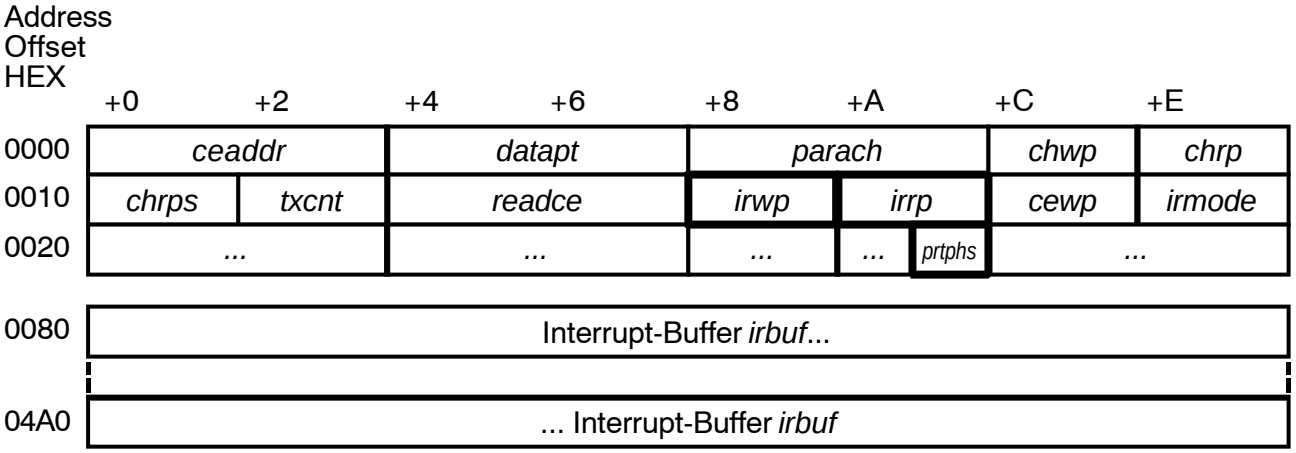


Tabelle 3.3.1: Relevante Zellen des Interrupt-Buffers

Dem Rx-Protokoll genügen im Normalfall folgende Strukturelemente des **Interrupt-Buffers** :

Name	Offset [HEX]	Organisation	Bedeutung
readce	14	Langwort	absolute Adresse des wartenden Rx-Buffers (*iobuff*)
irwp	18	Wort	aktueller Write-Pointer im Datenbereich *irbuf0* (*kann* zwecks Synchronisation vom Protokoll gesetzt werden)
irrp	1A	Wort	aktueller Read-Pointer im Datenbereich *irbuf0* (*muß* vom Protokoll verwaltet werden)
prtphs	2B	Byte	Flags zur Steuerung des Protokolls
irbuf	40		Interrupt-Buffer, Länge: $400

Tabelle 3.3.2: Relevante Strukturelemente des Interrupt-Buffers

Hinweis:
Bis auf die Zellen *irrp*, *prtphs* und gegebenenfalls *irwp* sind alle weiteren Zellen für das Anwenderprogramm read-only!

Außerdem wird aus dem **Datenkanal** (Struktur *iobuff*) ein Pointer benötigt:

Name	Offset [HEX]	Organisation	Bedeutung
iobuff	20 *)	Langwort	Pointer auf Datenbereich

*) Offset im Datenkanal!

Tabelle 3.3.3: Pointer auf Datenbereich

3.3.4 Protokolleinbindung für Rx-Betrieb

Werden Zeichen empfangen, so werden sie auf Interrupt-Ebene aus dem Controller ausgelesen, gegebenenfalls auf Software-Handshake-Zeichen, bzw. auf 'Ende'-Zeichen überprüft und auf dem Rx-Ring-Buffer abgelegt. Danach wird, falls vom Rx-Buffer verlangt, das Anwenderprotokoll ausgeführt. Dies kann nun unter Kenntnis des aktuellen 'Write-Pointers' *irwp* und des (selbstverwalteten) 'Read-Pointers' *irrp* die eingetragenen Zeichen untersuchen. Ist die Protokollbedingung nicht erfüllt, so wird der Rückgabeparameter '0' übergeben, und das Protokoll wird bei den nächsten eintreffenden Zeichen wieder aktiviert.

Ist die Protokollbedingung erfüllt, so wird sich das Anwendungsprogramm um den Transfer der Zeichen in den Rx-Buffer kümmern: Der Pointer auf den wartenden Rx-Buffer steht in der Zelle *readce* und ist vom Strukturtyp *iobuff*. In der Zelle *iobuff* dieser Struktur steht die Anfangsadresse des Datenbereiches, in den die Zeichen zu übertragen sind.

Nach dem Transfer aller Zeichen wird in D0 die Anzahl der gültigen Bytes übergeben; das MSB kann als Flag für ein fehlerhaftes Protokoll verwendet werden. Der VME-ISER-Server gibt anschließend entsprechend der Parametrierung den Rx-Buffer an den VME-Master zurück.
Das Register A1 dient als Basisadresse für die aktuelle Struktur *irbuf* und darf während des Protokolls nicht verändert werden!

Auf Server-Ebene sollte darauf geachtet werden, daß die Abarbeitung des Protokolls zeitlich optimiert ist, da keine weiteren Zeichen während dieser Zeit verarbeitet werden können (Datenverlust!)!

Beispiel:

```
          DC.B     'PROT'
entry:    LEA    irbuf0(A1),A2     ; A2: Pointer auf Rx-Datenbereich
          MOVE.W  irrp(A1),D1      ; letzter Read-Pointer
          MOVE.B  0(A2,D1.W),D0    ; Zeichen aus Rx-Buffer
          CMPI.B  =char,D0         ; Testen des Zeichens
          BNE.S   exit             ; nicht OK
          ADD.W   =len,D1          ; nächster Read-Pointer
          MOVE.L  D1,irrp(A1)
transfer  LEA     0(A2,D1.W),A2    ; Pointer auf Zeichenkette
          MOVEA.L readce(A1),A4    ; Pointer auf Rx-Buffer 'iobuff'
          MOVEA.L iobuff(A4),A4    ; Pointer auf Rx-Datenbereich
          MOVE.W  =len-1,D2        ; Transfer Länge
tloop     MOVE.B  (A2)+,(A4)+      ; Transfer Zeichenkette
          DBF     D2,tloop         ;
          MOVE.W  =len,D0          ; Rückgabewert
          JMP     (A3)             ; zum VME-ISER-Server
exit      MOVEQ   =0,D0            ; Flag: Noch nicht fertig
          JMP     (A3)             ;
```

Bei den Zugriffen auf den Datenbereich im Interrupt-Buffer ist zu beachten, daß es sich hierbei um einen Ringpuffer der Länge 1k-Byte handelt, d.h. alle Pointer müssen Modulo `$3FF` behandelt werden!

Beispiel für Parametrierung (esn-stx/etx-Protokoll):

Bei diesem Protokoll empfiehlt sich folgende Parametrierung:

- *iorecl* = `$0018`
- *iofnam* = `'PROT'`
- *iomode* = `$8700`
- *ioivev, ioilev* = `$00, $00` - kein Interrupt, oder
- *ioivec, ioilev* = vector, level - User defined IRQ

Für die Grundparametrierung über den Parameter-Kanal:

txbs	= `$13`	(115200 Baud)
rxbs	= `$13`	(115200 Baud)
chrls	= `$00`	(8 Bits/Char)
stpls	= `$00`	(1 Stop-Bit)
parts	= `$00`	(keine Parität)
hnds	= `$03`	(kein Handshake)
rtime0s/rtime1s	= `$0000`	oder Time-Out in msec ($ 3 !)

Index

A
ANCHOR 5
Anwenderprotokolle 39
ASCII 5, 11, 31
Asynchron 3

B
Basisadresse 5
Baudrate 20-23
Beispiel-
  Data-Input 37
  Data-Output 36
  Initialisierung 35
  Parametrierung 38
  Speicher allozieren 15
Bits per Character 20
Bits/Char 20
Break 34
Buffer-Kanal 9, 17
Buffer-Pool 12
Bufferzuteilung 15
BUSERROR 5

C
Card-Id 4
character-to-character 21
Clear 25
CPU 3, 5
CPUID 5

D
Datenbereich 6, 8, 10, 11
Datenkanäle-
  Aufteilung 4
  Beschreibung 17
  Verwaltung 12
Datenrichtung 10
Default-Kanal-
  Darstellung 18
  Datenkanal 17
DTACK-Signal 5

E
Empfangsbetrieb 4, 11
Empfangskanal 10
Empfangsmode 8
encodes 22

F
forpointer 6
Framing-Error 34

H
Handshake 20, 31, 32
Handshake-Mode 20, 23
HDLC-Mode 22
Header-
  ioback 9
  iobnum 9
  iobuff 10
  iocmmd 9
  iodata 11
  iofnam 11
  iofor 9
  iofree 11
  ioilev 9
  ioivec 9
  ioldn 10
  iolen 10
  iomode 10
  ioname 9
  ionext 9
  iorecl 10
  iorxln 11
  iosema 9
  iostat 9
  iostio 10
  iotyp 9
  Speichergröße 6
  Versorgung 17

I
Identifier 19
Initialisierung 4, 25
Input-Kanäle 31
Interrupt-
  Betrieb 32
  Buffer 31
  Slave 9
  Vektor 32
Interrupter-Kanal-
  Anzahl 4
  Beschreibung 26
  Darstellung 27
  iobnum 9

Typ 9
iorecl 34
iorxln 11
iotout 32
irbuf 41

K
Kanäle-
Beschreibung 19
Beschreibungen 17
Darstellung 13, 14, 19
Freigabe 15
Header 3
Identifier 8
Kommando 8
Semaphore 8
Status 8, 9
Struktur 6, 7, 10
Strukturdarstellung 17
Typen 3, 8
Übersicht 4
Verkettung 12
Kommandoübergabe 25

M
Multitasking 15
Multiuser 15

O
Overrun-Error 34

P
Parameter-Beschreibung 20
Parameter-Index-
baud 20
chri 20
hndi 20
pari 20
stpi 20
Parameter-Kanäle-
Beschreibung 20
Struktur 19
Typ 9
Parameterstruktur 19
Parametrierung 3, 9, 10
Parity 20, 34
Paritytyp 20, 23
Pointer 6, 8, 9
Polling 4
PROT 39
protoks 22

R
RCHACHx 29
Receive-Errlog 25
Receive-Error-Mode 34
Receive-On 25
Receiver-Status 24
Receiverbaudrate 20, 23
RESET 5, 25
rirtrig 29
ROOT-Pointer 4, 5, 12
rtime0 33
Rx-Buffer 12, 13
Rx-Error 24
Rx-Interrupt 23
Rx-Server 29
Rx-Timeout 23
rxclkmods 21

S
Semaphore 9, 15
Sendekanal 10, 19
Sendemode 8
Slave-Server 4, 15
Speicher 3, 5
Speicherverkettung 6
Status 19
Stopbits 20, 23
Suchverkettung 6
Synchron 3
Synchronisation 19

T
TAS 15
TCHACHx 28
Time-Out 8, 21, 32
tirtrig 28
Transmitter-Status 24
Transmitterbaudrate 20, 23
Triggern 28, 29
ttime 33
Tx-Buffer 12, 13
Tx-Server 28
txclkmods 21

U
UART-Mode 22

Übertragungsmode 4

V
VMEbus-
Interrupt 4
IRQ-Level 8
IRQ-Vektor 8
Master 19
Master-Programm 26, 31
Master-Server 4