VME - CAN2

Intelligent VMEbus/CAN-Bus Interface Board

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 : 0511/633650
Email: pm@esd.h.eunet.de

Beschriebene Platinenversion	CAN2 Rev 0.2

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
1.3.7	Neu: Bei Einstellung der Baudratenkodierschalter auf $F verhält sich die VME-CAN2 passiv auf dem CAN-Bus. **Dies ist auch die neue Standardeinstellung bei Auslieferung der CAN2!**
-	
-	
-	

Weitere technische Änderungen vorbehalten.

Inhaltsverzeichnis

Seite

1. Hardware 3

1.1 Blockschaltbild 3

1.2 Technische Daten 5
1.2.1 Funktion 5
1.2.1.1 Rechnerkern 5
1.2.1.2 High-Speed CAN-Controller 5
1.2.1.3 CAN-Protokoll 5
1.2.1.4 Dezentrale Kommunikation 5
1.2.2 VMEbus-Prozeßrechner 6
1.2.3 CAN-Bus-Interface 6
1.2.4 Allgemeine technische Daten 6
1.2.5 Software 7
1.2.6 Bestellinformationen 7

1.3 Konfiguration-Steckbrücken 9
1.3.1 Default-Einstellung 11
1.3.2 Basisadresse und AM (J1,J2,J6,J7,J8) 13
1.3.3 Speichergröße und DTACK (J4, J11) 18
1.3.4 RESET und Time-Out (J16,J17) 20
1.3.5 CAN-Software-Mode, PIT 68230-Ports PB4, PB5, PB6 (J5) 21
1.3.6 Brücken der CAN-Baugruppen 22
1.3.6.1 Zusätzliche Belegung des Steckers P2 mit den CAN-Bus-Signalen (BR5...BR8) 22
1.3.6.2 Einsatz anderer ’Physical Layer’(BR11, BR12) 23
1.3.6.3 CAN-Monitor-Funktion (J3) 24
1.3.7 Kodierschalter der CAN-Baugruppen (SW1...SW4) . . . 25

1.4 Adreßbelegung der CAN2 27
1.4.1 Adreßbereiche der Baugruppen bei lokalen Zugriffen 27
1.4.2 Adreßbereiche der Baugruppen bei VMEbus-Zugriffen . 27
1.4.3 Adreßbelegung des I/O-Bereiches 28
1.4.4 Speicherbelegung SRAM1 - CAN-Strukturen 29

1.5 Bestückung der SRAM- und EPROM-Bausteine 31
1.5.1 Festlegung der Speichergröße und des Speicherbereichs 31
1.5.2 Zuordnung der Zugriffszeiten der EPROMs und SRAMs zur Taktfrequenz der lokalen CPU 31
1.5.3 Akkupufferung der SRAM-Bausteine 32

1.6 Interrupt-Verarbeitung 33
1.6.1 Belegung der lokalen Prozessorinterrupts 33
1.6.1.1 Belegung der Interrupt-Level 33
1.6.2 VMEbus-Slave-Interrupt-Logik 34
1.6.2.1 Erzeugung eines VMEbus-Interrupts über den PIT 68230 34
1.6.3 Erzeugung eines lokalen Interrupts über den VMEbus 35
1.6.3.1 Programmierung und Rücklesen des Interrupt-Vektors 35
1.6.3.2 Auslösen eines lokalen Interrupts über das PAL GIRL 37

1.7 Watchdog-Schaltung 39

1.8 Externe Erzeugung eines lokalen RESET (VFR) oder ABORT (VFA) 39

1.9 Der PIT 68230 41
1.9.1 Belegung von Port A des PIT 41
1.9.2 Belegung von Port B des PIT 42
1.9.3 Belegung von Port C des PIT 43

Inhaltsverzeichnis

Seite

1.10 CAN-Bus-Interface 45
1.10.1 Sende-und Empfangsschaltung des CAN-Bus-Interface (Physical Layer) 45
1.10.2 Einsatz anderer Physikalischer Layer (Add-On-Option) 47

1.11 NOT AUS-Eingänge und Meldeausgänge 49
1.11.1 Meldeausgänge (Relais) 49
1.11.2 NOT AUS-Eingänge 50

2. Anhang 51

2.1 Steckerbelegungen 51
2.1.1 VMEbus P1 51
2.1.2 I/O-Stecker P2 52
2.1.3 CAN-Bus-Stecker P6, P8 (9pol DSUB) 53
2.1.4 CAN-Bus-Stecker P5, P7 (10pol. Pfostenstecker) . . 54
2.1.6 Serielle Schnittstellen Port1 und Port2 56

2.2 Frontplatte 57

2.3 Stromlaufpläne 59

2.4 Datenblätter 61

2.5 Index 63

1. Hardware

1.1 Blockschaltbild

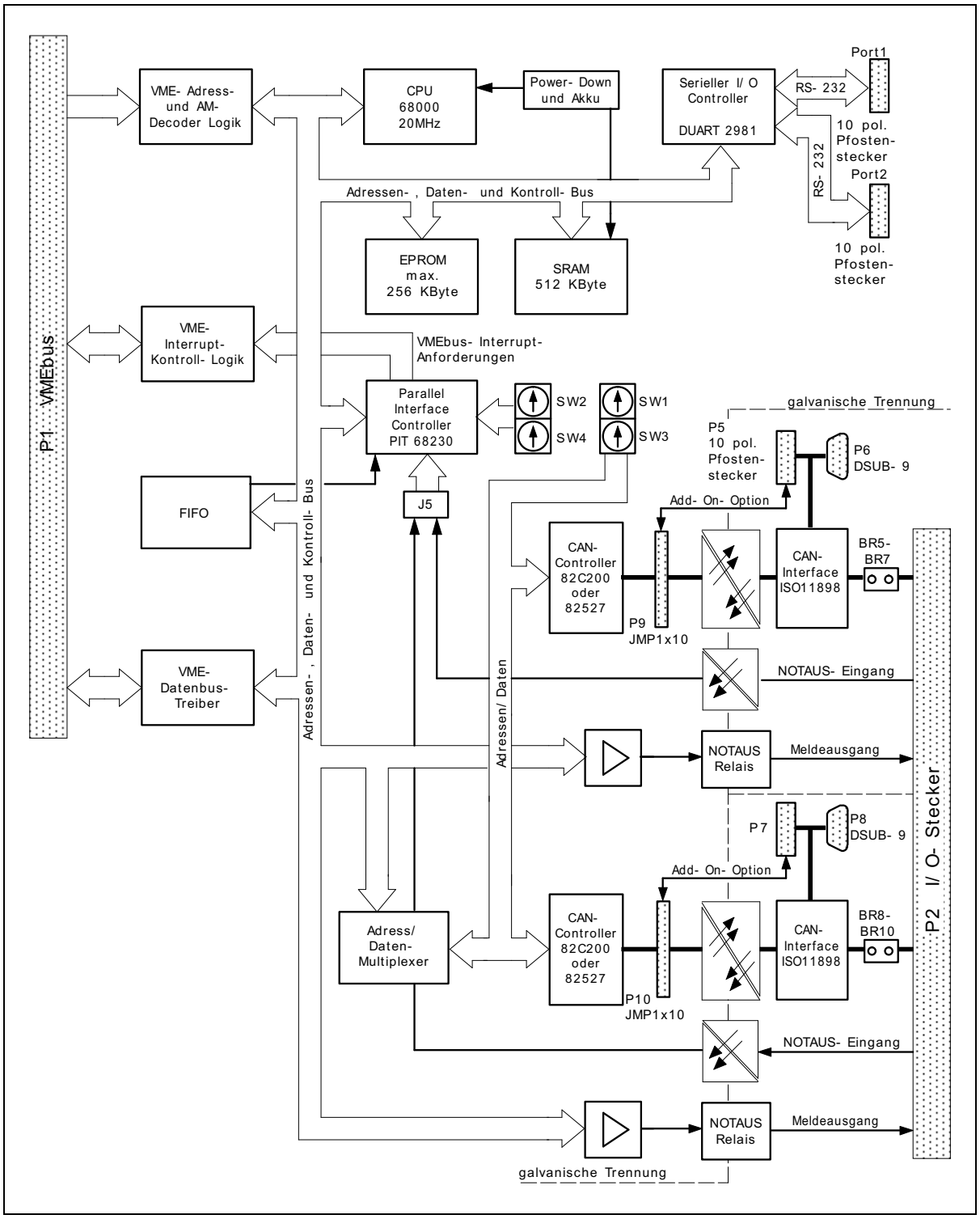


Abb. 1.1: Blockschaltbild des CAN2

1.2 Technische Daten

1.2.1 Funktion

1.2.1.1 Rechnerkern

Die VMEbus-Baugruppe VME-CAN2 ist eine intelligente Schnittstellen-Karte mit 2 unabhängigen CAN-Bus-Kanälen (im weiteren Kanal 1 und 2 genannt) und zwei zusätzlichen seriellen Ports für Programmierung und Service. Die Karte ist im Doppel-Europa-Format mit 4 TE (1 Slot) Breite aufgebaut.

Die VME-CAN2 verfügt über eine lokale MC 68000 CPU mit 20 MHz Taktfrequenz und 512 kB Shared-SRAM sowie eine bidirektionale 4-kWorte-Schnittstelle zur Abarbeitung komplexer CAN-Datenübertragungsprotokolle.

1.2.1.2 High-Speed CAN-Controller

Als Controller-Bausteine sind standardmäßig die BASIC-CAN-Controller 82C200 eingesetzt, optional steht auch eine Ausführung mit dem FULL-CAN-Controller-Baustein 82527 zur Verfügung.

Das physikalische CAN-Layer ist galvanisch vom VMEbus und den anderen lokalen Baugruppen getrennt und gemäß ISO 11898 für eine Bitrate von 1 MBit/s aufgebaut.
Optional sind andere Physical Layers per einsetzbarem Piggyback realisierbar.

1.2.1.3 CAN-Protokoll

Das außerordentlich fehlertolerante CAN-Protokoll (Hamming Distanz = 6) eignet sich hervorragend zum Aufbau von dezentralen I/O-Netzen, in denen ein sicheres und herstellerunabhängiges Kommunikationsprotokoll aufgrund der Flexibilität der Problemstellung notwendig ist.

Die Vorteile des CAN-Protokolls liegen in dem aus der Automobilindustrie "genormten" Übertragungsrahmen, seiner hohen Übertragungsgeschwindigkeit und der Fehlertoleranz gegenüber hochgradig gestörter Umgebung. Zusätzlich ist das CAN-Protokoll selbstarbitrierend und multimaster-fähig, sodaß der Anwender in einem CAN-Netz diverse Sensoren, Aktoren, CAN-I/O-Module oder Prozeßrechner (VMEbus, esd-Xbus) miteinander koppeln kann.

1.2.1.4 Dezentrale Kommunikation

Die VME-CAN2 kann unter Ausnutzung der Firmware direkt zur Kommunikation mit anderen esd-VMEbus- oder esd-Xbus-Rechnersystemen (z.B. esd-XCAN), PC’s oder aber zum Betrieb eines schnellen dezentralen I/O-Netzes mit esd-CAN-Bus-I/O-Modulen verwendet werden.

Die Firmware ist so ausgelegt, das die VME-CAN2 über den VMEbus-Master gesteuert werden kann. Die Kommandos und Daten werden über Software-Schnittstellen (Kommando-Buffer, Daten-Buffer etc.) im Shared-RAM und über ein FIFO mit der CAN2 ausgetauscht.

1.2.2 VMEbus-Prozeßrechner

- 16/32 Bit - Mikroprozessor MC68000/20 MHz
- 512 kByte SRAM und 128 kByte EPROM (Standard)
- VMEbus-Interface A32/D16 und A24/D16 nach Rev. C
- WATCHDOG-Schaltung mit 0.1 oder 1.0 sec Time-Out
- akkugepuffertes RAM
  (min. 35 Tage Standby-Betrieb möglich)
- 2 serielle Schnittstellen RS232C/9600 Baud
- Interrupts auf VMEbus

1.2.3 CAN-Bus-Interface

- 2 CAN-Schnittstellen, jeweils wahlweise Basic-CAN-Controller 82C200 oder Full-CAN-Controller 82527

- Physikalische Layer: differentiell, ISO 11898
  Übertragungsrate 10 kBit/s bis 1 MBit/s
  jeweils galvanische Trennung mit schnellen Optokopplern zum CAN-Bus

1.2.4 Allgemeine technische Daten

VMEbus - Interface	IEEE 1014 / C.1
Data-Transfermode	SADO32 - Slave mit A32/D16-Zugriff SD16 - Slave mit A24/D16-Zugriff
Temperaturbereich	0...50°C Umgebungstemperatur
Luftfeuchtigkeit	max. 90%, nicht kondensierend
Steckverbinder	P1,P2 - DIN 41612-C96 P3,P4,P5,P7 - 10-pol. Pfostenstecker P6,P8 - DSUB 9 (Stifte)
Größe der Platine	160 mm x 233 mm
VMEbus-Einbau	6 HE hoch / 4 TE breit
Gewicht	ca. 550 g
Spannungsversorgung	VMEbus P1: 5V ±5% / 3 A VMEbus P1: +12V ±5% / 0.25A -12V ±5% / 0.10A

Tabelle 1.2.1: Allgemeine technische Daten der CAN2

1.2.5 Software

Ein Basistreiber für die esd-VME-CAN-Baugruppe mit umfangreicher Service- und Diagnose-Firmware on-board ist im Lieferumfang enthalten. Die Firmware ermöglicht den Betrieb des 82C200 als Full-CAN-Controller.

Die Steuerung der VME-CAN2 erfolgt durch den VMEbus-Master.

Zusätzlich kann ein Software-Paket "esd-CMS" (CAN-Message-Specification) für die ISO/OSI Schicht 7 (Anwenderschicht, angelehnt an MAP-MMS) zur Verfügung gestellt werden.

Durch die Shared-RAM- und die FIFO-Schnittstelle von der VMEbus-Seite ist auch ein Einsatz der Baugruppe in unterschiedliche Betriebssysteme wie z.B. VxWorks, OS-9, UNIX, LINUX, pSOS, Windows-NT oder RTOS-UH einfach zu realisieren.

1.2.6 Bestellinformationen

Name	Beschreibung	Bestell-Nr.
VME-CAN2	Intelligentes CAN-Controller Board mit 2 unabhängigen CAN-Schnittstellen, CAN-Controller 82C200	V.1405.02
VME-CAN2	Intelligentes CAN-Controller Board mit 2 unabhängigen CAN-Schnittstellen, CAN-Controller 82527	V.1405.03
VME-CAN2-OS9	C-Treiber für OS-9 als Quellcode	P.1405.50
VME-CAN2-VxW	C-Treiber für VxWorks als Quellcode	P.1405.56
VME-CAN2-RTOS	C-Treiber für RTOS als Quellcode	P.1405.54
VME-CAN2-MD	Zusätzliches Anwenderhandbuch in deutsch	M.1405.20
VME-CAN2-ME	Zusätzliches Anwenderhandbuch in englisch	M.1405.21

Tabelle 1.2.2: Bestellinformationen

1.3 Konfiguration-Steckbrücken

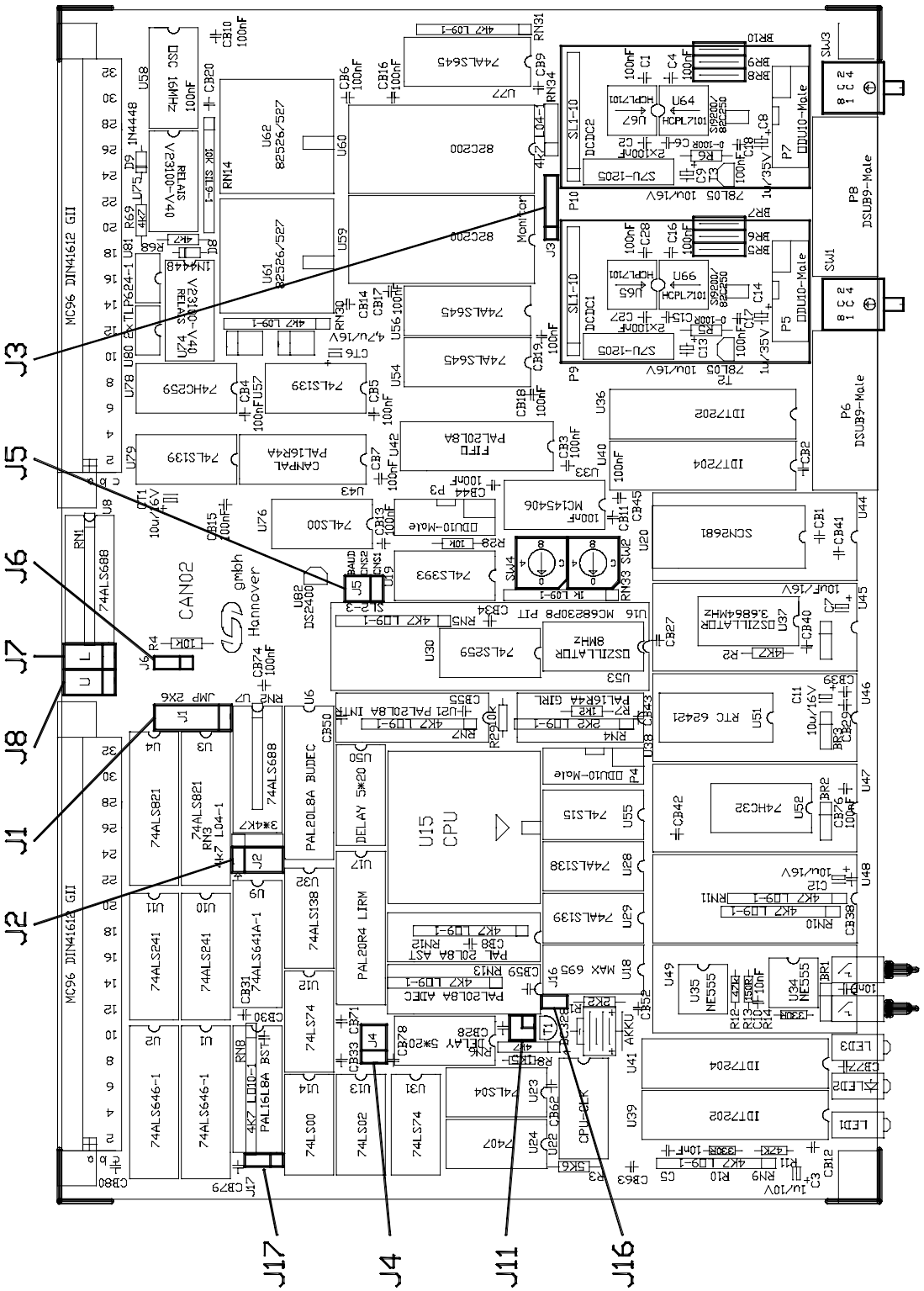


Abb. 1.3.1: Lage der Steckbrücken auf der VME-CAN2

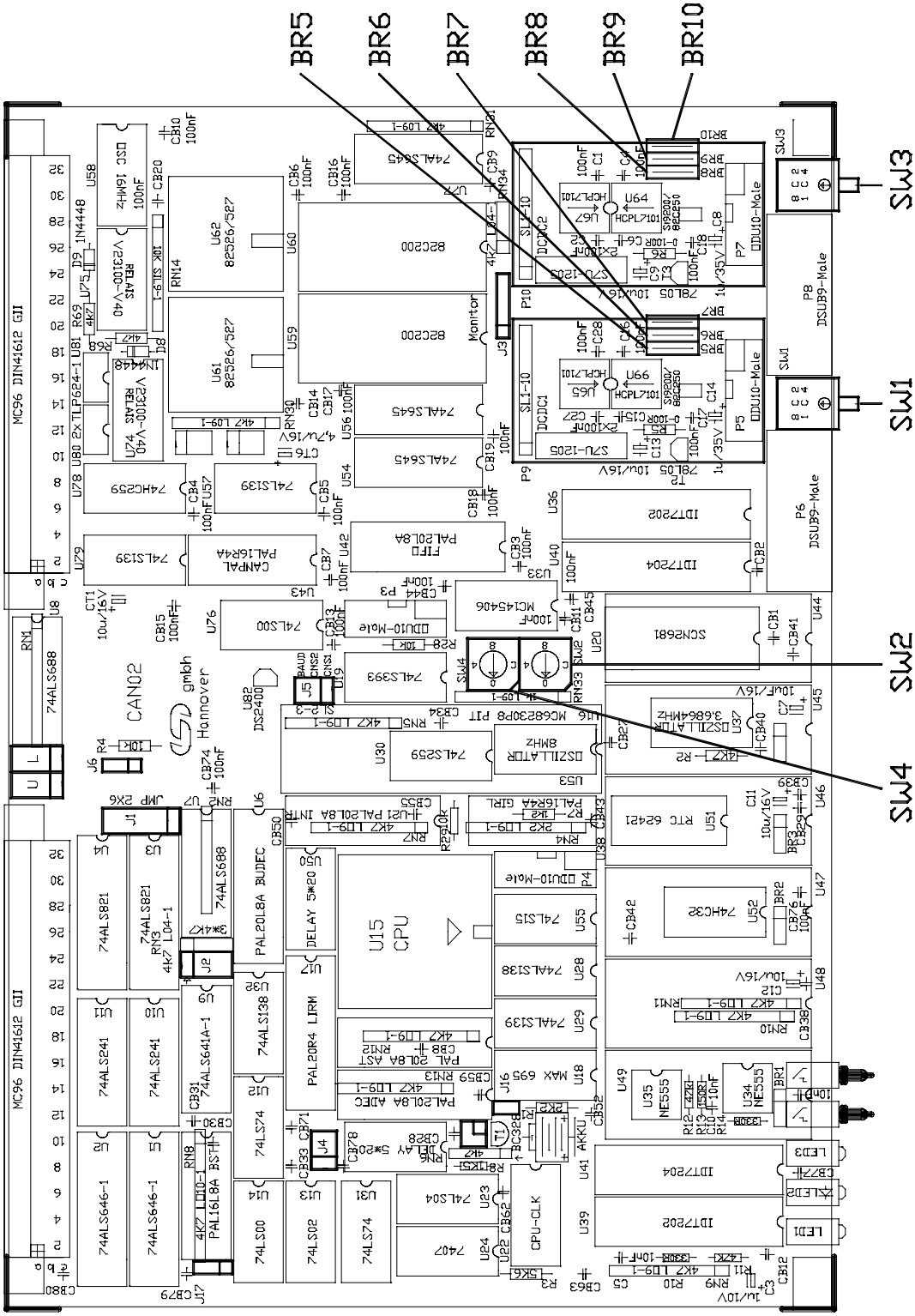


Abb. 1.3.2: Lage der Drahtlötbrücken und der Dip-Switches auf der VME-CAN2

1.3.1 Default-Einstellung

Die jeweilige Default-Einstellung (siehe nachfolgende Tabelle) bei Auslieferung der Karte ist eingetragen.
Die Anordnung der Steckbrücken ist dem Bestückungsplan (Abb.1.3.1) zu entnehmen. Die Anordnung der Drahtlötbrücken und der Dip-Switches ist in Abb. 1.3.2 dargestellt.
Eine gesetzte Steckbrücke entspricht dem ’0’(Low)-Pegel eines Signals.

Default-Einstellung der Steckbrücken J1 bis J17 :

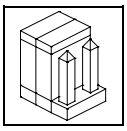
Steckbrücke	Funktion	Einstellung
J1	Address-Modifier (AM), A19	AM2 don’t care, d.h. Zugriff mit Supervisor und User VME-Zugriff A24/D16
J2	Adressen A20...A23	Basisadresse $xxC00000
J3	CAN-Monitor (für Service & Diagnose)	2 x CAN-I/O (wird über Kanal 2 aktiviert)
J4	DTACK-Verzögerung	140 ns bei VME-Zugriff
J5	CAN-Software-Mode, Freigabe der CAN-Kanäle (PIT-Ports)	PIT-Port PB6-offen-> CMS-Software, bzw. CAN2B-Software aktiv Port PB4, PB5 - offen -> CAN-Kanäle freigegeben
J6	A19-Auswertung	A19 wird zur Dekodierung der Basisadresse herangezogen
J7,J8	Adressen A24..A31	offen wegen A24-Mode (J1)
J11	EPROM-Speicher-Größe	2 x 64 kByte EPROM (U46,U49)
J16	Watchdog MAX695	offen (1.0 Sekunden)
J17	VME-RESET	kein VME-Reset bei CPU-Reset

Tabelle 1.3.1: Default-Einstellung der Steckbrücken

Default-Einstellung der Lötbrücken BR5 bis BR12 und der Schalter SW1 bis SW4:

Lötbrücke	Funktion	Einstellung
BR5, BR6, BR7	CAN-Kanal 1 wird zusätzlich über P2 herausgeführt	offen, d.h. P2 nicht mit Kanal 1 belegt
BR8, BR9, BR10	CAN-Kanal 2 wird zusätzlich über P2 herausgeführt	offen, d.h. P2 nicht mit Kanal 2 belegt
BR11, BR12	Piggyback-Anschluß für andere Physical Layer	Standard nach ISO 11898
SW1	CAN-Net-No. für Firmware des CAN-Kanal 1	$0 (= Netz 1)
SW2	Defaulteinstellung Bitrate CAN-Kan. 1	$F: keine Bitrate ausgewählt, d.h. dieser Kanal der CAN2 ist passiv auf dem CAN-Bus
SW3	CAN-Net-No. für Firmware des CAN-Kanal 2	$1 (=Netz 2)
SW4	Defaulteinstellung Bitrate CAN-Kan. 2	$F: keine Bitrate ausgewählt, d.h. dieser Kanal der CAN2 ist passiv auf dem CAN-Bus

Tabelle 1.3.2: Default-Einstellung der Lötbrücken und Schalter



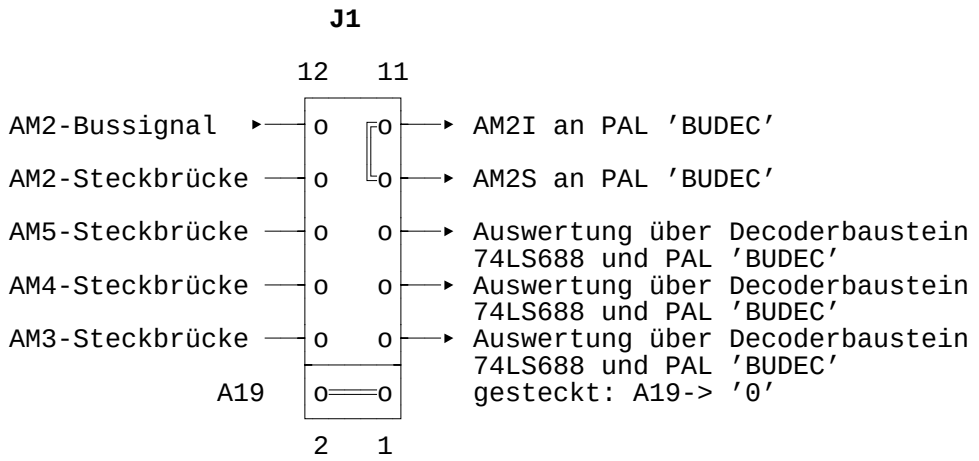
1.3.2 Basisadresse und AM (J1,J2,J6,J7,J8)

Die Address-Modifier (AM) und A19-Selektion auf Steckbrücke J1:

Die CAN2 kann mit den Zugriffsarten A24/D16 (Standard) oder A32/D16 (Extended) betrieben werden.
In der unten dargestellten Default-Stellung wird in der Betriebsart A24/D16 zugegriffen. AM2 wird dabei ignoriert, so daß sowohl im User- als auch im Supervisor-Mode adressiert werden kann.

Außerdem wird mit J1 das Adreßbit A19 gesetzt, falls es über die Steckbrücke J6 zur Ausdekodierung der Basisadresse der CAN2 herangezogen wird (siehe auch Steckbrücke J6). Ist die Steckbrücke J1 1-2 gesetzt, so wird A19 als ’0’ ausdekodiert.

Default-Einstellung: Standard supervisory and nonprivileged data access (A24-Mode), A19



Die Dekodierung der AM-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 Stellung der Steckbrücke J1 1-2 (A19) bleibt in den folgenden Darstellungen unberücksichtigt.

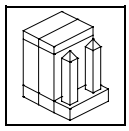
Die AM-Signale des VMEbus werden wie folgt ausgewertet:

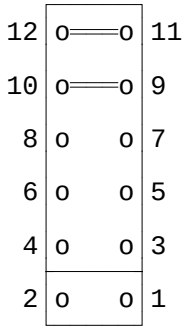
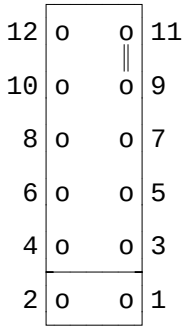
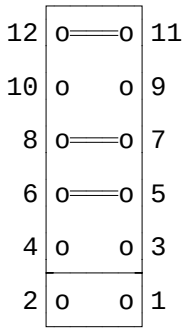
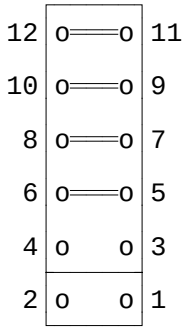
AM-Signal	zugelassene Level	Kodierung über J1
AM0	1	keine Auswahl möglich
AM1	0	keine Auswahl möglich
AM2	0 1 0 und 1	Steckbrücke auf 9-10 und 11-12 Steckbrücke auf 11-12 Steckbrücke auf 9-11
AM3	0 1	Steckbrücke auf 3-4 Steckbrücke 3-4 offen
AM4, AM5	0,0 1,1	die Karte 'antwortet' nur, wenn AM4 und AM5 den gleichen Level haben:
AM4 AM5	0 0	Steckbrücke auf 5-6 Steckbrücke auf 7-8
AM4 AM5	1 1	Steckbrücke 5-6 offen Steckbrücke 7-8 offen

Tabelle 1.3.3: Zugelassene Level von AM-Signalen

Aus der oben angegebenen Tabelle ergeben sich folgende sinnvolle Steckbrückenkombinationen und Adressierungsarten (A19 ist aus Gründen der Übersichtlichkeit in den Darstellungen immer freigelassen worden):

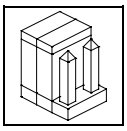
J1	zugelassene AM-Codes							Adressierungsart
	AM5	AM4	AM3	AM2	AM1	AM0	HEX	
12 o═══o 11 10 o o 9 8 o o 7 6 o o 5 4 o o 3 2 o o 1	1	1	1	1	0	1	3D	Standard supervisory data access (A24)



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)

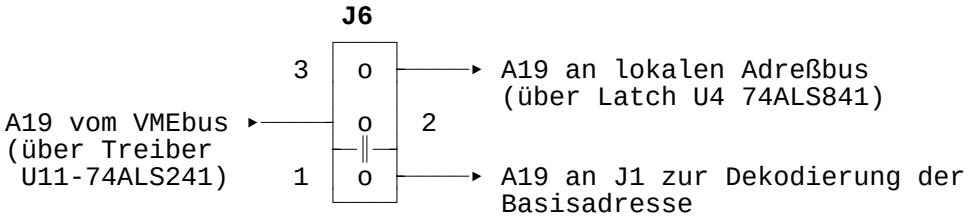
J1	zugelassene AM-Codes							Adressierungsart
	AM5	AM4	AM3	AM2	AM1	AM0	HEX	
12 o o 11 10 o o 9 8 o═o 7 6 o═o 5 4 o o 3 2 o o 1	0	0	1	0	0	1	09	Extended nonprivileged data access (A32) &
	0	0	1	1	0	1	0D	Extended supervisory data access (A32)
12 o - o 11 10 o - o 9 8 o - o 7 6 o - o 5 4 o═o 3 2 o o 1	a	b	0	x	0	1	-	Reserved Anmerkung: Auf Kombinationen mit AM3 = 0 'antwortet' die Karte. Sie sind jedoch lt. VMEbus-Specification ANSI/IEEE SDT1014 'reserviert'.
	! a = b a = 0,1 b = 0,1 x = 0,1 - mögliche Steckbr.-Stellg. siehe oben							

Tabelle 1.3.4: Auswahl der Adressierungsarten (AM-Codes) über J1



Zuordnung des VMEbus Adreßsignals A19 zu der Dekodierlogik der Basisadresse oder zum lokalen Adreßbus mit J6:

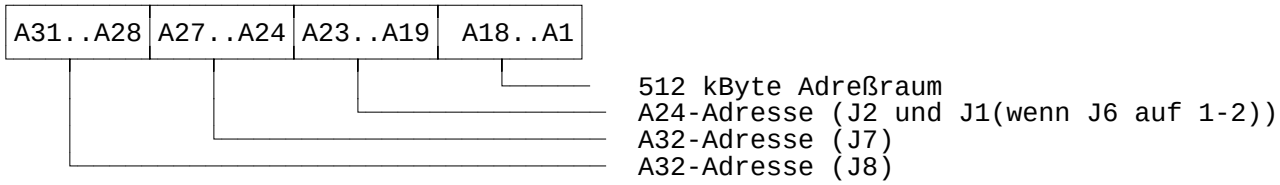
Mit J6 wird festgelegt, ob die Adreßleitung A19 zur Auswertung der Basisadresse der Karte herangezogen, oder mit A1 bis A18 auf den lokalen Adreßbus gelegt wird.
Soll die CAN2 möglichst wenig Adreßraum auf dem VMEbus belegen, empfiehlt es sich A19 mit zur Auswertung der Basisadresse hinzuzuziehen.



Standardbestückung: A19 wird zur Dekodierung der Basisadresse herangezogen

Adreß-Einstellung mit Steckbrücken J1, J2, J7 und J8:

Die Basisadresse der CAN2 ergibt sich wie folgt:



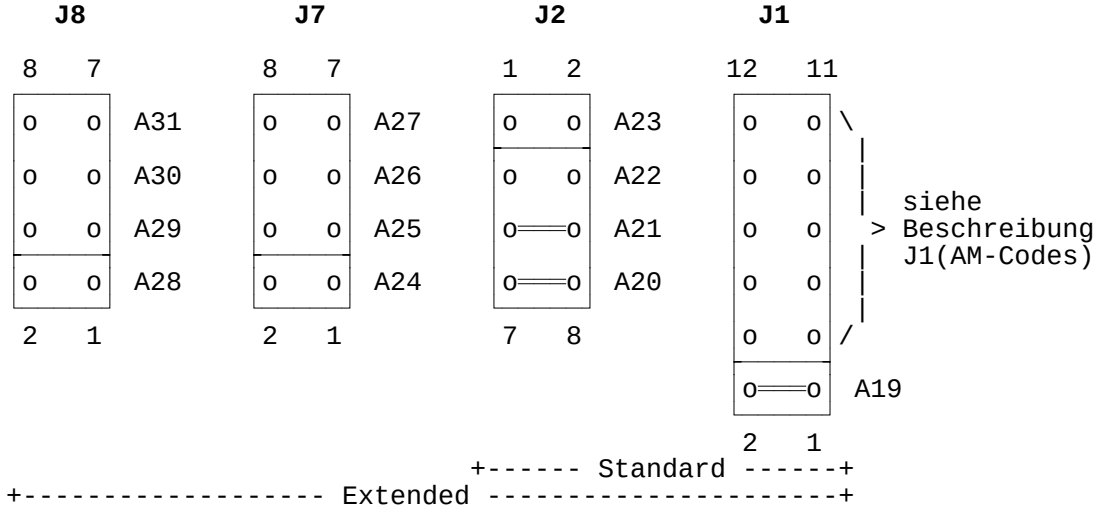
Steckbrückenfelder J7,J8 : Basisadresse für Extended-Addressing-Codes (A24-A31)

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

Steckbrückenfeld J1 : A19 der Basisadresse für Standard-Addressing-Codes (sofern J6 auf 1-2 gesetzt ist)

Wenn mit J1 der A24-Zugriff (Standard Addressing) eingestellt ist, werden die Steckbrückenfelder J7 und J8 nicht ausdekodiert.
Ist mit J1 der A32-Zugriff (Extended Addressing) eingestellt, werden die Steckbrückenfelder J7, J8 <u>und</u> J2 ausgewertet ((A19)A20-A31)!

J1 (1-2) wird nur ausgewertet, wenn die Steckbrücke J6 entsprechend gesetzt ist.



Die Basisadresse der VME-CAN2 ist ab Werk standardmäßig auf
xxC00000 HEX
eingestellt (A24-Mode mit J1 gesteckt).

Bei bis zu vier VME-CAN2-Platinen in einem VMEbus-System, werden die Basisadressen wie folgt eingestellt:

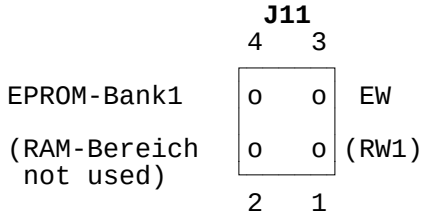
xxC80000 HEX
xxD00000 HEX
xxD80000 HEX

Das auf der Vorseite dargestellte Beispiel zeigt die Steckbrückenbelegung für die Basisadresse xxC00000 HEX.

1.3.3 Speichergröße und DTACK (J4, J11)

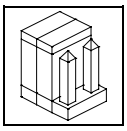
Steckbrückenfeld J11: Festlegung des Speicherbereiches der EPROM-Bank1 (U46, U49)

Die CAN2 kann mit zwei EPROMs von 64 kByte oder 128 kByte Größe bestückt werden. Standardmäßig werden 2x64 kByte bestückt.



	EW gesetzt	EW nicht gesetzt
EPROM 3..4 (**EW**) U46, U49	0C0000-0FFFFF 2x128 kByte z.B.2x27C010	0C0000-0DFFFF 2x64 kByte z.B.2x27C512

Tabelle 1.3.5: Festlegung der EPROM-Baustein-Größen mit J11



Steckbrückenfeld J4 : Verzögerung des DTACK-Signals für VMEbus-Zugriffe nach dem Chip-Select der einzelnen Baugruppen

Anmerkung:

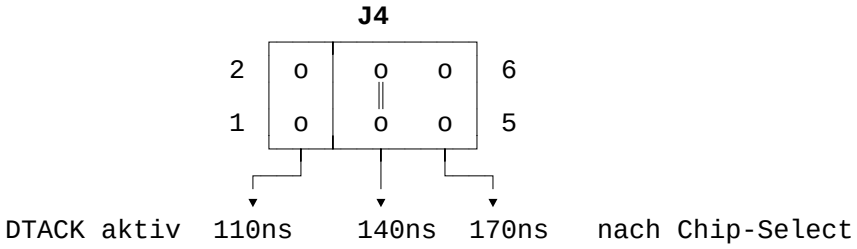
Die effektive Verzögerung des VMEbus-DTACK-Signals nach dem Address-Strobe des VMEbus läßt sich nicht konkret angeben:

Das VMEbus-DTACK-Signal wird nur gesendet, wenn der VMEbus-Master eine Baugruppe auf der CAN2 adressiert hat. Der Zugriff des VMEbus-Masters auf eine Baugruppe kann aber nur erfolgen, wenn er die Kontrolle über den lokalen Daten/Adreßbus der CAN2 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.

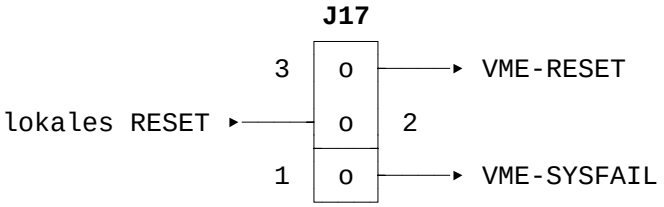
Da diese Faktoren von der Bestückung der Platine bzw. vom Zufall abhängen, wird hier nur die Zeit zwischen dem Chip-Select der Speicher oder Peripheriebausteine und dem DTACK angegeben:



Die DTACK-Verzögerung für Zugriffe der lokalen CPU ist nicht veränderbar. Hier müssen die Zugriffszeiten der Speicherbausteine entsprechend der Taktfrequenz der lokalen CPU ausgewählt werden (siehe auch Abschn.’Bestückung mit SRAM- und EPROM-Bausteinen’).

1.3.4 RESET und Time-Out (J16,J17)

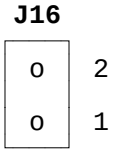
Steckbrückenfeld J17 : Verbindung des lokalen RESET mit VME-RESET, bzw. VME-SYSFAIL



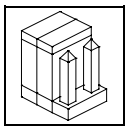
Standardbestückung: Keine Steckbrücke eingesetzt

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

Jumper nicht gesetzt: RESET nach 1,0 sec
Jumper gesetzt : RESET nach 100 msec



Standardbestückung: Keine Steckbrücke eingesetzt



1.3.5 CAN-Software-Auswahl und Kanalabschaltung (J5)

Das Daten-Bit D6 wird zur Auswahl der lokalen CAN2-Software verwendet. Die Auswertung der Steckbrücke ist abhängig von dem eingesetzten CAN-Controller:

Bestückter CAN-Controller	Steckbrücke J5, Pin 5-6 (D6)	Auswertung
2 x 82C200	nicht gesetzt (default-Stellung)	CMS-Protokoll aktiv, kein zyklischer Tx-Transfer möglich
	gesetzt	kein CMS-Protokoll aktiv, zyklischer Tx-Transfer möglich
2 x 82527	nicht gesetzt (default-Stellung)	CAN2B-Software aktiv
	gesetzt	reserviert

Die Bits D5 und D4 können von den NOT-AUS-Eingängen (CAS1, CAS2), die den CAN-Kanälen zugeordnet sind, oder von der Steckbrücke J5 auf ’0’ gesetzt werden. Wird die entsprechende Steckbrücke von J5 gesetzt, so wird der zugehörige CAN-Kanal abgeschaltet.

Die Signale sind jederzeit auf der relativen Adresse **$06017** (als Byte-Zugriff) lesbar.

Steckbrücke J5

				Datenbit	Funktion	offen	gesteckt
6	o	o	5	D6	CAN-Software	siehe oben	
4	o	o	3	D5	setzt CAS2	’1’(*)	’0’
2	o	o	1	D4	setzt CAS1	’1’(*)	’0’

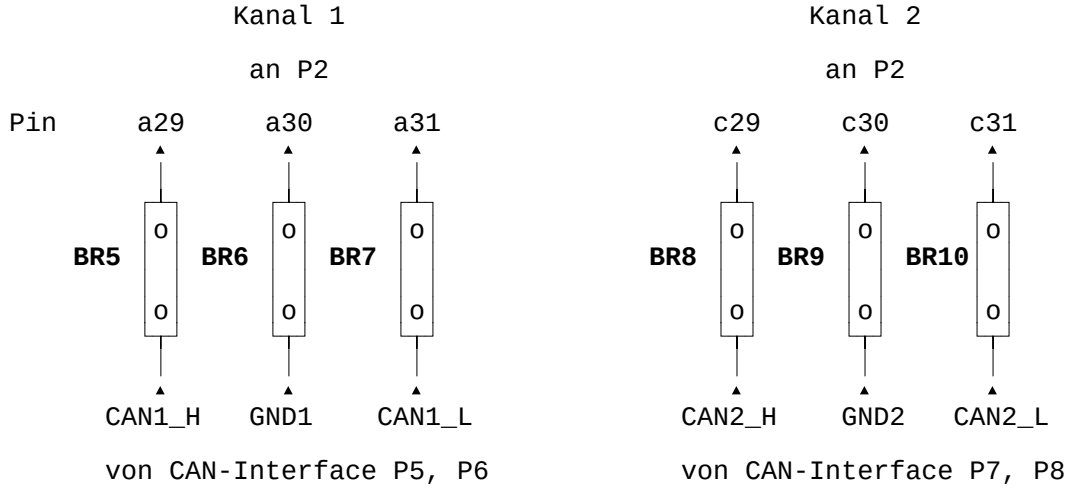
(*) Sind diese Steckbrücken offen, so können D4 und D5 durch Aktivierung der über P2 zugänglichen Eingänge CAS1 und CAS2 auf ’0’ gesetzt werden, andernfalls sind D4 und D5 fest auf ’0’.

1.3.6 Brücken der CAN-Baugruppen

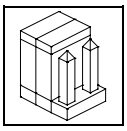
1.3.6.1 Zusätzliche Belegung des Steckers P2 mit den CAN-Bus-Signalen (BR5...BR8)

Über die Drahtbrücken BR5, BR6 und BR7 werden die CAN-Signale von Kanal 1 auf P2 gelegt.
Über die Drahtbrücken BR8, BR9 und BR10 werden die CAN-Signale von Kanal 2 auf P2 gelegt.

Achtung: Durch die Leiterbahnführung quer über die Platine verringern sich die Kriechstrecken zu den galvanisch getrennten Baugruppen auf nur noch 0,2mm!



Standardbestückung: Keine Drahtbrücke gesetzt



1.3.6.2 Einsatz anderer ’Physical Layer’ (BR11, BR12)

Sollen andere physikalische Interfaces als die ISO 11898 kompatible zum Einsatz kommen, kann die CAN2 für jeden Kanal getrennt mit einem Add-On bestückt werden.

Ist die Platine mit dem CAN-Treiber 82C250 oder Si9200 bestückt, müssen die Steckverbinder des CAN-Bus von dem Treiber getrennt werden. Die Trennung erfolgt durch das Entfernen von dünnen Leiterbahnen zwischen den Pads der Lötbrücken BR11 bzw. BR12.
Desweiteren muß der Rx-Ausgang der Optokoppler U66 (Kanal 1) bzw. U64 (Kanal2) des Standard-Interfaces abgetrennt werden, da die Koppler auch bei offenem Eingang ihren Ausgang setzen. Dies kann durch Auslöten der ICs U66 bzw. U64 (HCPL7100) oder durch Trennen von Pin 6 der ICs erfolgen.

Die Lötbrücken befinden sich auf der Lötseite unterhalb der Add-On-Steckplätze.

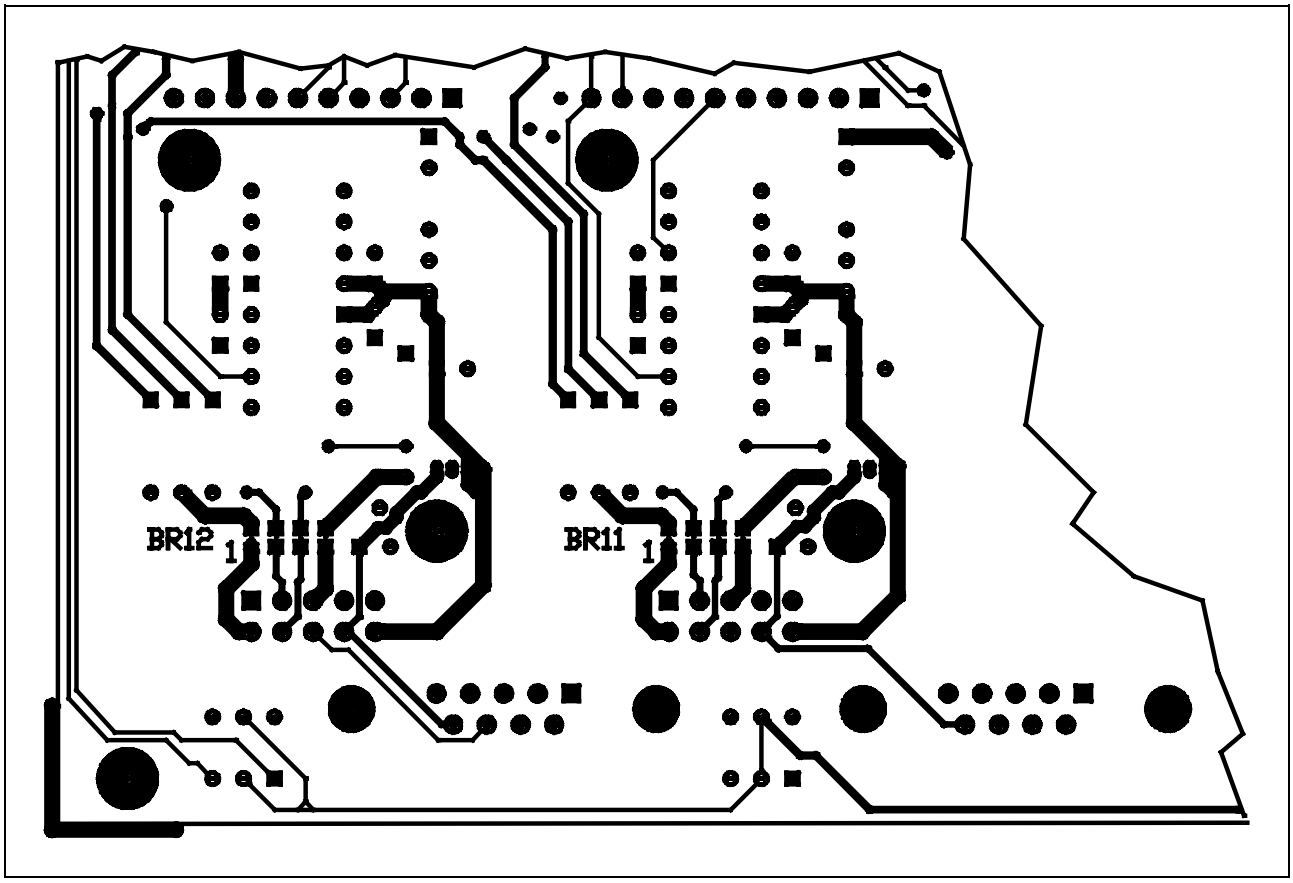
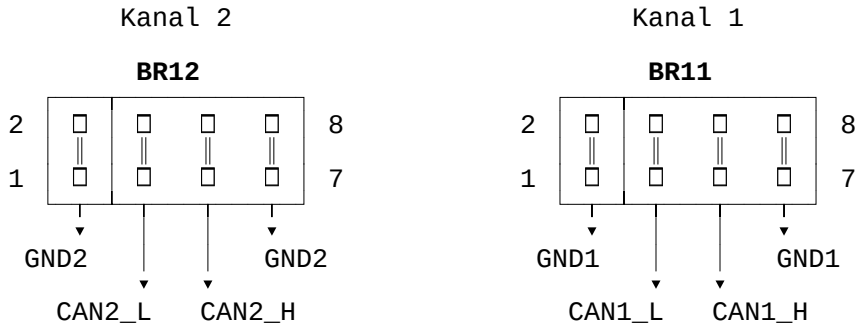


Abb. 1.3.3: Lage der Lötbrücken BR11 und BR12 (Ansicht von Lötseite)



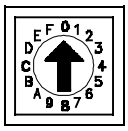
Standardeinstellung: Alle Lötbrücken verbunden.

1.3.6.3 CAN-Monitor-Funktion (J3)

Mit Hilfe der Steckbrücke J3 ist es möglich, die Signale des CAN-Kanal 2 von Kanal 1 überwachen zu lassen.

Funktion	Stellung der Steckbrücke **J3**
Keine Monitor-Funktion: Zwei voneinander unabhängige CAN-Kanäle.	o o o═══o o o 1 2 3 4 5 6
Monitor-Mode	Noch nicht dokumentiert.

Tabelle 1.3.6: Monitor-Mode über Steckbrücke J3



1.3.7 Kodierschalter der CAN-Baugruppen (SW1...SW4)

Mit den Kodierschaltern wird die ’CAN-Net-No.’ und die Bitrate der CAN-Kanäle gesetzt.

	CAN-Net-No.	Bitrate
Kanal 1	SW1	SW2
Kanal 2	SW3	SW4

Tabelle 1.3.7: Zuordnung der Kodierschalter zu den Kanälen

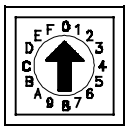
Kodierschalter SW1 und SW3:
Der CAN-Master-Treiber unterstützt bis zu vier CAN2-Boards in einem VME-System. Daraus ergeben sich max. acht verschiedene ’CAN-Netze’, denen über die Kodierschalter eine CAN-Netz-Nummer zugeordnet wird. Mit den Kodierschaltern sind Einstellungen bis zu F (HEX) möglich. Es sind aber nur Einstellunegn von 0 bis 7 zulässig. Die Einstellungen 8 bis F sind reserviert. **Die Einstellung ’0’ kennzeichnet das erste Netz und die Einstellung ’7’ das achte!**

Kodierschalter SW2 und SW4:
Die unten dargestellte Tabelle zeigt die Einstellungen der Bitrate über die Kodierschalter SW2 und SW4 mit Angabe der Registerinhalte der Controller 82C200. Die minimal erreichbaren Leitungslängen ergeben sich aus den ’worst case’-Verzögerungszeiten der eingesetzten Bauteile.

SW2, SW4	82C200-Register		Bit-Rate [kBit/s]	typische Werte der erreichbaren Leitungslänge l_{max} [m]	minimal erreichbare Leitungslänge l_{min} [m]
	BTR0 [HEX]	BTR1 [HEX]			
0	00	14	1000	37	20
1	00	18	666.6	80	65
2	00	1C	500	130	110
3	01	18	333.3	180	160
4	01	1C	250	270	250
5	02	1C	166	420	400
6	03	1C	125	570	550
7	04	1C	100	710	700
8	45	2F	66.6	1000	980
9	09	1C	50	1400	1400
A	4B	2F	33.3	2000	2000
B	18	1C	20	3600	3600
C	5F	2F	12.5	5400	5400
D	31	1C	10	7300	7300
E	00	16	800	59	42
F	Keine Baudrate definiert. Der Kanal ist passiv auf dem CAN-Bus. (Weitere Hinweise siehe folgende Seite.)				

Die Angaben in der Tabelle basieren auf den Grenzwerten des Bit-Timings des CAN-Protokolls, den Laufzeiten des lokalen CAN-Interface und den Laufzeiten des Kabels. Die Laufzeit des Kabels ist mit ca. 5,5 ns/m angenommen. In den Angaben sind weitere Einflüsse z.B. durch fehlende Abschlußwiderstände, den spezifischen Widerstand, die Geometrie des Kabels oder äußere Störeinflüsse bei der Übertragung nicht miteinbezogen!

Tabelle 1.3.8: Setzen der Übertragungsgeschwindigkeit mit SW2 oder SW4



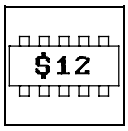
Hinweise zur Einstellung der Baudrate
Bei Auslieferung der VME-CAN2 sind die Baudraten-Kodierschalter beider Kanäle auf $F eingestellt. Die Karte verhält sich daher nach dem Hochlaufen passiv auf dem CAN-Bus.

Grundvoraussetzung für das Senden und Empfangen von Nachrichten auf dem CAN-Bus ist die Einstellung der Baudrate. Sie kann entweder über die Kodierschalter erfolgen (siehe vorangegangene Tablelle) oder mit Hilfe der Software (siehe Software-Handbuch).
Die per Kodierschalter gewählte Baudrate ist als Grundeinstellung zu betrachten, die per Software jederzeit ’überschrieben’ werden kann.

Die Standardeinstellungen der Kodierschalter sind folgende:

	CAN-Net-No. (SW1, SW3)	Bitrate (SW2, SW4)
Kanal 1	0 HEX (= Net-No. 1)	keine
Kanal 2	1 HEX (= Net-No. 2)	keine

Tabelle 1.3.9: Standardeinstellung der Kodierschalter



1.4 Adreßbelegung der CAN2

Die Adreßbelegung der CAN2 teilt sich in 3 Bereiche auf:

1. SRAM (SRAM1 beinhaltet die CAN-Software-Strukturen)
2. I/O-Bereich (beinhaltet lokale Register, PIT 68230, DUART SCN2681, CAN-Controller, u.a.)
3. EPROM

Je nach Zugriff der lokalen CPU oder des VMEbus, ergeben sich verschiedene Adreßbereiche.

1.4.1 Adreßbereiche der Baugruppen bei lokalen Zugriffen

Adresse	Bauteil / Register	siehe auch Kapitel...
FFFFF C0000	EPROM 1 U46/U49	Speichergröße und DTACK
97FFF 80000	I/O-Bereich	Adreßbelegung des I/O-Bereiches
7FFFF 40000	SRAM2 U45/U48	Bestückung der SRAM-Bausteine, Speicherbelegung SRAM1 - CAN-Strukturen
3FFFF 00000	SRAM1 U44/U47	

Tabelle 1.4.1: Adreßbelegung der CAN2 bei lokalen Zugriffen

1.4.2 Adreßbereiche der Baugruppen bei VMEbus-Zugriffen

Durch Verschiebung des I/O-Bereiches in den oberen Adreßraum der RAM-Bank 2 ist dieser SRAM-Bereich vom VMEbus aus nicht mehr zugänglich. Die EPROMs sind vom VMEbus aus ebenfalls nicht zugänglich.

Adresse	Bauteil / Register	siehe auch Kapitel...
7FFFF **78000**	I/O-Bereich	Adreßbelegung des I/O-Bereiches
77FFF 40000	SRAM2 U45/U48	Bestückung der SRAM-Bausteine, Speicherbelegung SRAM1 - CAN-Strukturen
3FFFF 00000	SRAM1 U44/U47	

Alle Adressen sind relativ zur Basis-Adresse angegeben.

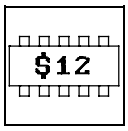
Tabelle 1.4.2: Adreßbelegung der CAN2 bei VMEbus-Zugriffen

1.4.3 Adreßbelegung des I/O-Bereiches

Adresse	Bauteil / Register	siehe auch Kapitel...
20000	WATCHDOG-Register	Watchdog-Schaltung
07001	VFR (**V**MEbus **f**orced **R**eset)	Externe Erzeugung eines lokalen RESET (VFR) oder ABORT (VFA)
07401	VFA (**V**MEbus **f**orced **A**bort)	
06001	PIT 68230	Erzeugung eines VMEbus-Interrupts über den PIT, sowie Ein-und Ausgangstreiber und -Signale an P2
04001	DUART SCN2681	Steckerbelegung Port 1 und Port 2
03001	RTC 62421	Speichergröße und DTACK, sowie Interrupt Verarbeitung
02001	Slave-Interrupt-Vektor-Register	VMEbus-Slave-Interrupt-Logik
01602	16 Bit Fifo Data to VMEbus	Software-Handbuch
01600	16 Bit Fifo Data from VMEbus	Software-Handbuch
01401	Melde-Relais, Add-On-Ansteuerung und Switches SW1 u. SW3	Meldeausgänge, sowie Kodierschalter SW1...SW4
01201	CAN-Kanal 2-Controller Adresse	
01001	CAN-Kanal 1-Controller Adresse	
00001	IRFIRE (Slave-Interrupt-Logik)	VMEbus-Slave-Interrupt-Logik

Alle Adressen sind relativ zur I/O-Basis-Adresse angegeben.

Tabelle 1.4.3: Adreßbelegung des I/O-Bereiches



1.4.4 Speicherbelegung SRAM1 - CAN-Strukturen

Adresse	Register / Inhalte	siehe auch Kapitel...
3FFFF 30000	Betriebssystem, Firmware	-
2FFFF 28000	CAN-Kanal 2-Struktur (Control)	Software
27FFF 20000	CAN-Kanal 2-Struktur (Data)	Software
1FFFF 18000	CAN-Kanal 1-Struktur (Control)	Software
17FFF 10000	CAN-Kanal 1-Struktur (Data)	Software
0FFFF 00000	Betriebssystem	-

Tabelle 1.4.4: Belegung des Speichers SRAM1

1.5 Bestückung der SRAM- und EPROM-Bausteine

1.5.1 Festlegung der Speichergröße und des Speicherbereichs

Die CAN2 wird in der Standardausführung mit SRAM-Bausteinen von 2 x 128 kByte bestückt. Optional ist eine Erweiterung auf 4 x 128 kByte möglich. Hierzu sind lediglich zwei weitere SRAMs in die freien Steckplätze zu setzen. Änderungen an Steckbrücken sind nicht notwendig.

Bitte beachten Sie die unten angegebenen Einschränkungen für die Spezifikation der SRAM-Bausteine und den Haftungsumfang!

Weiterhin werden 2 EPROM-Bausteine mit 64 kByte bestückt. Die Auswahl der Speichergröße der EPROMs wird über das Steckbrückenfeld J11 getroffen. Hier wird auch die Lage des Speicherbereiches festgelegt.

Die Steckbrücke J4, mit der die DTACK-Verzögerung der VMEbus-Zugriffe festgelegt wird, sowie J11 sind bereits in dem Abschnitt ’Speichergröße und DTACK’ beschrieben worden.

1.5.2 Zuordnung der Zugriffszeiten der EPROMs und SRAMs zur Taktfrequenz 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 Zugriffszeit der eingesetzten Speicherbausteine darf bei der CPU 68000/20MHz die in der folgenden Tabelle angegebenen Werte nicht überschreiten!

CPU/Taktfrequenz	EPROM max. Zugriffszeit	SRAM max. Zugriffszeit
68000/20 MHz	120 ns	85 ns

Tabelle 1.5.1: Zuordnung der Zugriffszeit der Speicherbausteine zu der Taktfrequenz der eingesetzten CPU

1.5.3 Akkupufferung der SRAM-Bausteine

Die SRAM-Bausteine und die RTC (Real-Time-Clock) werden bei fehlender Versorgungs-spannung über einen Ni-Cd-Akkumulator gespeist. Der Akku liefert eine Spannung von 2,4V und hat eine Nennkapazität von 30mAh.
Bei der Bestückung der Karte mit 4 SRAM-Bausteinen von je 128KBitx8 werden die Daten im 'Standby-Betrieb' unter folgenden Bedingungen mindestens 35 Tage gehalten:

Lagertemperatur	T = 25°C
max. Entladestrom	I = 6 µA

Bei höheren Lagertemperaturen sinkt die Kapazität des Akkumulators!

Der Endladestrom wird vor der Auslieferung der Karte gemessen. Falls Sie die Bestückung der SRAM-Bausteine nachträglich verändern, stellen Sie sicher, daß der Strombedarf der Speicher nicht über 10µA ansteigt.

Der Hersteller haftet nicht für den Datenerhalt, wenn der Anwender selbst Bestückungsänderungen auf der Platine vornimmt!

Nach 3 Jahren Betrieb der CAN2 ist der Akku (bedingt durch die Dauerladung) so weit gealtert, daß er nur noch 70% seiner Nennkapazität besitzt. Wir empfehlen, den Akku spätestens nach 3 Jahren zu ersetzten.

Für den Akkumulator gelten folgende Temperaturgrenzwerte:

Betriebszustand	T min [°C]	T max [°C]
Laden	0	45
Entladen	-20	50 (60)*
Lagern	-40	50

* Max. zulässig sind 60°C für 24 Stunden.

Tabelle 1.5.2: Temperaturgrenzwerte des Akkumulators

1.6 Interrupt-Verarbeitung

1.6.1 Belegung der lokalen Prozessorinterrupts

1.6.1.1 Belegung der Interrupt-Level

Die Interrupt-Level der lokalen CPU der CAN2 sind wie folgt belegt:

IR-Level	Baugruppe	Typ	Treiber	Freigabe
7	ABORT	Auto Vektor	RTOS	immer frei
6	CAN-Device 1,2	Auto Vektor	Firmware	immer frei
5	DUART-IRQ	Auto Vektor	RTOS	immer frei
4	entfällt	-	-	-
3	PIT-Timer IRQ	User Vektor	-	immer frei
2	GIRL	User Vektor	Mailbox	immer frei
1	FIFO	Auto Vektor	Firmware	immer frei

Tabelle 1.6.1: Belegung der lokalen Interrupt-Level

1.6.2 VMEbus-Slave-Interrupt-Logik

Die VMEbus-Slave-Interrupt-Logik bietet die Möglichkeit, von der CAN2 (PIT68230) 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 PIT68230

Neben dem Timer-Interrupt des PIT (TIRQ*) kann der PIT auch auf Port PC5IRQ einen Interrupt auslösen. Dieser Interrupt löst einen VME-IRQ aus.
Die Interrupts werden über die PIT-Eingänge H1 bis H4 ausgelöst.

Die Eingänge H3 und H4 werden vom CAN-Bus-Interface der Platine gesetzt und von der lokalen Firmware verwaltet. Weitere Informationen zu der Funktion dieser Eingänge finden sich im ’Software-Handbuch’ der CAN2.

Dem Anwender stehen die PIT-Eingänge H1 und H2 für selbstdefinierte Anwendungen zur Verfügung.

Bei einem Flankenwechsel der Eingänge H1 bis H4 generiert der PIT einen Interrupt und gibt in der Interrupt-Routine einen entsprechenden Vektor aus. Das weitere Interrupt-Handling des PIT ist im Datenblatt des PIT68230 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 1.6.2: VMEbus-Interrupt-Level von PIT-Port C0 bis C2 vorgegeben

Der Zugriff (Starten des Interrupts) auf die Signale H1 und H2 erfolgt durch das Anlegen entsprechender Adressen:

Signal	Funktion
IRFIRE*	’Chip-Select’
A1	A1=1 --▸ Hx=1 A1=0 --▸ Hx=0 (x = 1...2)
A2,A3	Auswahl von H1,H2

Tabelle 1.6.3: Zugriff auf die PIT-Eingänge H1 und H2 über Adreßbits

Die Signale H1 und H2 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
$000019	H2	==▸ 0
$00001B	H2	==▸ 1
$00001D	H1	==▸ 0
$00001F	H1	==▸ 1

(Adressen relativ zur I/O-Basis-Adresse.)

Tabelle 1.6.4: Setzen der PIT-Eingänge H1 und H2 über Adreßbits

1.6.3 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.

1.6.3.1 Programmierung und Rücklesen des Interrupt-Vektors

Adresse des Interrupt-Vektors: $002001
(relativ zur I/O-Basisadresse)

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 Interrupt-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

1.6.3.2 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 Interrupt-Signale 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 1.6.5: 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	
$000001 $000003	F0 res. F0 res.	==▸ 0 ==▸ 1	Dieser Interrupt ist für die Firmware reserviert (siehe auch ’Software-Handbuch’ der CAN2).
$000005 $000007	F1 F1	==▸ 0 ==▸ 1	
$000009 $00000B	F2 F2	==▸ 0 ==▸ 1	
$00000D $00000F	F3 F3	==▸ 0 ==▸ 1	

(Adressen relativ zur I/O-Basis-Adresse.)

Tabelle 1.6.6: 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 und Interrupts'). Lokal ist die Funktion bei eingesetzter Firmware 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. 1.0 Sekunden, bei gesteckter Brücke 0.1 Sekunden.
Nachgetriggert wird der Watchdog durch Schreiben oder Lesen auf das WATCHDOG-Register $20000 (siehe Mapping: 'Adreß-Belegung der CAN2').

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

Der Anwender hat die Möglichkeit, über den VMEbus auf der CAN2 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 I/O-Basisadresse der CAN2) aktiviert:

RESET: $07001

ABORT: $07401

Wird ein RESET ausgelöst, bricht die lokale CPU alle laufenden Programme ab. Das lokale System wird neu initialisiert. Alle aktuellen Daten des Betriebssystems gehen verloren.
Nach einem ABORT bricht die CPU ebenfalls alle laufenden Programme ab. Die Parameter und Daten des Betriebssystems bleiben jedoch erhalten.

Die aktuellen Parametersätze der Firmware sind im geschützten Teil des SRAMs abgelegt und bleiben in jedem Fall erhalten.

1.9 Der PIT68230

Über die PIT-Ports werden auf der CAN2 das Konfigurationsfeld J5 gelesen, ein Teil der VMEbus-Interrupt-Logik gesteuert, die Platinen-Identifier-Nr. eingelesen, der Zustand der FIFOs ausgewertet und zwei der DIP-Switches gelesen.

1.9.1 Belegung von Port A des PIT

Über Port A werden die Switches 2 und 4 eingelesen.
Die Eingangsbits von Port A können über das Port A-Alternate-Register gelesen werden.

Adresse: 006015 HEX (Alternate-Register)
(relativ zur I/O-Basisadresse)

Bit	Ein/Aus	Name	Funktion - siehe Abschnitt...
PA0	Eingang		Kodierschalter SW2 für die CAN-Kanal1-Bitratenselektion
PA1	Eingang		
PA2	Eingang		
PA3	Eingang		
PA4	Eingang		Kodierschalter SW4 für die CAN-Kanal2-Bitratenselektion
PA5	Eingang		
PA6	Eingang		
PA7	Eingang		

Tabelle 1.9.1: PIT68230-Port A

1.9.2 Belegung von Port B des PIT

Port B dient zur Steuerung von optionalen Add-Ons für andere physikalische Interfaces, zum Lesen der NOT AUS-Eingänge (CANS1, CANS2), zum Einstellen des CAN-Software-Modes über J5 und zum Auslesen der seriellen platinenspezifischen ID-Nummer.

Der Zugriff auf Port B erfolgt über das Port B-Alternate Register. Dieses Register kann nur gelesen werden.

Adresse: 006017 HEX (Alternate-Register)
(relativ zur I/O-Basisadresse)

Bit	Ein/Aus	Name	Funktion - siehe Abschnitt...	
PB0	n.festg.	D0	reserviert	CAN-Kanal 1: Add-On-Option für zus. Physical Layer
PB1	n.festg.	D1	reserviert	
PB2	n.festg.	D2	reserviert	CAN-Kanal 2: Add-On-Option für zus. Physical Layer
PB3	n.festg.	D3	reserviert	
PB4	Eingang	D4	CANS 1	NOT AUS-Eingang CANS1
PB5	Eingang	D5	CANS 2	NOT AUS-Eingang CANS2
PB6	Eingang	D6	CAN-SW-Mode	siehe Beschreibung J5
PB7	Eingang	D7	Identifier	Platinen-Identifier-Nr.

n.festg. - Datenrichtung ist noch nicht festgelegt (abhängig von Add-On-Typ)

Tabelle 1.9.2: PIT68230-Port B

1.9.3 Belegung von Port C des PIT

Die Port C-Pins werden als Output-Bits zur VMEbus-Interrupt-Erzeugung und als Timer benutzt.

Die Programmierung der einzelnen Bits ist dem Handbuch des PIT68230 zu entnehmen.

Die Registeradressen sind relativ zur Basisadresse des PIT angegeben. (Die Register-Select-Pins RS1...RS5 sind mit den lokalen Adreßbits A1...A5 verbunden. Der Zugriff erfolgt als ’Byte’ auf ungerade Adressen (local LDS=0).)

Bit	Ein/Aus	Name	Funktion - siehe Abschnitt...
PC0	Ausgang	PC0	VMEbus-IRQ-Level, siehe auch: ’Erzeugung eines VMEbus-Interrupts über den PIT68230’
PC1	Ausgang	PC1	
PC2	Ausgang	PC2	
PC3	Ausgang	TIRQ*	Timer-IRQ, s.a. Datenblatt PIT68230
PC4	Ausgang	FIFO FULL*	Input FIFO voll
PC5	Ausgang	PITIRQ*	Auslösen des VMEbus-IRQ, s.a.’Erzeugg.e. VMEbus-Interrupts ü. PIT’
PC6	Eingang	PIACK*	VME-IRQ-Acknowledge, s.a.’Erzeugg.e. VMEbus-Interrupts ü. PIT’
PC7	Eingang	TIACK*	Timer-Interrupt-Acknowledge, s.a. Datenblatt PIT68230

*...Signal ist ’active low’

Tabelle 1.9.3: PIT68230-Port C

1.10 CAN-Bus-Interface

1.10.1 Bitrate

Die CAN2 ist mit zwei unabhängigen CAN-Interfaces gemäß ISO 11898 bestückt. Es sind Übertragungsgeschwindigkeiten von 10 kBit/s bis 1 MBit/s einstellbar.

Die Bitrate wird über die Kodierschalter SW2 und SW4 oder per Software eingestellt. Die programmierte Bitrate besitzt dabei die höhere Priorität. Weitere Informationen hierzu finden sich im Software-Handbuch der CAN2.

Eine Übersicht der möglichen Bitrate ist der Beschreibung der Kodierschalter zu entnehmen.

1.10.2 Sende-und Empfangsschaltung des CAN-Bus-Interface (Physical Layer)

Als CAN-Bus-Transceiver werden auf der Platine z.B. der Si9200 oder der 82C250 eingesetzt. Optional sind andere Physical Layers per einsetzbarem Piggyback realisierbar.

Die Spannungsversorgung der Treiberbausteine ist über DC/DC-Wandler galvanisch vom VMEbus und den anderen lokalen Baugruppen getrennt. Die galvanische Trennung der Signale erfolgt über Optokoppler.

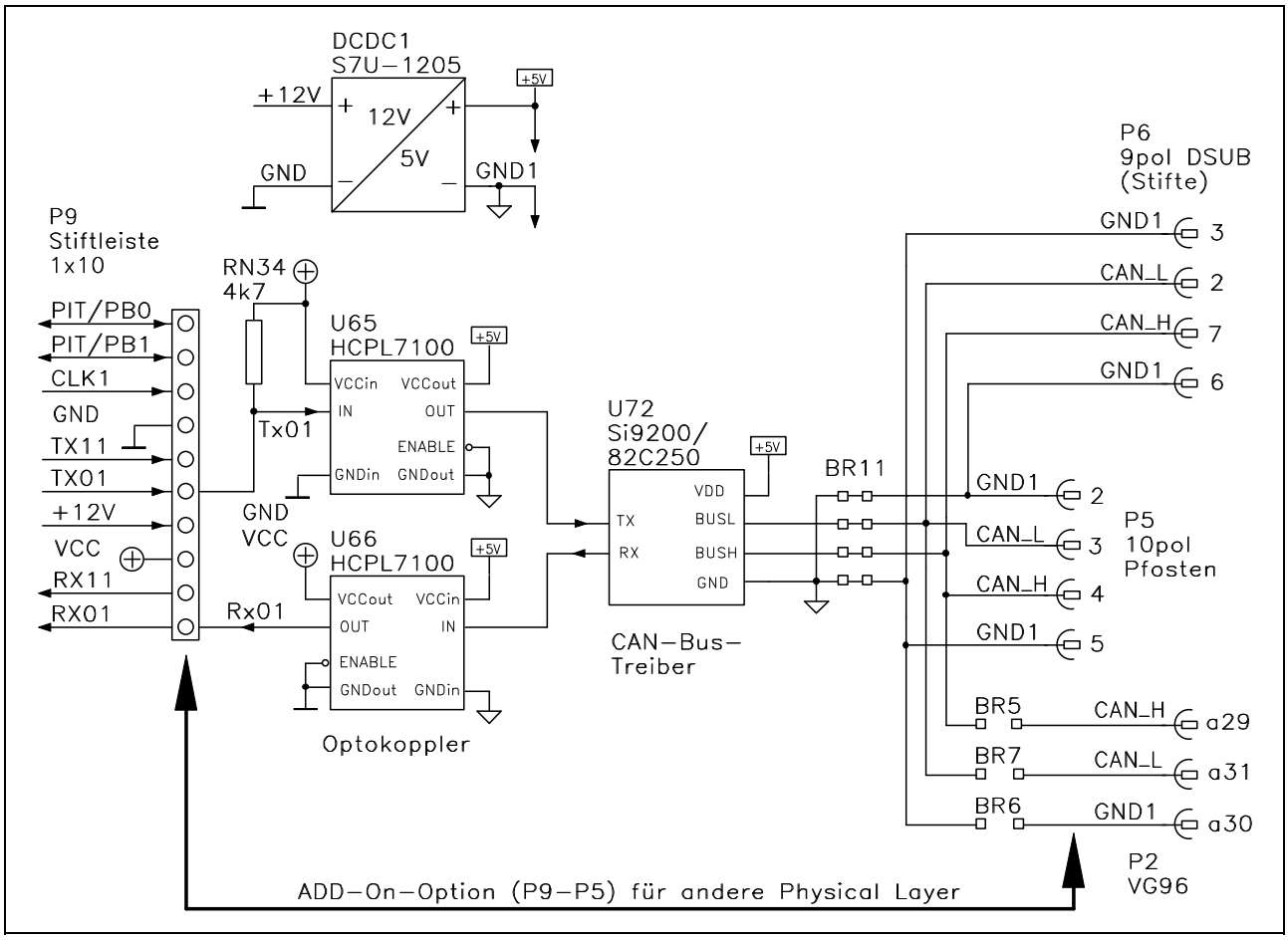
Werden die beiden CAN-Kanäle über die 9poligen DSUB-Stecker P6 und P8 in der Frontplatte angeschlossen, beträgt die Länge der Kriechstrecken zu den galvanisch getrennten Baugruppen ca. 2mm. Daraus ergibt sich eine Bezugsspannung nach VDE0110b §8 - Tabelle 4, Isolationsgruppe C, von U = 60VDC/75VAC. Beim Anschluß der Kanäle über P2 (mit den Drahtbrücken BR5...BR10) bleibt die galvanische Trennung erhalten, die Länge der Kriechstrecken auf der Platine verringert sich jedoch auf nur noch 0,2mm! (Ein Bezug zu VDE0110b §8 ist bei diesem Maß nicht mehr möglich.)

Hinweis:	**Die beiden CAN-Netze sind absolut galvanisch getrennt. Um zu verhindern, daß zwischen den beiden freien Potentialen (auch gegenüber dem VMEbus-Potential) zu großen Spannungsdifferenzen auftreten, muß das CAN_GND-Potential der Netze verbunden und an *einem* Punkt geerdet werden (PE)!**

Von der Frontplatte aus sind die beiden CAN-Kanäle über 9polige DSUB-Stecker (Stifte) zugänglich (Kanal 1 - P6, Kanal 2 - P8).

Außerdem sind die Pfostenstecker P5 (Kanal 1) und P7 (Kanal 2) mit den CAN-Bus-Signalen belegt. Von diesen Steckern aus können Flachbandkabel direkt 1:1 auf 9polige DSUB-Stecker (Buchsen) geführt werden.

Die genauen Steckerbelegungen befinden sich im Anhang.




```
+12V, VCC, GND.......Versorgungsspannungen des VMEbus
+5V, GND1 ..........Versorgungsspannung der CAN-Treiber

CAN_L, CAN_H.........Signale des CAN-Bus
Tx01, Rx01...........Signale des CAN-Controllers 82C200 (hier U59), die auf den
                     CAN-Bus geführt werden

Tx11, Rx11...........Signale des CAN-Controllers 82C200 für zukünftige Anwendungen
                     (Add-On)
PIT/PB0, PIT/PB1.....PIT68230-Ports für zukünftige Anwendungen (Add-On)
CLK1.................Takt für zukünftige Anwendungen (z.B. serielle Speicher) auf
                     Add-On; erzeugt von Latch 74HC259 (U78)
```

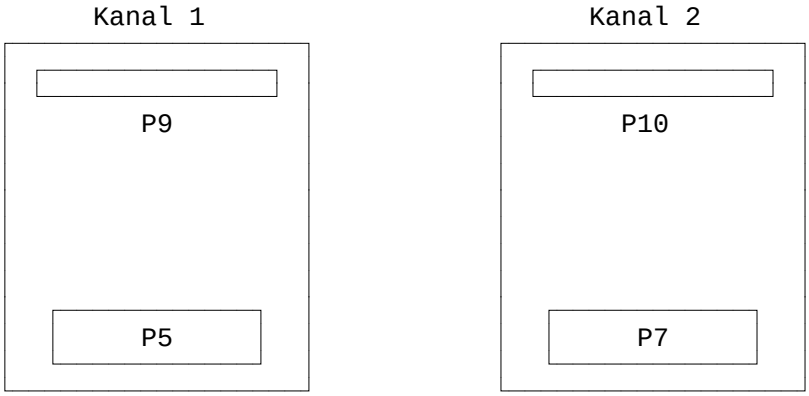
Abb. 1.10.1: Funktionsschaltbild des CAN-Bus-Interface (Beispiel: Kanal 1)

1.10.3 Einsatz anderer Physikalischer Layer (Add-On-Option)

Die VME-CAN2 bietet die Option, den CAN-Bus mit anderen Physical Layers zu betreiben. Für jeden Kanal ist zu diesem Zweck ein Add-On-Steckplatz vorhanden, der eine Platine mit der gewünschten Schaltung aufnehmen kann.

Ist die Platine mit den CAN-Treibern 82C250 oder Si9200 bestückt, müssen die Steckverbinder des CAN-Bus von den Treibern getrennt werden. Die Trennung erfolgt durch das Entfernen von dünnen Leiterbahnen zwischen den Pads der Lötbrücken BR11 bzw. BR12.

Desweiteren muß der Rx-Ausgang der Optokoppler U66 (Kanal 1) bzw. U64 (Kanal2) des Standard-Interfaces abgetrennt werden, da die Koppler auch bei offenem Eingang ihren Ausgang setzen. Dies kann durch Auslöten der ICs U66 bzw. U64 (HCPL7101) oder durch Trennen von Pin 6 der ICs erfolgen.



Die elektrische Verbindung der Add-Ons erfolgt über die Stiftleisten P9 und P10, und die Pfostenstecker P5 und P7 (sofern P5 und P7 nicht für Flachkabelanschlüsse verwendet werden). Die Pinbelegung und -beschreibung findet sich im Anhang im Kapitel ’Steckerbelegungen’.

Die folgende Abbildung gibt die maximal zulässigen Abmessungen des Add-Ons an. Zu beachten ist, daß die Bestückung des Add-Ons den Freiräumen oberhalb (siehe VMEbus-Spezifikation) und unterhalb des Add-Ons gerecht wird. Da die Maße dieser Freiräume von der Länge der gewählten Leiterplatten-Verbinder abhängen, muß der Anwender selbst auf die Einhaltung der Bauhöhen achten.

Die Mindestlänge der Stifte der Steckverbinder P5, P7, P9 und P10 beträgt incl. Kunststoffsockel 8mm. Die Stifte der Steckverbinder haben einen quadratischen Querschnitt von 0,635mm².

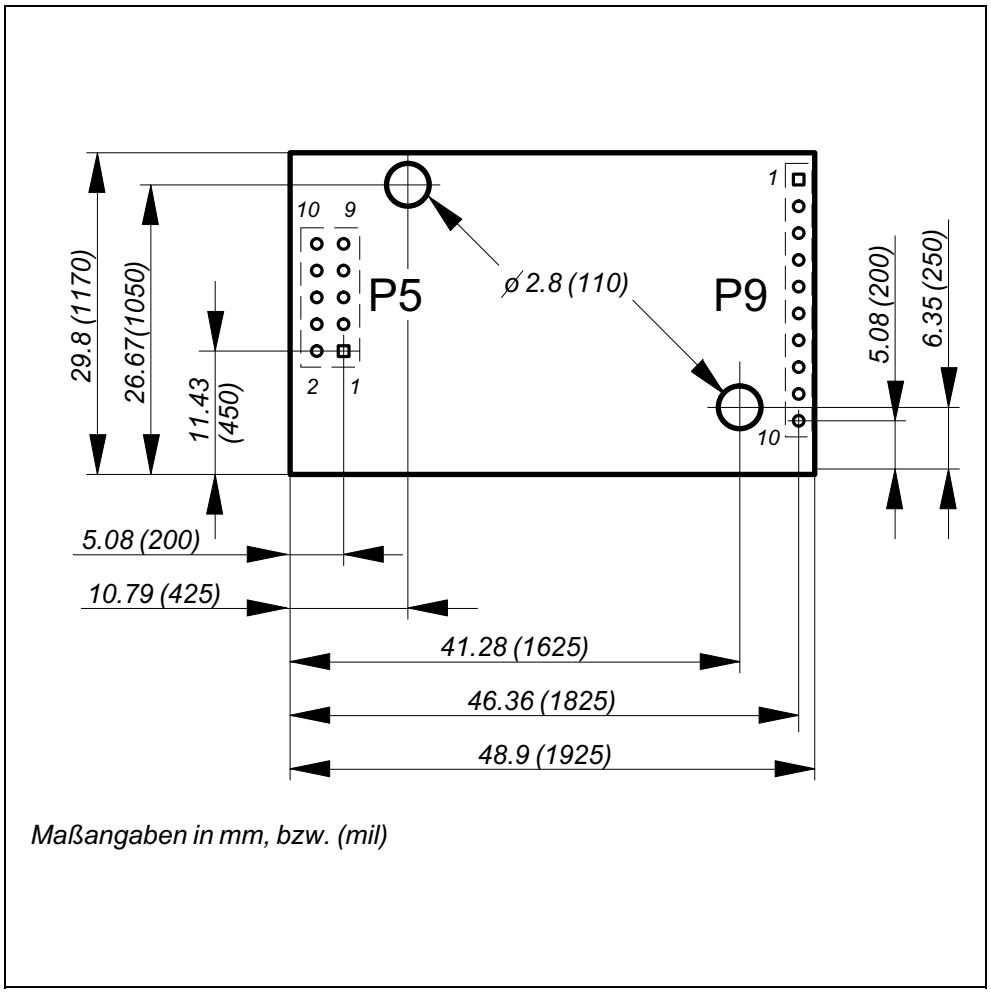


Abb. 1.10.2: Einbaumaße des Add-Ons (Draufsicht)

Für die Befestigung des Add-Ons mit 2,5mm-Schrauben stehen zwei Bohrungen zur Verfügung.

1.11 NOT AUS-Eingänge und Meldeausgänge

Über kanalweise getrennte NOT AUS-Eingänge und Meldeausgänge läßt sich die CAN2 in eine NOT AUS-Kette integrieren.

1.11.1 Meldeausgänge (Relais)

Jeder Kanal verfügt über ein Relais, das im stromführenden Zustand einen auf den Stecker P2 geführten Kontakt schließt. Über die Software kann der Kontakt z.B. beim Auftreten eines Notfalls geschlossen werden.

Die Relais werden geschaltet, indem das Datenbit D7 über die folgenden lokalen Adressen gesetzt wird (Byte-Zugriff):

Kanal	1	2
Relais	U74	U75
Pins P2	c17, c18	c23, c24
lokale Adresse	01401 (HEX)	01403 (HEX)
Kontakt öffnen (Byte, schreiben)	00 (HEX)	00 (HEX)
Kontakt schließen (Byte, schreiben)	FF (HEX)	FF (HEX)

(Adressen relativ zur I/O-Basisadresse.)
Nach einem RESET sind die Meldeausgänge geöffnet.

Tabelle 1.11.1: Setzen der Meldeausgänge

Belastbarkeit der Relaiskontakte (Gleichspannung):

Schaltspannung max. 100V
Schaltstrom max. 0,5A
Schaltleistung max. 10W

1.11.2 NOT AUS-Eingänge

Die über Optokoppler galvanisch getrennten NOT AUS-Eingänge sind auf die Ports PB4 (Kanal1) und PB5 (Kanal2) des PIT68230 geführt. Die Auswertung erfolgt über die Software. (siehe auch Kap. 'Der PIT68230')
Die Eingänge sind über den Stecker P2 zugänglich (Steckerbelegung im Anhang).

Wird die Steckbrücke J5 gesetzt, so wird der entsprechende NOT AUS-Eingang des PIT68230 auf 'LOW' gezogen. Der CAN-Kanal ist abgeschaltet. An den NOT AUS-Eingängen anliegende Signale werden nicht ausgewertet.

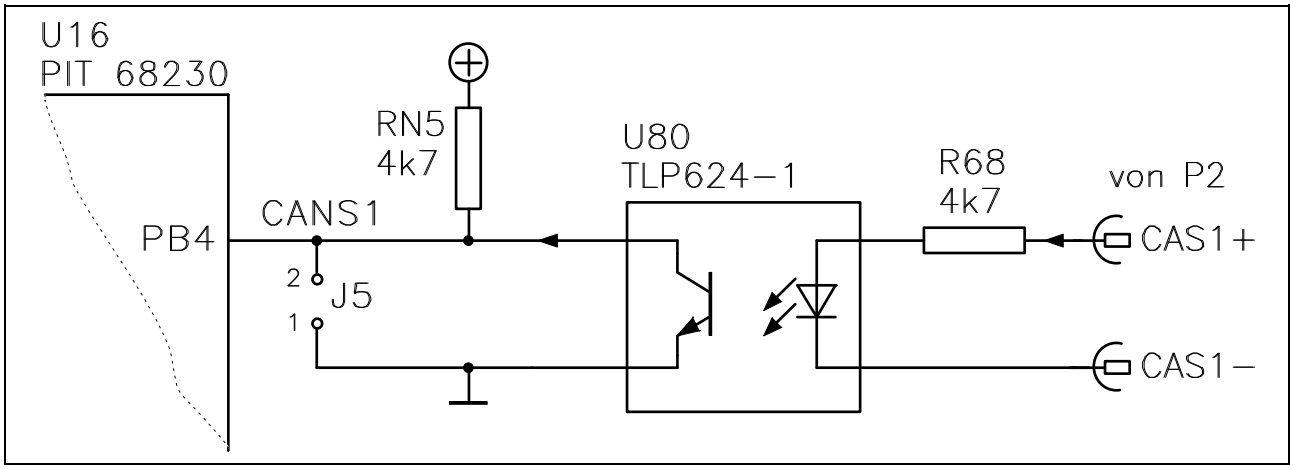
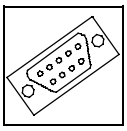


Abb. 1.11.1: Schaltung der NOT AUS-Eingänge (Beispiel: Kanal 1)

Eingangspegel der PIT68230-Ports	Eingangsspannung der NOT AUS-Eingänge
High	0 bis ca. 2,5V DC
Low	9V bis 30V DC

Tabelle 1.11.2 Eingangsspannung der NOT AUS-Eingänge



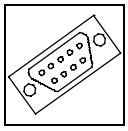
2. Anhang

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	+ 5V STAN	+ 12V
32	+ 5V	+ 5V	+ 5V

Messerleiste nach DIN41612 Bauform C96/a+b+c
Signal with *: Active Low
Imax per Pin : 1.0 A



Steckerbelegung

2.1.2 I/O-Stecker P2

Pin	Reihe a	Reihe b	Reihe c
1	-	+5V	-
2	-	GND	-
3	-	-	-
4	-	ADDR 24	-
5	-	ADDR 25	-
6	-	ADDR 26	-
7	-	ADDR 27	-
8	-	ADDR 28	-
9	-	ADDR 29	-
10	-	ADDR 30	-
11	-	ADDR 31	-
12	-	GND	-
13	-	+5V	-
14	-	-	CAS1-
15	-	-	CAS1+
16	-	-	-
17	-	-	NOTAUS 1-2
18	-	-	NOTAUS 1-1
19	-	-	-
20	-	-	CAS2-
21	-	-	CAS2+
22	-	GND	-
23	-	-	NOTAUS 2-2
24	-	-	NOTAUS 2-1
25	-	-	-
26	-	-	-
27	-	-	-
28	-	-	-
29	CAN1_H P2	-	CAN2_H P2
30	GND1 P2	-	GND2 P2
31	CAN1_L P2	GND	CAN2_L P2
32	-	+5V	-

Messerleiste nach DIN41612 Bauform C96/a+b+c
Imax per Pin : 1.0 A

<u>Erklärung der Signale:</u>

-..... not connected

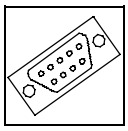
ADDR 24...ADDR31.... VMEbus-Adreß-Signale
+5V, GND..... VMEbus-Versorgungsspannung

CAS1-, CAS1+..... NOTAUS-Eingang Kanal1
CAS2-, CAS2+..... NOTAUS-Eingang Kanal2
(siehe auch Abschn. ’NOTAUS-Eingänge’)

NOTAUS 1-1..... Meldeausgang Kanal 1
NOTAUS 1-2..... Meldeausgang Kanal 2
(siehe auch Abschn. ’Meldeausgänge’)

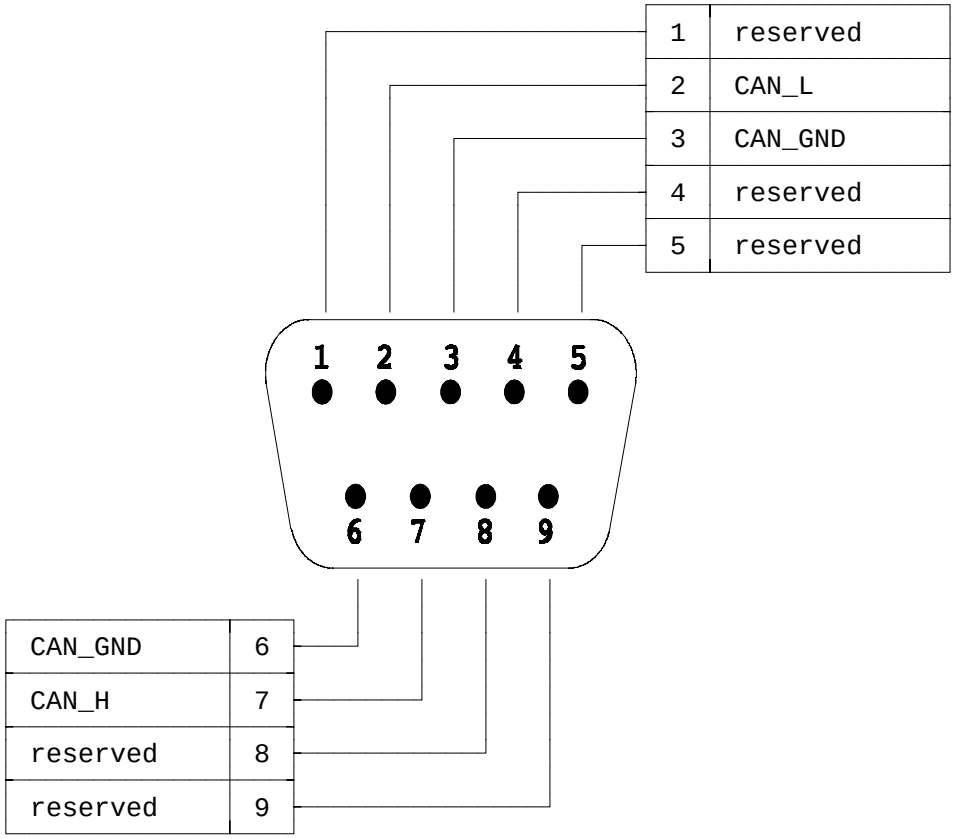
Sind die Drahtlötbrücken BR5 bis BR10 gesetzt, ist P2 mit den Signalen des CAN-Bus belegt:

CAN1_H P2, GND1 P2, CAN1_L P2 -> Kanal1
CAN2_H P2, GND2 P2, CAN2_L P2 -> Kanal2



2.1.3 CAN-Bus-Stecker P6, P8 (9pol DSUB/Stifte)

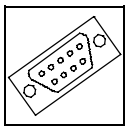
Der Stecker P6 ist mit dem CAN-Kanal 1 und der Stecker P8 mit Kanal 2 belegt.
Die Stecker sind in der Frontplatte der CAN2 montiert.



CAN_L, CAN_H...CAN-Signalleitungen

CAN_GND .....Bezugspotential des Physical Layers

reserved .....reserviert für zukünftige Anwendungen



2.1.4 CAN-Bus-Stecker P5, P7 (10pol. Pfostenstecker)

Der Stecker P5 ist mit dem CAN-Kanal 1 und der Stecker P7 mit Kanal 2 belegt. Die Stecker befinden sich hinter den DSUB-Steckern der Frontplatte direkt auf der Platine. Sie können für Flachbandkabelanschlüsse (z.B. auf 9pol. DSUB Buchse) oder für optionale Add-Ons (andere Physical Layers) genutzt werden.

Signal	Pin		Signal
reserved	1	2	CAN_GND
CAN_L	3	4	CAN_H
CAN_GND	5	6	reserved
reserved	7	8	reserved
reserved	9	10	VCC

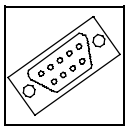
10poliger Pfostenstecker

CAN_L, CAN_H...CAN-Signalleitungen

CAN_GND .....Bezugspotential des Physical Layers

VCC.....Versorgungsspannung des Physical Layers

reserved .....reserviert für zukünftige Anwendungen



2.1.5 Add-On-Steckverbinder P9, P10 (10pol. Stiftleiste 1x10)

Die Stecker P9 (Kanal 1) und P10 (Kanal 2) nehmen in Verbindung mit den Pfostensteckern P5 und P7 die optionalen Add-Ons für andere Physical Layers auf.

Die Lage der Stecker ist in dem Kapitel ’Einsatz anderer Physikalischer Layer’ dargestellt.

Kanal 1

P9

Pin	Signal
1	PIT/PB0
2	PIT/PB1
3	CLK1
4	GND
5	TX11
6	TX01
7	+12V
8	VCC
9	RX11
10	RX01

Stiftleiste 1x10

Kanal 2

P10

Pin	Signal
1	PIT/PB2
2	PIT/PB3
3	CLK2
4	GND
5	TX12
6	TX02
7	+12V
8	VCC
9	RX12
10	RX02

Stiftleiste 1x10

Signalbeschreibung:

Übergeordnet:

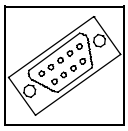
+12V, VCC, GND.....Versorgungsspannungen des VMEbus (=lokale Versorgung)

Kanal 1 (P9):

TX01, RX01.....Signale des CAN-Controllers 82C200 (hier U59), die auf den CAN-Bus geführt werden
TX11, RX11.....Signale des CAN-Controllers 82C200 für zukünftige Anwendungen (Add-On)
PIT/PB0, PIT/PB1.....PIT68230-Ports für zukünftige Anwendungen (Add-On)
CLK1.....Takt für zukünftige Anwendungen (z.B. serielle Speicher) auf Add-On; erzeugt von Latch 74HC259 (U78)

Kanal 2 (P10):

TX02, RX02.....Signale des CAN-Controllers 82C200 (hier U60), die auf den CAN-Bus geführt werden
TX12, RX12.....Signale des CAN-Controllers 82C200 für zukünftige Anwendungen (Add-On)
PIT/PB2, PIT/PB3.....PIT68230-Ports für zukünftige Anwendungen (Add-On)
CLK2.....Takt für zukünftige Anwendungen (z.B. serielle Speicher) auf Add-On; erzeugt von Latch 74HC259 (U78)



Steckerbelegung

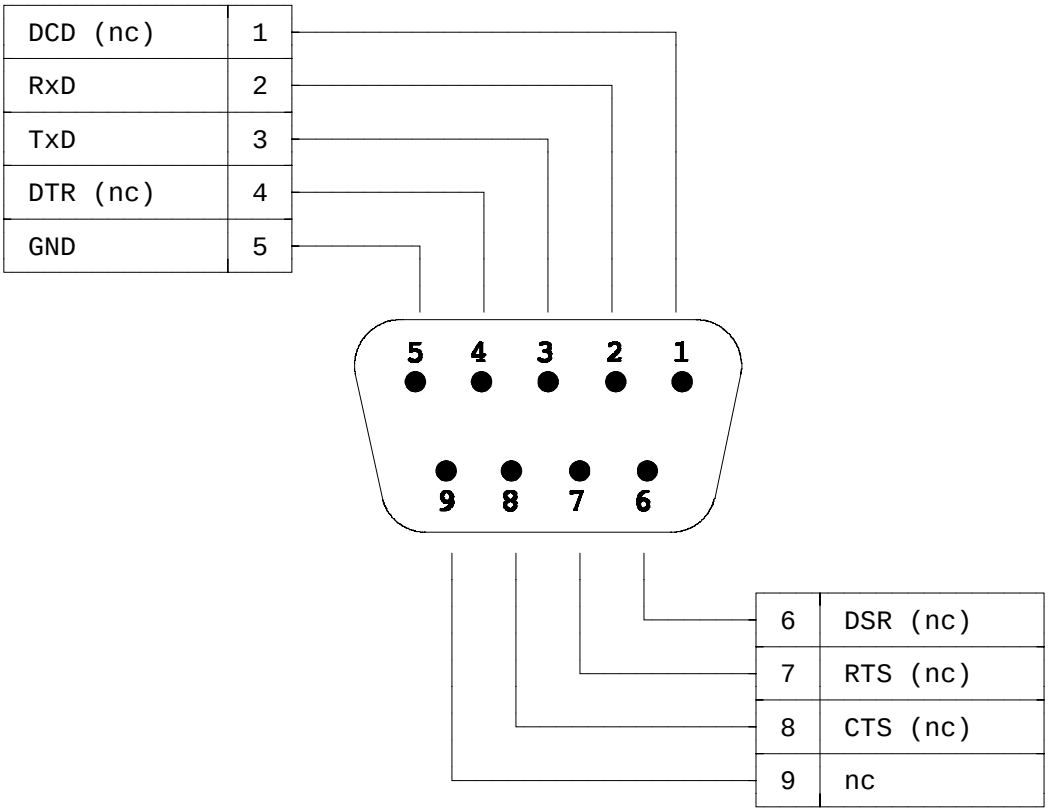
2.1.6 Serielle Schnittstellen Port1 und Port2

Die beiden seriellen Schnittstellen entsprechen RS-232C.
Beide Ports sind nur mit Rxd, TxD und GND belegt.

Nach einem RESET sind die Schnittstellen wie folgt eingestellt:

- XON/XOFF-Handshake
- 8 Bit Data
- 1 Stop Bit
- No Parity
- Televideo-Terminal-Mode
- 9600 Baud

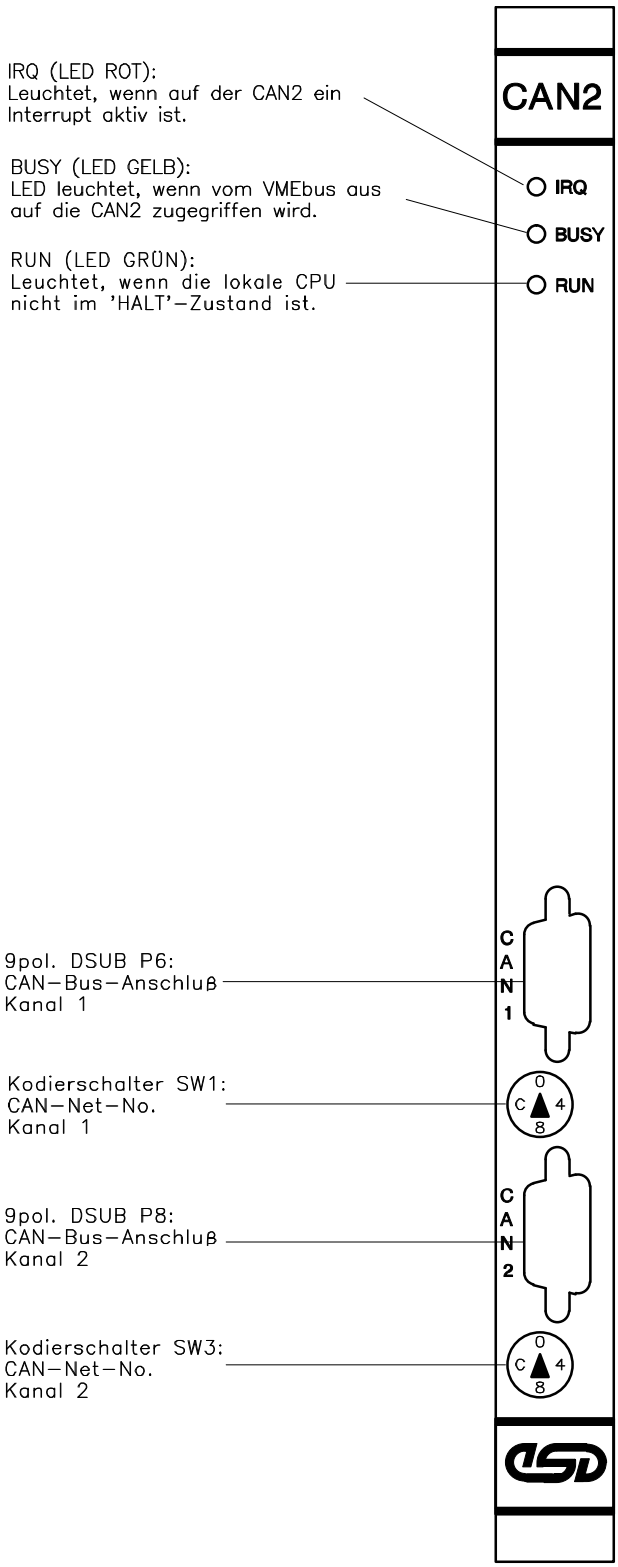
Port 1 (A1) und Port 2 (A2) sind jeweils auf einen 10poligen Pfostenstecker (P3 bzw. P4) geführt, die direkt mit einem Flachbandkabel 1:1 an eine 9-polige DSUB-Buchse für den Anschluß von Terminal oder Laptop verbunden werden können.



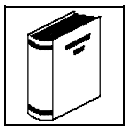
<u>Signalbeschreibung</u> (9polige DSUB-Buchse):

nc.....not connected
DCD..... Data Carrier Detect (nc)
RxD.....Receive Data (Eingang)
TxD.....Transmit Data (Ausgang)
DTR.....Data Terminal Ready (nc)
GND.....Signal Ground (Masse)
DSR.....Data Set Ready (nc)
RTS.....Request To Send (nc)
CTS.....Clear To Send (nc)

2.2 Frontplatte



2.3 Stromlaufpläne



2.4 Datenblätter

2.5 Index

82C200 5, 6, 25, 46, 55

A
ABORT 28, 33, 39
Add-On 23, 28, 42, 45-47, 55
Adreßbelegung 27-29
Address-Modifier 11, 13
Adressen 11, 27, 28, 34, 35, 37, 39, 43, 49
Akku 32
AM-Codes 14-17, 16
Anhang 45, 47, 50, 51

B
BASIC-CAN 5, 6
Basisadresse 11, 13, 17, 18, 35, 39, 41-43, 49
Bestellinformationen 7
Bitrate 5, 12, 25
Bitratenselektion 41
Blockschaltbild 3
BR1 51
BR10 12, 22, 52
BR11 12, 23, 47
BR12 12, 23, 47
BR2 51
BR3 51
BR5 12, 22, 52
BR6 12, 22
BR7 12, 22
BR8 12, 22
BR9 12, 22

C
CAN-Bus-Interface 6, 34, 45, 46
CAN-Controller 5-7
CAN-Interface 6, 22, 45-48, 53, 54
CAN-Knoten 25
CAN-Monitor 11, 24
CAN-Net-No. 12
CAN-Protokoll 5
CAN1_H 22, 23, 52
CAN1_L 22, 23, 52
CPU 5, 11, 19, 27, 31, 33, 35, 39

D
Datenbit 21, 49, 35
Datenblätter 61
Datenbus 35
Dip-Switches 10, 11, 25, 41
Drahtbrücken 22
Drahtlötbrücken 10, 11, 52
DTACK 11, 18, 19, 27, 28, 31, 51
DUART 27, 28, 33

E
Endladestrom 32
EPROM 6, 11, 18, 19, 27, 31

F
Fehlertoleranz 5
FIFO 7, 28, 33, 43
Firmware 5, 7, 12, 29, 33, 34, 37, 39
Format 5
Frontplatte 22, 45, 53, 54, 57
FULL-CAN 5

G
Gewicht 6
GIRL 33-35, 37

H
Hardware 3, 39

I
Identifier 41, 42
Identifier-Nr. 42
Interrupt 6, 28, 33-37, 39, 41, 43
Interrupt-Acknowledge 36, 43
Interrupt-Level 33, 34
Interrupt-Vektor 28, 36, 35
ISO11898 5

J
J1 11, 13-18
J11 11, 18, 31
J16 11, 20
J17 11, 20
J2 11, 13, 17
J3 11, 24
J4 11, 18, 19, 31
J5 11, 21, 41, 42, 50, 56
J6 11, 13, 17
J7 11, 13, 17
J8 11, 13, 17

K
Kodierschalter 25, 28, 41
Kontakt 49
Kriechstrecken 22, 45

L
Lagertemperatur 32
Luftfeuchtigkeit 6

M
MAX695 11, 39
Meldeausgänge 28, 49, 50, 52
Meldeeingänge 49

N
Nennkapazität 32
NOTAUS 21, 42, 49, 50, 52

O
Optokoppler 23, 45, 47, 50

P
P1 6, 51
P10 47, 55
P2 6, 7, 12, 21, 22, 28, 45, 49, 50, 52
P5 6, 22, 45, 47, 54, 55
P6 6, 22, 45, 53
P7 6, 22, 45, 47, 54, 55
P8 6, 22, 45, 53
P9 47, 55
Pfostenstecker 6, 45, 47, 54, 56
Physical Layer 5, 12, 23, 42, 45
Piggyback 5, 12, 45
Pinbelegung 47
PIT68230 21, 27, 28, 34, 41-43, 46, 50, 55
Ports 5, 11, 21, 34, 41, 46, 50, 55, 56
Priorität 35

R
Rechnerkern 5
Relaiskontakte 49
RESET 11, 20, 28, 35, 37, 39, 49, 56
RS232 6, 11, 21
RTC 28, 32
RTOS 7, 33

S
Schaltleistung 49
Schaltspannung 49
Schaltstrom 49
Shared-RAM 7
Software 7, 27-29, 34, 37, 49, 50
Spannungsversorgung 6, 45, 53, 54
Speicher 11, 19, 32, 46, 55
Speichergröße 18, 27, 28, 31
SRAM 5-7, 19, 27, 31, 32
Standby-Betrieb 6, 32
Steckbrücken 9, 11, 17, 21, 31
Steckerbelegungen 45, 47, 51
Steckverbinder 6, 23, 47, 55
Stromlaufpläne 59
Supervisor-Mode 13
SW1 12, 25, 28
SW2 12, 25, 41
SW3 12, 25, 28
SW4 12, 25, 28, 41

T
Taktfrequenz 5, 19, 31
Temperaturbereich 6
Temperaturgrenzwerte 32
Time-Out 20
Timer-Interrupt 34, 43

U
User-Mode 13

V
Versorgungsspannung 32, 45, 46, 52, 54
VFA 28, 39
VFR 28, 39
VME-RESET 11, 20
VME-SYSFAIL 20
VMEbus 5-7, 14, 16-19, 27, 28, 31, 34, 35, 37, 39, 41, 43, 45-47, 51, 52, 55
VMEbus-Interface 6
VMEbus-Prozeßrechner 6

W
WATCHDOG 6, 11, 20, 28, 39

X
XON/XOFF 56

Z
Zugriffsarten 13
Zugriffszeit 31

VME - CAN2

Intelligent VMEbus / CAN-Bus Interface Board

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
D-30165 Hannover

Tel.: 0511/37298-0
FAX : 0511/633650
email: pm@esd.h.eunet.de

beschriebene Softwareversion	cansrv58, cansrv58.cms

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 Vorversion
3.4	Neues Kapitel: 'Tx-Buffer' eingefügt.

Weitere technische Änderungen vorbehalten.

Inhaltsverzeichnis

Seite

1. Einführung . . . 3
1.1 Übersicht . . . 3
1.2 Speicher-Struktur . . . 3
1.3 Mögliche Übertragungsmodes der Identifier . . . 4
1.4 Einführung in die Arbeitsweise der Firmware . . . 4
1.4.1 Rx-Transfers . . . 5
1.4.1.1 Ablaufbeschreibung . . . 5
1.4.1.2 Dauer eines CAN-Rx-Transfers/Zeitdiagramm . . . 6
1.4.2 Tx-Transfers . . . 8
1.4.3 Statusmeldungen der CAN2 und Statusabfrage durch den VMEbus-Master . . . 9

2. Modes der Datenübertragung . . . 11
2.1 Die Strukturfelder CAN Data 1 und CAN Data 2 . . . 11
2.1.1 Funktion und Aufbau . . . 11
2.1.2 Die Bytes eines Daten-Strukturelementes . . . 12
2.2 Die Kontroll-Strukturen CAN CTRL 1 und CAN CTRL 2 . . . 18
2.2.1 Funktion und Aufbau . . . 18
2.2.2 Die Bytes eines CTRL-Strukturelementes . . . 19
2.3 Rx-Modes . . . 22
2.3.1 Empfangen von Rx-Daten ohne Interrupt . . . 22
2.3.2 Empfangen von Rx-Daten mit Interrupt-Meldung auf dem VMEbus . 22
2.3.3 Weitere Optionen für den Empfang von Rx-Daten . . . 22
2.4 Tx-Modes . . . 23
2.4.1 Senden von Tx-Daten ohne Interrupt . . . 23
2.4.2 Senden von Tx-Daten mit Interrupt-Meldung auf dem VMEbus . . 23
2.4.3 Senden von RTR-Frames . . . 23
2.4.4 Weitere Optionen für die Sendung von Tx-Daten . . . 24

3. Spezielle Funktionen . . . 25
3.1 Die Monitor-Buffer . . . 25
3.1.1 Funktion und Aufbau . . . 25
3.1.2 Erklärung der Bytes eines Monitor-Strukturelementes . . . 26
3.2 CAN-Serial-Mode . . . 27
3.2.1 Funktion . . . 27
3.2.2 Struktur der CAN-Serial-Data-Blöcke . . . 27
3.2.3 Erklärung der Bytes eines CAN-Serial-Data-Blocks . . . 28
3.2.4 CAN-Serial-Data-Interrupts . . . 28
3.3 Rx-Buffer . . . 29
3.3.1 Funktion . . . 29
3.3.2 Struktur des Rx-Buffers . . . 29
3.4 Tx-Buffer . . . 31
3.4.1 Funktion . . . 31
3.4.2 Struktur des Tx-Buffers . . . 32
3.5 CMS-Implementierung . . . 34
3.5.1 CMS-Domain-Transfers . . . 34
3.5.1.1 CMS-Buffer . . . 34
3.5.1.2 Ablauf der Transfers . . . 36
3.5.1.2.1 Zu den folgenden Beschreibungen . . . 36
3.5.1.2.2 Initialisierung des CMS-Transfer-Buffers . . . 36
3.5.1.2.3 Client Download . . . 37
3.5.1.2.4 Client Upload . . . 41
3.5.1.2.5 Server Download . . . 45
3.5.1.2.6 Server Upload . . . 49
3.5.2 CMS-Watchdog . . . 53

4. Der Parameter-Buffer . . . 55
4.1 Funktion und Aufbau . . . 55
4.2 Übergabe von Parametern und Kommandos . . . 56
4.3 Beschreibung der Parameter . . . 57
4.3.1 Schreib- und Lesbare Parameter . . . 57
4.3.2 'Nur lesbare' Parameter . . . 58
4.4 Kommandos des Parameter-Buffers . . . 61
4.4.1 Übersicht der implementierten Kommandos . . . 61
4.4.2 Beschreibung der einzelnen Kommandos . . . 63
4.4.2.1 Setzen der Bitrate . . . 63
4.4.2.2 Triggern der Monitor-Funktion . . . 63
4.4.2.3 Verknüpfung von CAN-Identifiern verschiedener Netze . 64

4.4.2.4 Periodisches Senden von Tx-Nachrichten 65
4.4.2.5 Freigabe des Karten-Interrupts mit iovec und iolev . 65
4.4.2.6 Betriebsart (Mode) auswählen 66
4.4.2.7 CMS-Initialisierung 69
4.4.2.8 Freigabe des CMS-Buffers 69
4.4.2.9 Rx-Buffer 69
4.4.2.10 Insert/Release Identifier
(nur mit CAN-Controller 82527) 70
4.4.2.11 Tx-Buffer 70

5. Zusammenfassung der Interrupt-Optionen 71
5.1 Übersicht 71
5.2 Anwenderdefinierte Interrupts/Interrupt Handling 71
5.2.1 Allgemeines/PIT 68230 71
5.2.2 'NOT-EIN'-Interrupt 72
5.2.3 FIFO 'VME-to-CAN-Full'-Interrupt 73
5.2.4 CAN-Server-Interrupt 73
5.2.5 Erkennen der Endebedingung ohne Interrupt (Polling) 74
5.3 Firmware-definierte Interrupts 75

6. Leitfaden zu Konfiguration und Betrieb 77

1. Einführung

1.1 Übersicht

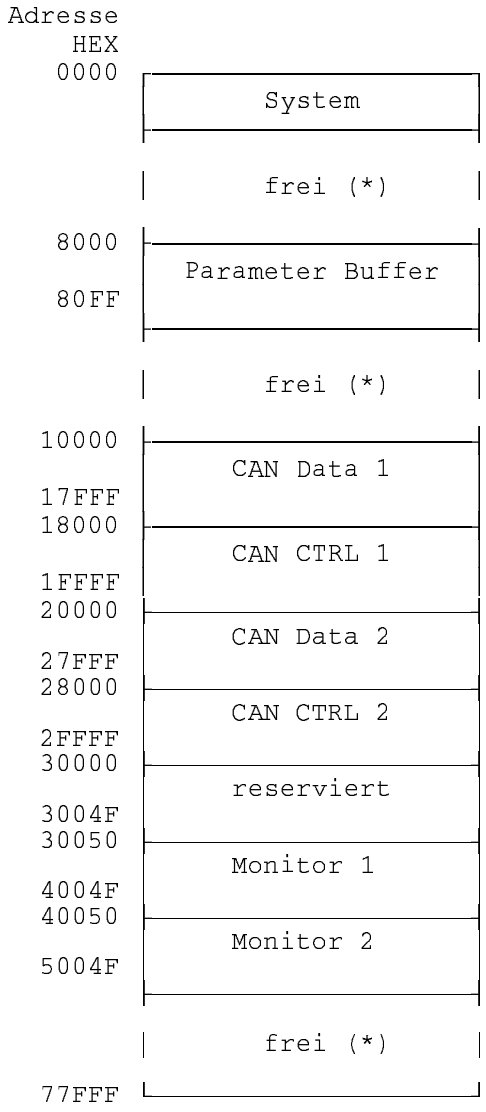
Die CAN2 ist eine intelligente Schnittstellenkarte für den VMEbus, die lokal 2 CAN-Kanäle (auch Netze oder Nets genannt) verwaltet.
Die Verwaltung der CAN-Kanäle wird von der im lokalen EPROM abgelegten Firmware übernommen. Sie steuert die CAN-Controller 82C200(VME-CAN2), bzw. 82527(VME-CAN2B) und verwaltet das lokale batteriegepufferte SRAM.

Die Firmware der CAN2 ist für die Bedienung der Karte über den VMEbus ausgelegt! Ein Stand-Alone-Betrieb und die Steuerung über die lokale CPU sind nicht vorgesehen.

Der Anwender wählt und steuert die gewünschten Betriebsarten der CAN-Identifier durch Setzen und Lesen von Zellen des linear organisierten Speichers. Die zu sendenden Daten werden in diesem Speicher abgelegt und empfangene Daten werden hier gelesen.

1.2 Speicher-Struktur

Der Speicher ist in verschiedene Bereiche unterteilt, über die jeweils Daten, Parameter und Kommandos übergeben werden.



(*) In den als 'frei' gekennzeichneten Speicherbereichen richtet die lokale Firmware bei Bedarf die Rx-Buffer oder die CMS-Buffer ein.

Die Schnittstelle zum CAN-Bus bilden zwei Strukturfelder, über die Daten empfangen oder gesendet werden können ('**CAN Data 1**', '**CAN Data 2**'). Die Daten eines CAN-Identifiers (CAN-Id.) werden zusammen mit spezifischen Parametern in sogenannten 'Strukturelementen' abgelegt.

Im '**Parameter-Buffer**' werden Parameter und Kommandos für den Datentransfer übergeben (Bitrate, etc.).

Jeder CAN Data-Struktur ist ein Kontroll-Feld ('**CAN CRTL 1**', '**CAN CTRL 2**') zugeordnet, das in erster Linie der Firmware zur Ablage aktueller Parameter dient. Der Anwender wählt in den Strukturelementen der CRTL-Felder den Betriebsmode des jeweiligen CAN-Identifiers.

Die '**Monitor-Buffer**' dienen zur sequentiellen Ablage von empfangenen Daten. Im Gegensatz zu den Strukturfeldern CAN Data 1 und CAN Data 2, in denen die Nachrichten (im Rx-Mode) immer wieder überschrieben werden, speichern die Monitor-Buffer die Nachrichten in der zeitlichen Reihenfolge ihres Eintreffens in einem Feld ab.

1.3 Mögliche Übertragungsmodes der Identifier

Für jeden Identifier ist einer der folgenden sechs Übertragungsmodes möglich:

1. Tx only
   In ein Strukturelement eingetragene Daten können nur gesendet werden.

2. Rx only
   In diesem Strukturelement können nur Daten empfangen werden.

3. Tx-Auto-Remote
   Die Daten dieses Strukturelementes werden beim Empfang eines 'Remote-Request'-Frames automatisch gesendet.

4. Monitor
   Empfangene Daten werden nicht in dem zugehörigen Strukturelement, sondern im Monitor-Buffer sequentiell mit 'Time-Stamp' abgelegt.

5. C-Net (Bridge Funktion)
   Bei dieser Übertragungsart wird eine bidirektionale Verknüpfung zwischen zwei beliebigen Strukturelementen geschaffen, d.h. es können z.B. zwei physikalisch getrennte CAN-Netze über die CAN2 verbunden werden.

6. CAN-Serial
   Eintreffende Daten werden nicht nur im Strukturelement, sondern auch im FIFO abgelegt, um dem VMEbus-Master die Möglichkeit zu geben, eintreffende Daten in ihrer zeitlichen Reihenfolge über einen Interrupt-Zyklus abzuarbeiten.

1.4 Einführung in die Arbeitsweise der Firmware

Im weiteren werden zum besseren Verständnis der Arbeitsweise der Firmware einfache Betriebsmodes des Rx- (CAN2 empfängt Nachrichten vom CAN-Bus) und des Tx-Transfers (CAN2 sendet Nachrichten auf den CAN-Bus) erläutert. Im Anschluß daran werden kurz die Möglichkeiten für Rückmeldungen von der CAN2 zum VMEbus-Master über das FIFO vorgestellt.

Die vollständige Erklärung der im folgenden verwendeten Begriffe ist in den anschließenden Beschreibungen der Speicher-Strukturen und -Buffer zu finden. In der Regel wird der Anwender nur mit den Daten- und Kontroll-Strukturen arbeiten. Die Monitor-Struktur und der Parameter-Buffer werden für weiterführende Funktionsabläufe benötigt.

Den Abschluß dieses Handbuchs bildet eine Übersicht über die möglichen Interrupt-Quellen und deren Bearbeitung.

1.4.1 Rx-Transfers

1.4.1.1 Ablaufbeschreibung

Hier wird der einfachste Mode eines Rx-Transfers beschrieben: Daten werden vom CAN-Bus empfangen und im lokalen Speicher abgelegt (ohne 'Time Out').

Initiator	Hardware	Hardware	Firmware	Firmware
Ursache	Senden von Tx-Frames anderer CAN-Teilnehmer	Controller empfängt Frame mit zugelassenem Identifier und löst Rx-IRQ aus	Rx-Interrupt	Eintreffen eines gültigen Rx-Strukturelementes
Wirkung	CAN-Controller empfängt alle gesendeten Frames und filtert Frames mit gewünschtem Identifier heraus	Übertragung der Information an Rx-Software-Filter	Selektieren der empfangenen Strukturen nach Rx-Software-Filter	Aktualisieren des Strukturelementes im CAN_Data-Strukturfeld

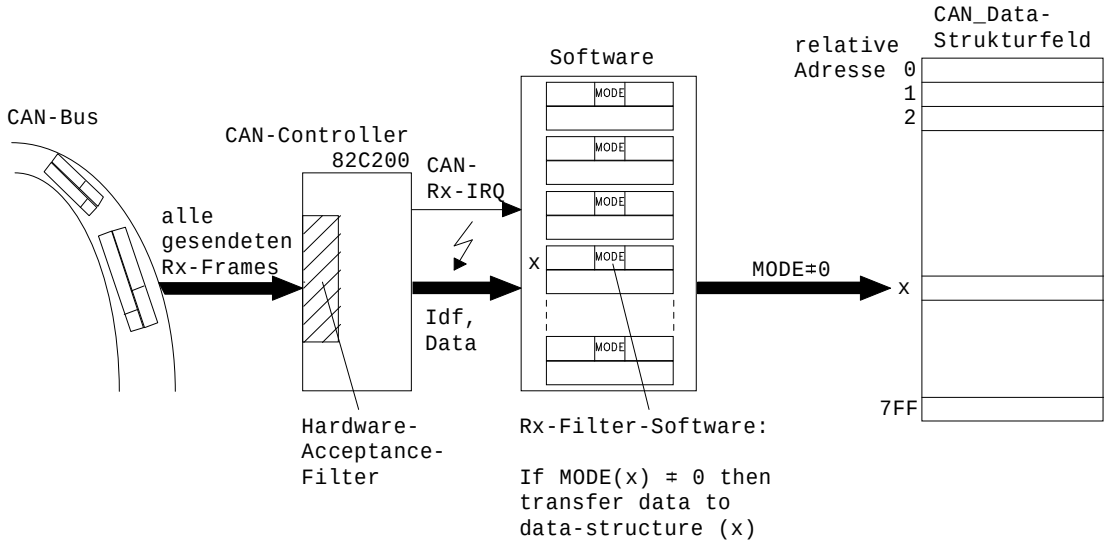


Abb. 1.4.1: Ablauf eines Rx-Transfers

Beschreibung der Abb. 1.4.1:

Der CAN-Controller 82C200 empfängt generell alle auf dem CAN-Bus gesendeten Nachrichten. Wenn er gültige Daten empfangen hat, löst er einen Interrupt aus und überträgt die Daten an die lokale Firmware.

Der Controller besitzt einen Hardware-Filter (Acceptance-Filter), der nur ausgewählte Identifier passieren läßt. Da die Durchlaßbedingungen dieses Filters es nicht gestatten, beliebige Identifier-Kombinationen zu selektieren, wird hier nur eine Vorauswahl der Identifier getroffen und ein Software-Filter hintergeschaltet.

Das Software-Filter der Firmware selektiert die ankommenden Nachrichten nach der Bedingung '...lasse alle ankommenden Identifier, für deren zugehöriges Strukturelement im lokalen Speicher ein Rx-Mode gewählt wurde, passieren' (MODE ≠ 0).

Die Nachrichten, die alle gestellten Bedingungen erfüllen, werden im lokalen SRAM ('CAN Data...') unter der Adresse, die (relativ gesehen) der gesendeten CAN-Identifier-Nr. entspricht, abgelegt. Hierbei wird nur die Anzahl Daten-Bytes des entsprechenden Identifiers überschrieben, die in der Nachricht angegeben ist ('LENGTH').

In Abb. 1.4.1 nicht dargestellte Zusammenhänge:

Der aktuelle Zustand der Übertragung kann im Daten-Strukturfeld ('CAN Data...') vom Anwender gelesen werden ('STATUS').

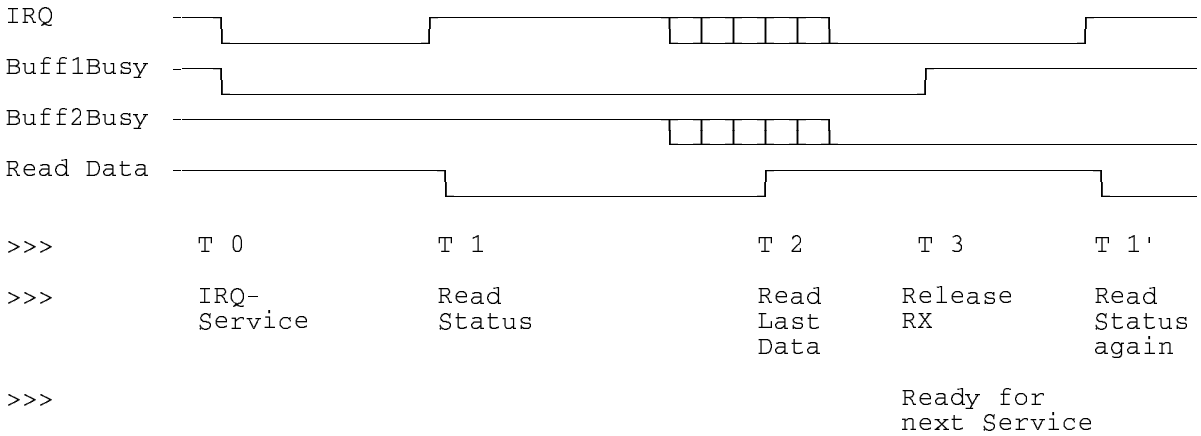
Die in einem Strukturelement abgelegten Daten werden immer überschrieben (aktualisiert), wenn auf diesem Identifier neue Daten eintreffen. Um den Anwender darüber zu informieren, daß auf diesem Identifier Nachrichten empfangen wurden, kann nach Abschluß des Rx-Transfers ein Interrupt ausgelöst werden ('EVTRIG'), dem der Anwender eine selbst definierte Interrupt-Routine zuordnen muß.

Beim Auftreten eines Übertragungsfehlers ('STATUS') kann ebenfalls ein Interrupt ausgelöst werden.

Die in Klammern (' ') angegebenen Hinweise beziehen sich auf Speicher-Strukturen oder -zellen, die in späteren Kapiteln beschrieben werden.

1.4.1.2 Dauer eines CAN-Rx-Transfers/Zeitdiagramm

Das folgende Zeitdiagramm zeigt den Ablauf eines Rx-Transfers beginnend mit dem Rx-Interrupt (IRQ) des CAN-Controllers 82C200/82527. Aus den dargestellten Parametern ergibt sich die max. benötigte Zeit, die die VME-CAN2 benötigt, um die eintreffenden Daten in ihrem lokalen Speicher (Strukturelement) abzulegen.



Ablaufbeschreibung:

Zeitpunkt T 0:
Die eintreffenden Daten haben den 'Acceptance-Filter' des CAN-Controllers passiert - der Controller empfängt die Daten. Der Controller 82C200/82527 besitzt zwei Daten-Buffer, in denen er eintreffende Daten zwischenspeichern kann. Er legt die Daten in einem seiner Buffer ab und löst seinen Rx-Interrupt aus, der eine lokale Interrupt-Routine startet. Der alternative Buffer steht dann noch zum verlustfreien Empfang der nächsten Message zur Verfügung.

Zeitpunkt T 1:
Die Interrupt-Routine liest den Status des Controllers, stellt fest, daß Daten eingetroffen sind und beginnt mit dem Auslesen des Buffers. Die Zeit, die für das Auslesen des Buffers nötig ist, hängt im wesentlichen von der Anzahl der eintreffenden Daten-Bytes ab (LENGTH).

Zeitpunkt T 2:
Das letzte Byte des (ersten) Buffers wird gelesen.

Zeitpunkt T 3:
Der Lesezyklus ist beendet, Buffer 1 wird wieder freigegeben. Zu diesem Zeitpunkt kann bereits wieder ein Interrupt aktiv sein, wenn Buffer 2 frei gewesen ist, und Daten eingetroffen sind.

Zeitpunkt T 1':
Der Status des Controllers wird von der Firmware wieder abgefragt und die Daten werden aus Buffer 2 ausgelesen.

Aus diesen Zusammenhängen ergeben sich die folgenden Zeiten für Lesezyklen:

Full READ-IRQ-Service time: T 0/3 (Worst Case)
(Die Abarbeitung der eintreffenden Rx-Frames erfolgt nacheinander, weil der jeweils nächste Frame erst nach vollständiger Abarbeitung des vorhergehenden eintrifft.)

Follow-Up-READ-IRQ-Service time: T 1/3' (Overlapping IRQs)
(Die Abarbeitung der eintreffenden Rx-Frames erfolgt 'parallel', weil der jeweils nächste Frame während der Abarbeitung des vorhergehenden eintrifft.)

DataLength	T 1 [us]	T 2 [us]	T 3 [us]	T 1' [us]	T 0/3 [us]	T 1/3' [us]
8	16	18	2	10	36,00	30,00
7	16	17,5	3,5	10	37,00	31,00
6	16	17,5	2,5	10	36,00	30,00
5	16	17	2	10	35,00	29,00
4	16	16	2	10	34,00	28,00
3	16	16	2,5	10	34,50	28,50
2	16	15	2	10	33,00	27,00
1	16	14,5	2	10	32,50	26,50
0	16	-	16	10	32,00	26,00
'*disabled*'	16	-	-	8,5	16,00	8,50

'*disabled*'....Dieser CAN-Id. wurde vom Hardware-Acceptance-Filter akzeptiert, nicht aber vom Software-Filter der Firmware.

Tabelle 1.4.1: Zeitbedarf der Abarbeitung von Rx-Frames in Abhängigkeit von der Datenlänge

T 0/3 ist die gesamte Abarbeitungszeit für Interrupts auf IDLE- oder USER-Task-Ebene (oder jeder anderen beliebigen Interrupt-Routine mit einem Level, der kleiner als IRQ6 ist), bis zum Lesen des letzten Daten-Bytes und der Freigabe des Rx-Buffers für den nächsten Transfer.

T1/3' ist die Abarbeitungszeit für einen gesamten Rx-Lesezyklus bei ineinander übergehenden (überlappenden) Interrupts eines oder beider CAN-Controller.

1.4.2 Tx-Transfers

In der folgenden Abbildung wird das Senden von Daten auf den CAN-Bus in zwei Schritten dargestellt:

Im ersten Schritt schreibt der Anwender vom VMEbus aus die zu übertragenden Daten ('Data1', 'Data2', usw.) in das Strukturelement des lokalen Speichers (CAN Data...), das dem gewünschten Identifier zugeordnet ist.

Initiator	Anwender	Anwender	Hardware	Firmware	Hardware
Ursache	Datentransfer in den CAN_Data-Buffer	Setzen des Parameters 'LENGTH'	FIFO not empty	FIFO-Interrupt	CAN-Tx-free-IRQ
Wirkung	Aktualisierung der Daten im Buffer	Ablage von 'LENGTH' und Transfer des CAN-Identifier in FIFO	FIFO-Interrupt	Ordnen der Struktur-elemente nach Priorität, Verketten der Strukturel. durch 'NEXT_POINTER'	Senden auf CAN-Bus nach Priorität zu anderen Frames auf Bus

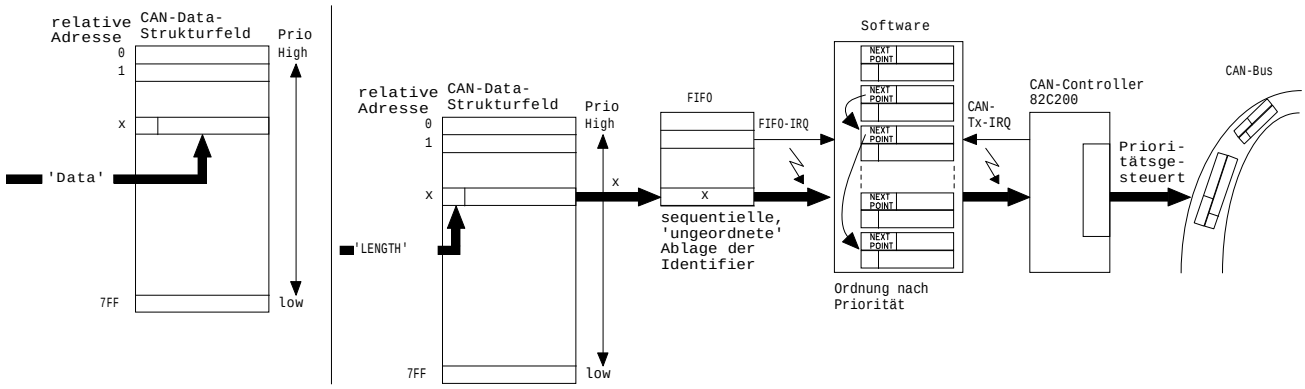


Abb. 1.4.2: Ablauf eines Tx-Transfers

Im zweiten Schritt wird die Übertragung z.B. durch den Eintrag der Anzahl der zu übertragenden Bytes ('LENGTH') mit negativem Vorzeichen gestartet.

Das FIFO bildet die Pufferzone zwischen dem VMEbus und dem CAN-Bus: In ihm werden die Pointer auf das entsprechende Strukturelement nach dem Triggern des Datentransfers sequentiell abgelegt. Das FIFO löst einen Interrupt aus, sobald es nicht mehr leer ist. Dieser veranlaßt die lokale Software, das Daten-Strukturelement zusammen mit dem CTRL-Strukturelement (gleichen Identifiers) in eine Prioritätskette aufzunehmen und die Strukturelemente untereinander durch Pointer zu verbinden.

Jedem Strukturelement ist durch seinen Identifier (hier: 'x') und somit seiner relativen Adresse eine Sendepriorität zugeordnet. Der Identifier mit der niedrigsten Adresse besitzt die größte Priorität, der mit der höchsten Adresse die kleinste Priorität.

Während die Firmware die zu sendenden Strukturelemente ordnet, wartet sie auf den Tx-Interrupt des Controllers 82C200/82572. Dieser Interrupt zeigt an, daß der Controller seinen letzten Sendeauftrag beendet hat, und jetzt bereit ist, einen neuen Auftrag abzuarbeiten.

Nach dem Interrupt wird die Nachricht mit der höchsten lokalen Sendepriorität in den Controller geladen. Der Controller versucht jetzt, die Nachricht zu senden. Die Vergabe der Sendepriorität auf dem CAN-Bus erfolgt, wie die lokale Prioritätsvergabe, nach dem Identifier.

In Abb. 1.4.2 nicht dargestellte Zusammenhänge:

Der aktuelle Zustand der Übertragung kann im Daten-Strukturfeld ('CAN Data...') vom Anwender gelesen werden ('STATUS').

Die erfolgreiche Beendigung des Tx-Transfers oder aufgetretene Übertragungsfehler werden im Daten-Strukturelement des Identifiers ('STATUS') angezeigt und können einen Eintrag in das FIFO 'Data To VMEbus' nach sich ziehen ('EVTRIG'). Das FIFO kann nach Prüfung des FIFO-Status gelesen werden. Ist ein VMEbus-Interrupt freigegeben, so löst der Eintrag in das FIFO den Interrupt aus. Diesem anwender-definierten Interrupt muß eine Interrupt-Routine zur Abarbeitung des Interrupts folgen.

Sollte der Anwender vom VMEbus aus so viele Sendeaufträge starten, daß das FIFO vollständig gefüllt ist, wird ein Interrupt auf dem VMEbus erzeugt. Dieser Interrupt teilt dem Anwender mit, daß er den Datentransfer drosseln muß, um Daten-verluste zu verhindern ('iovec').

Die in Klammern (' ') angegebenen Hinweise beziehen sich auf Speicher-Strukturen oder -zellen, die in späteren Kapiteln beschrieben werden.

1.4.3 Statusmeldungen der CAN2 und Statusabfrage durch den VMEbus-Master

Wie bereits erwähnt, ist es bei verschiedenen Übertragungszuständen notwendig, dem VMEbus-Master Nachrichten von der CAN2 zukommen zu lassen. Dies können z.B. Statusmeldungen zum Verlauf des Transfers sein.

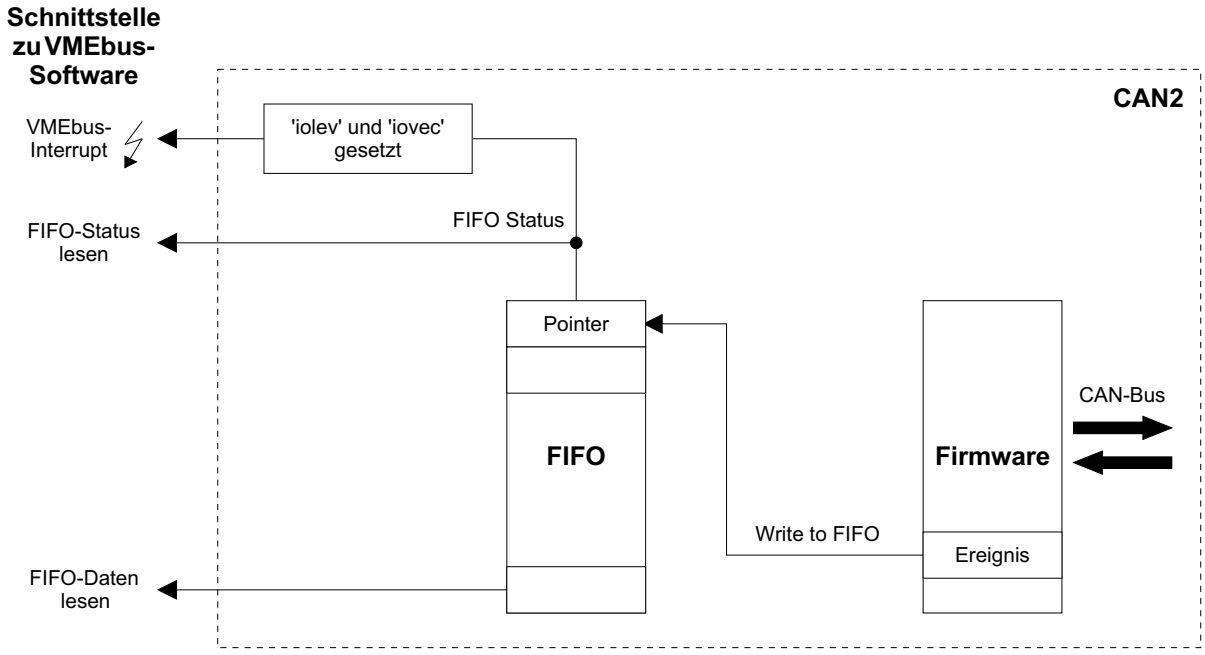


Abb. 1.4.3: Funktion des FIFOs 'Data to VMEbus'

Tritt ein Ereignis auf dem CAN-Bus auf, wie z.B. das Ende eines Transfers, so kann die lokale Firmware bei entsprechender Parametrierung einen Pointer in ein FIFO schreiben. Dieser Pointer enthält den betroffenen Identifier und die Nummer des betroffenen Netzes.

Der VMEbus-Master kann den Status dieses FIFOs durch Polling überwachen und die eingetroffenen Pointer auslesen.
Außerdem ist es möglich, von der CAN2 einen VMEbus-Interrupt auslösen zu lassen, wenn das FIFO nicht mehr leer ist.

Das Polling und das Interrupt-Handling der CAN2 werden in dem Kapitel 'Zusammenfassung der Interrupt-Optionen' ausführlich beschrieben.

2. Modes der Datenübertragung

Dieses Kapitel beschreibt die Daten- und Kontrollfelder der beiden CAN-Kanäle. Außerdem werden die grundlegenden Modes der Datenübertragung mit Anleitungen zur Vorgehensweise beschrieben.

2.1 Die Strukturfelder CAN Data 1 und CAN Data 2

2.1.1 Funktion und Aufbau

Über die Strukturfelder CAN Data 1 (Net 1) und CAN Data 2 (Net 2) werden die Daten des VMEbus auf den CAN-Bus übertragen.

Ein Strukturfeld ist in 2048 Elemente unterteilt.

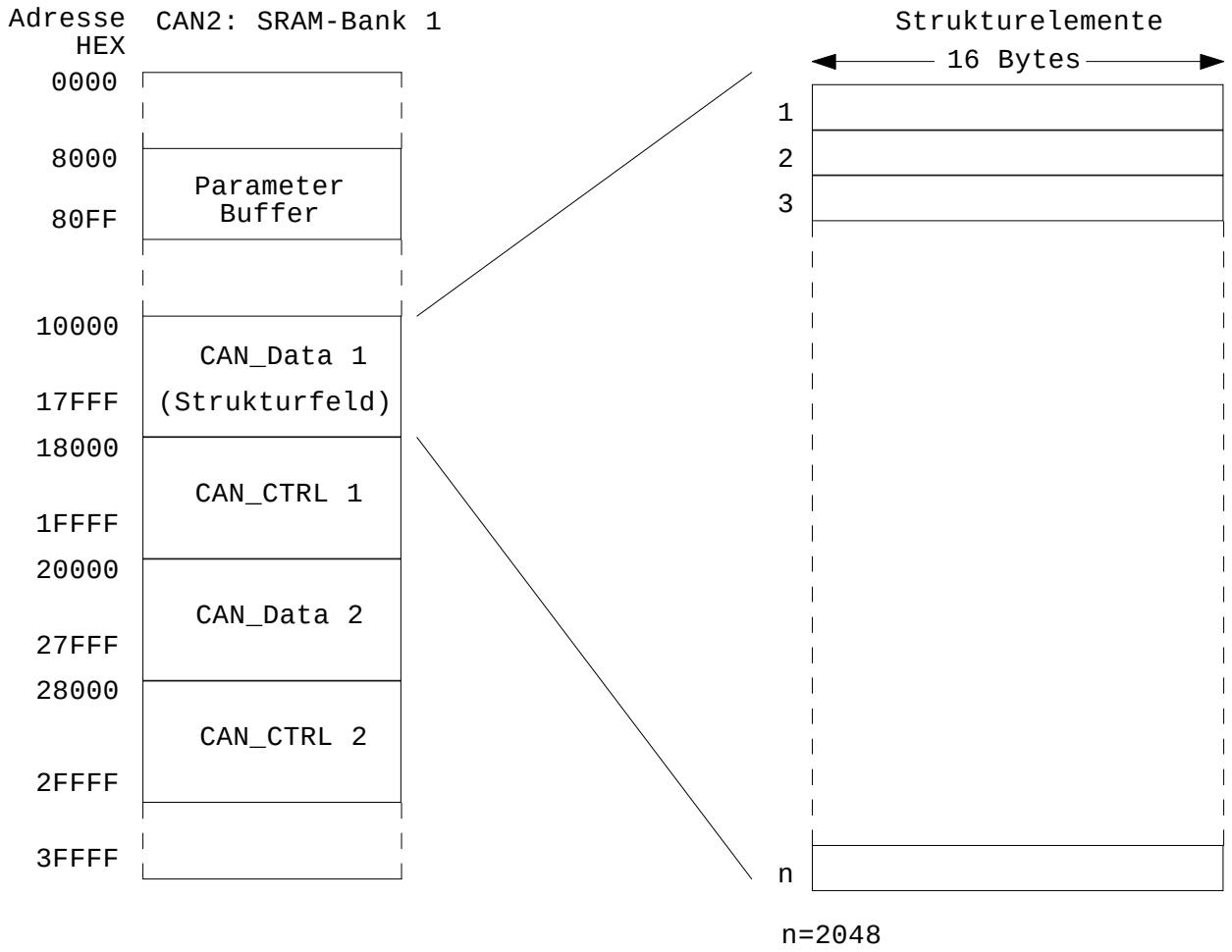


Abb. 2.1.1: Aufteilung eines CAN Data Strukturfeldes (Beispiel: CAN Data 1)

Jedem möglichen CAN-Identifier eines CAN-Kanals ist ein Strukturelement zugeordnet (max. 2048 Ids sind möglich).

Ein Strukturelement ist 16 Byte lang und enthält neben den übertragenen Daten (max. 8 Byte) 8 Byte Kontroll- und Steuerparameter.

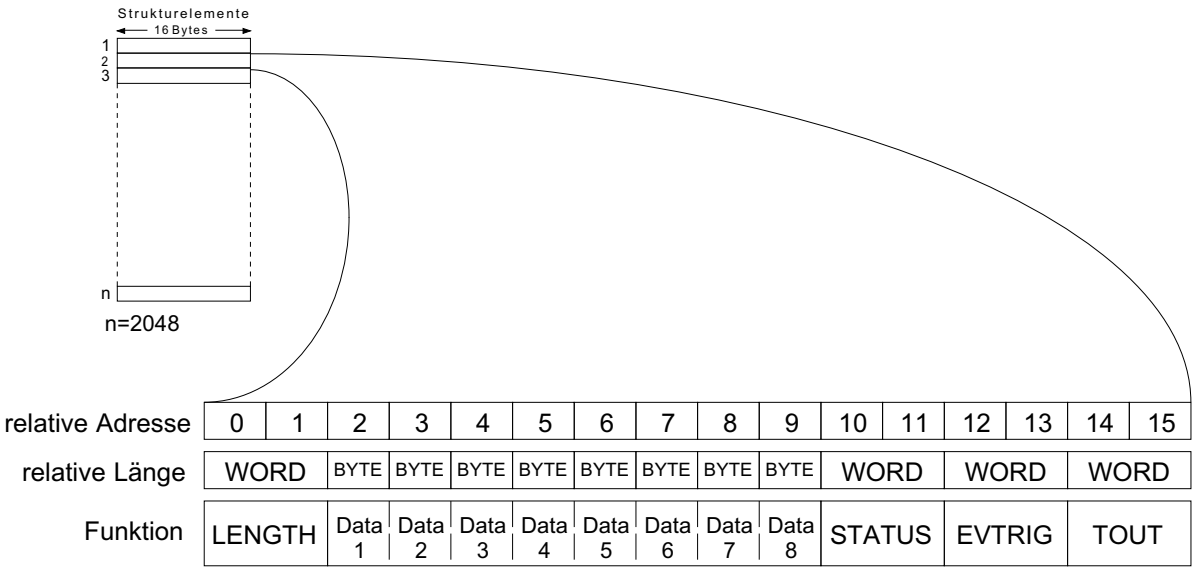


Abb. 2.1.2: Inhalt eines CAN Data Strukturelementes

2.1.2 Die Bytes eines Daten-Strukturelementes

Im folgenden werden die Bytes eines Daten-Strukturelementes in aufsteigender Reihenfolge beschrieben.

LENGTH.....gibt die Anzahl der gültigen Daten-Bytes an und bestimmt in Zusammenhang mit der Zelle 'MODE' in dem zugehörigen Kontroll-Strukturelement (siehe Kap. 'Parameter- und Kommandoübergabe über den Parameter-Buffer') den für diesen Identifier festgelegten Betriebs-Mode.

Mögliche Werte [HEX]:
0000...0008 (nur Anzahl der gültigen Bytes)
0020...0028 (LENGTH im Mode RTR-Senden)
0040...0048 (LENGTH im Mode Rx-Transfer mit Time-Out)
0060...0068 (Anzahl der gültigen Bytes und Start von Tx-Transfers)
FFFF...FFF7 (entspricht negativem Wert der Anzahl der gültigen Bytes und führt zum Start von Tx-Transfers)

Data1..Data8....enthalten die übertragenen Daten.
Es können null bis acht Byte übertragen werden.

STATUS.....gibt Informationen über den aktuellen Zustand der Übertragung.

Folgende Meldungen sind bisher implementiert:

Wert 'STATUS' (HEX)	Zustand der Übertragung
0000	Übertragung erfolgreich abgewickelt
0001	Rx-Time-Out (Fehler)
0002	Tx-Time-Out (Fehler)
0004	Tx-Error (Fehler)
0005	Controller 'Off Bus' (Fehler)
0006	Controller 'Busy' (Fehler)
0008	Zugriff auf nicht vorhandenes Netz (Fehler)
0010	CMS: Domain Time Out
0011	CMS: Falsches Acknowledge bei CMS-Protokoll
0012	CMS: Abort Domain Transfer
0020	CMS: Compare-Fehler des CMS-Watchdogs
0101	RTR-Frame wurde empfangen
FFFE	Warten auf Rx bei 'Remote Request'
FFFF	Übertragung noch nicht beendet

Tabelle 2.1.1: Funktion des STATUS-Wortes

Beschreibung der Statusmeldungen:

Rx-Time-Out..... (Fehler)
Innerhalb der in der Zelle 'TOUT' eingetragenen Zeit wurden keine Daten in diesem Strukturelement empfangen.

Die Aktivierung der Time-Out-Überwachung erfolgt nach dem Eintrag des entsprechenden Betriebs-Modes in das CTRL-Strukturelement des Identifiers ('MODE' über Parameter-Buffer eintragen) durch Triggern der Zelle 'LENGTH' des CTRL-Strukturelementes. Nach dem Triggern wird der in der Zelle 'TOUT' eingetragene Wert in den 'TIME COUNTER' des CTRL-Elementes eingetragen, das Element in die 'Rx-Time-Out-Kette' integriert und der 'TIME COUNTER' heruntergezählt.

Nach Überschreiten der Time-Out-Zeit wird dieser Fehlerstatus immer gesetzt - das FIFO 'Data To VME' kann mit einem Pointer auf das Strukturelement geladen und ein Interrupt kann auslöst werden (siehe Kap. 'Interrupts').

Tx-Time-Out..... (Fehler)
Das Strukturelement konnte nicht innerhalb der in der Zelle 'TOUT' eingetragenen Zeit gesendet werden. Der Übertragungsversuch wird abgebrochen.

Die Aktivierung der Time-Out-Überwachung erfolgt nach dem Eintrag des entsprechenden Betriebs-Modes in das CTRL-Strukturelement des Identifiers ('MODE' über Parameter-Buffer eintragen) durch Triggern der Zelle 'LENGTH' des CTRL-Strukturelementes. Nach dem Triggern wird der in der Zelle 'TOUT' eingetragene Wert in den 'TIME COUNTER' des CTRL-Elementes eingetragen, das Element in die 'Tx-Time-Out-Kette' integriert und der 'TIME COUNTER' heruntergezählt.

Nach Überschreiten der Time-Out-Zeit wird dieser Fehlerstatus immer gesetzt - das FIFO 'Data To VME' kann mit einem Pointer auf das Strukturelement geladen und ein Interrupt kann auslöst werden (siehe Kap. 'Interrupts').

Tx-Error..... (Fehler)
Dieser Fehler tritt auf, wenn der CAN-Controller 82C200 über seinen Tx-Interrupt von der Firmware neue Daten zur Übertragung anfordert, obwohl er sein 'Tx-complete'-Signal noch nicht gesetzt hatte. Die Fehlermeldung kann zu einem Eintrag in das FIFO 'Data TO VME' führen, der wiederum einen VMEbus-Interrupt auslösen kann.

Controller 'Off Bus'... (Fehler)
Treten bei Übertragungsversuchen des CAN-Controllers 82C200 grobe Fehler auf (z.B. Bitrate Controller ≠ Bitrate Bus), 'verläßt' der Controller den Bus, da weitere Übertragungsversuche zwecklos wären.

Tritt dieser Fehlerzustand während laufender Tx-Übertragungen auf, so wird diese Fehlermeldung nicht nur in das aktuelle Strukturelement eingetragen, sondern auch in alle anderen bereits in die lokalen Prioritätskette aufgenommenen Tx-Strukturen. Tritt der Fehler bei der ersten Übertragung eines Tx-Elementes auf, so wird die Meldung nur in dieses Strukturelement eingetragen, diese Übertragung abgebrochen und die Übertragung des nächsten Tx-Strukturelementes gestartet. Die Fehlermeldung kann zu einem Eintrag in das FIFO 'Data TO VME' führen, der wiederum einen VMEbus-Interrupt auslösen kann.

Controller 'Busy'..... (Fehler)
Diese Fehlerbedingung tritt ein, wenn die lokale Tx-Prioritätskette leer ist, über den VMEbus eine Tx-Anweisung gesendet wird und die Firmware bei der üblichen Abfrage des Controllers nach seinem Tx-Status die Meldung 'Busy' erhält, obwohl der Controller zu diesem Zeitpunkt nicht busy sein dürfte (-> fatal error!).

Die Fehlermeldung führt zu einem Eintrag in das FIFO 'Data TO VME' führen, wenn die Zelle 'EVTRIG' gesetzt ist. Der Eintrag wiederum kann einen VMEbus-Interrupt auslösen kann.

Zugriff auf nicht..... vorhandenes Netz (Fehler)
Es wurde ein nicht vorhandenes Netz angesprochen. Dieser Fehler tritt ein, wenn z.B. auf Net 7 zugegriffen wird, obwohl nur Net 2 und Net 3 vorhanden sind. Diese Fehlermeldung ist für den CAN-Master Treiber auf dem VMEbus von Bedeutung.
Die Fehlermeldung führt zu einem Eintrag in das FIFO 'Data TO VME' führen, wenn die Zelle 'EVTRIG' gesetzt ist. Der Eintrag wiederum kann einen VMEbus-Interrupt auslösen kann.

Die folgenden CMS-Fehlermeldungen führen immer zu einem Eintrag in das FIFO 'Data TO VME'. Der Eintrag kann einen VMEbus-Interrupt auslösen kann:

Domain-Time-Out...
Beim CMS-Domain-Transfer ist ein Time-Out-Fehler aufgetreten.

Falsches Acknowledge bei CMS-Protokoll...
Während des CMS-Transfers hat die CAN2 ein falsch kodiertes Acknowledge-Frame empfangen (z.B. falsches command specifier).

Abort Domain Transfer...
Der Transfer wurde abgebrochen, da der CMS-Partner einen Abort Domain Tansfer gesendet hat.

Compare-Fehler...
Der CMS-Watchdog hat einen Fehler beim Vergleich des Soll-Wertes mit dem Ist-Wert des Toggle-Bits oder des States festgestellt.

RTR-Frame wurde..... empfangen
Für dieses Strukturelement wurde ein RTR-Frame empfangen, das zum Senden der in diesem Element enthaltenen Daten auffordert. Die Status-Meldung bleibt nur solange aktiv, wie keine Sendung erfolgt ist.

Warten auf Rx bei.... 'Remote-Request'
Auf diesem Identifier wurde ein 'Remote-Request' gesendet. Die Firmware wartet jetzt auf das Eintreffen einer Nachricht für dieses Strukturelement. Der Status wird zurückgesetzt, wenn eine Nachricht eingetroffen ist.

Übertragung noch... nicht beendet
Der Transfer dieses Strukturelementes ist noch nicht abgearbeitet. Nach Beendigung des Transfers wird der Status zurückgesetzt und das Strukturelement aus der lokalen Prioritätskette entfernt.

EVTRIG.....
legt fest, ob ein Eintrag in das FIFO 'Data TO VME' erfolgen soll. Dieser Eintrag kann einen User-VME-Interrupt auslösen. EVTRIG ist nur eine Freigabebedingung für das Auslösen eines Interrupts. Zusätzlich müssen die Zellen 'iolev' und 'iovec' des Kommandos 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sein.

Sind diese Freigabebedingungen erfüllt, wird beim Auftreten der unten beschriebenen Interrupt-Bedingungen ein Interrupt auf dem VMEbus mit dem Level von 'iolev' und dem über 'iovec' festgelegten Vector ausgelöst.

Wert 'EVTRIG' (HEX)	Funktion
0000	kein FIFO-Eintrag nach Eintreten einer Endebedingung
0001 : 00FF	FIFO-Eintrag und evtl. Auslösen eines User-IRQs nach Eintreten einer Endebedingung

Tabelle 2.1.2: Funktion der EVTRIG-Bytes

Gültige Endebedingungen sind erfolgreiche oder durch Auftreten eines Fehlers beendete Übertragungen:

1. Übertragung erfolgreich abgewickelt
2. Rx-Time-Out (Fehler)
3. Tx-Time-Out (")
4. Tx-Error (")
5. Controller 'Off Bus' (")
6. Controller 'Busy' (Fehler)

Fehler bei CMS-Transfers führen immer zum Eintrag eines Pointers in das FIFO, unabhängig vom Wert der Zelle EVTRIG!

TOUT..... gibt bei Werten größer null ($0001...$7FFF) den Time-Out-Wert in msec an (Betrieb ohne CMS-Protokoll) oder die Watchdog Guard Time (Betrieb mit CMS-Protokoll).

Betrieb ohne CMS-Protokoll
Die Aktivierung der Time-Out-Überwachung erfolgt nach dem Eintrag des entsprechenden Betriebs-Modes in das CTRL-Strukturelement des Identifiers ('MODE' über Parameter-Buffer eintragen) durch Triggern der Zelle 'LENGTH' des CTRL-Strukturelementes. Nach dem Triggern wird der in der Zelle 'TOUT' eingetragene Wert in den 'TIME COUNTER' des CTRL-Elementes eingetragen, das Element in eine Time-Out-Kette integriert und der 'TIME COUNTER' heruntergezählt.
Die Time-Out-Option kann sowohl für Tx- als auch für Rx-Transfers genutzt werden:
Im Tx-Mode löst er einen Tx-Time-Out-Error aus, wenn die Übertragung nicht innerhalb der vorgegebenen Zeit erfolgen konnte. Der Tx-Time-Out wird in der Zelle 'STATUS' angezeigt und kann einen Interrupt auslösen. Desweiteren ist es in Verbindung mit dem Parameter-Buffer möglich, in periodischen Zeitabständen Tx-Transfers von bis zu 64 verschiedenen Strukturelementen zu starten.
Im Rx-Mode wird ein Rx-Time-Out in der Zelle 'STATUS' angezeigt, wenn innerhalb der Time-Out-Zeit keine Nachrichten auf diesem Identifier empfangen wurden. Der 'TIME COUNTER' wird mit jedem Empfang einer Nachricht wieder zurückgesetzt.

Wird das Kommando 'Periodical Tx-Transfer' mit den entsprechenden Parametern 'Net' und 'Idf' im Parameter-Buffer gesetzt, dann werden die negativen Werte von TOUT genutzt, um Tx-Transfers in regelmäßigen Abständen zu starten. Zum Start der Übertragungen muß nach dem Setzen der o. a. Zellen außerdem einmalig ein Tx-Transfer erfolgreich gestartet werden.
Die periodischen Tx-Transfers werden beendet, indem das Kommando im Parameter-Buffer zurückgesetzt, oder die Time-Out-Zeit auf positive Werte gesetzt wird.

> Sind die o. a. Zellen des Parameter-Buffers nicht gesetzt, wird mit negativen TOUT-Werten der Time-Out 'disabled'.
> Dies sollte unbedingt vermieden werden, da es vorkommen kann, daß sich das System 'aufhängt', falls eine Abarbeitung des Tx-Auftrags nicht möglich ist!

> In allen Betriebsarten sollte 'TOUT' niemals auf '0' und niemals auf Werte kleiner als 5 msec gesetzt werden, weil sich die Auflösung des Zählers in diesem Bereich befindet!

'TOUT' (HEX)	Funktion
0000 bis 7FFF	abhängig von Implementierung: ohne CMS: Time-Out in msec 0005 => 5 msec (*) 7FFF => 32,767 sec mit CMS: Watchdog Guard Time 0005 => 5 msec 7FFE =>32,767 sec
8000 bis FFFF	Wiederholungszeit für 'Periodical Tx-Transfers' FFFB => 5 msec 8000 =>32,768 sec

Tabelle 2.1.3: Funktion der 'TOUT-Bytes'

Betrieb mit CMS-Protokoll
Ist die CAN2 mit der CMS-Implementierung ausgestattet, so wird mit positiven Werten von TOUT die Watchdog Guard Time des CMS-Watchdogs übergeben. Der CMS-Watchdog wird in einem eigenen Kapitel noch näher beschrieben.

2.2 Die Kontroll-Strukturen CAN CTRL 1 und CAN CTRL 2

2.2.1 Funktion und Aufbau

Die Kontroll-Strukturen dienen der Firmware zur Zuordnung von Übertragungsparametern, Ablage von Statusinformationen und Variablen und Dokumentation der Reihenfolge der aktiven Daten-Strukturelemente.

Jedem Strukturelement der beiden CAN Data-Strukturen ist ein CAN-CTRL-Strukturelement zugeordnet. Die Strukturelemente der Kontrollstrukturen sind ebenso wie die der Datenstrukturen auf relativen Adressen der RAM-Bank1 abgelegt, die identisch zu den zugehörigen CAN-Identifier-Nummern sind. Es sind also für beide Kanäle je 2048 Kontroll-Strukturelemente vorhanden. Ein Kontroll-Strukturelement ist 16 Byte lang.

> Das direkte Setzen der Parameter der Kontroll-Strukturen ist weitgehend der Firmware vorbehalten. **Der Anwender darf lediglich die Zelle 'XTTID' direkt setzen!**

Die Strukturelemente sind wie folgt aufgebaut:

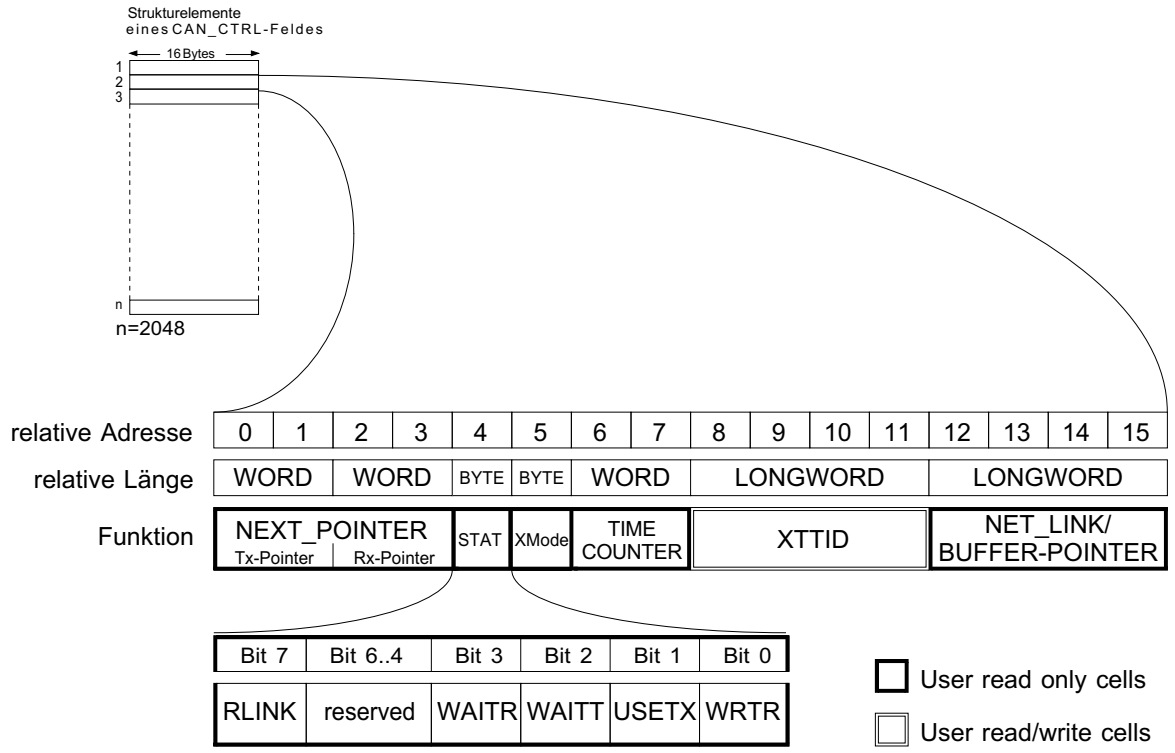


Abb. 2.2.1: Inhalt eines CAN CTRL-Strukturelementes

2.2.2 Die Bytes eines CTRL-Strukturelementes

Im folgenden werden die Bytes eines CTRL-Strukturelementes in aufsteigender Reihenfolge beschrieben. Bis auf 'XTTID' dürfen die Zellen vom Anwender nur gelesen werden.

NEXT POINTER..... Dieser Pointer enthält die absoluten Adressen von CAN CTRL-Strukturelementen.
In Byte 1 und 2 ist die Adresse des nächsten Tx-Strukturelementes (Tx-Pointer) enthalten. Byte 3 und 4 enthalten die Adresse des nächsten Rx-Strukturelementes (Rx-Pointer):

Tx-Pointer
Der Pointer enthält die Adresse des nächsten aktiven CRTL-Strukturelementes mit geringerer Priorität (= größerer Identifier == größere Adresse).
Diese Verkettung dient der lokalen Ordnung der Strukturelemente nach ihrer Sendepriorität. Hat der CAN-Controller-Baustein 82C200 seinen letzten Sendeauftrag abgearbeitet, fordert er das nächste CRTL-Strukturelement mit der höchsten Priorität an, verknüpft es mit dem dazugehörigen Data-Strukturelement zu einem CAN-Bus-Frame und sendet die Nachricht auf den CAN-Bus, sobald diesem Frame auf dem Bus die höchste Priorität zugestanden wird.
Das letzte Strukturelement der Kette hat den Eintrag '$FFFF' im Tx-Pointer.

Rx-Pointer
Der Rx-Pointer enthält die Adresse des nächsten aktiven Rx-Strukturelementes, für das der 'Time-Out'-Mode gewählt wurde.

Diese Verkettung dient dem kontinuierlichen Herunterzählen der 'Time-Counter' (siehe unten) der aktiven Rx-Elemente in der Reihenfolge in der Aktivierung des Time-Out-Modes. Die Firmware zählt die Counter der Elemente jeweils um eine Stufe weiter.
Das letzte Strukturelement in dieser Time-Out-Kette hat den Eintrag '$FFFF' im Rx-Pointer.

STAT..... Die Zelle 'STAT' teilt sich in 8 Bit mit individuellen Funktionen auf. Alle Bits dieser Zelle dürfen vom Anwender nur gelesen werden:

RLINK (Bit 7)
Rx-Struktur ist in Rx-Time-Out-Link eingebunden.

Bit 6...4
Diese Bits sind für zukünftige Anwendungen reserviert.

WAITR (Bit 3, active '1')
Dieses Bit zeigt an, daß sich das Strukturelement in dem Status 'waiting for Rx-End' befindet. Das Bits ist aktiv, solange sich das Strukturelement im Rx-Request-Mode befindet und keine gültige Endebedingung eingetreten ist. Endebedingungen sind das Eintreffen der Daten oder ein Rx-Time-Out.

WAITT (Bit 2, active '1')
Dieses Bit zeigt an, daß sich das Strukturelement in dem Status 'waiting for Tx-End' befindet. Das Bits ist aktiv, solange der Tx-Transfer noch nicht abgeschlossen ist, und wird nach Beendigung dieses Auftrags zurückgesetzt.

USETX (Bit 1, active '1')
Dieses Bit wird nur von der lokalen Firmware genutzt.

WRTR (Bit 0, active '1')
Dieses Bit zeigt an, daß sich das Strukturelement in dem Status 'wait for answer of RTR' befindet. Auf diesem Identifier wurde ein 'Remote-Request' gesendet, und die Firmware wartet auf den Empfang von Nachrichten. Nach Eintreffen einer Nachricht in dieses Strukturfeld wird das Bit zurückgesetzt.

XMode..... In der Zelle 'XMode' wird der Übertragungsmodus von der Firmware eingetragen, nachdem er vom Anwender über den Parameter-Buffer gesetzt wurde (siehe auch Kapitel 'Parameter- und Kommandoübergabe über den Parameter-Buffer').

TIME COUNTER..... In diesen beiden Bytes wird der aktuelle Wert des Time-Out-Zählers abgelegt.

Wird für ein Strukturelement in der Zelle 'MODE' der Time-Out-Mode gewählt, im Daten-Strukturelement ein positiver Wert in die Zelle 'TOUT' eingetragen und der gewählte Betriebsmode (Tx-Mode, Rx-Mode, usw.) aktiviert, so wird der Inhalt der Zelle 'TOUT' in den TIME COUNTER übernommen und dieses Strukturelement in eine Time-Out-Kette integriert. Hier werden die TIME COUNTER der Elemente in der Reihenfolge ihres zeitlichen Eintrags in die Kette kontinuierlich heruntergezählt.

Ist der TIME COUNTER abgelaufen bevor die Übertragung beendet ist, wird in der Zelle 'STATUS' im Daten-Strukturelement die entsprechende Time-Out-Fehlermeldung gesetzt.
Ist die Übertragung beendet, bevor der COUNTER abgelaufen ist, wird das Strukturelement aus der Time-Out-Kette entfernt.

XTTID..... Diese Zelle wird von der lokalen Firmware nicht belegt. Hier kann der Anwender auf vier Bytes eigene, frei wählbare Parameter ablegen. Ein möglicher Einsatz dieser Zelle ist z.B. die Ablage des Programm-Counters eines Anwenderprogramms in der Zelle.

In den Modes 'Rx-Transfers disabled (Tx-Only)', 'Rx-Transfer mit Time-Out-Option' und 'RTR senden' kann die Zelle 'XTTID' anstelle der Zelle 'EVTRIG' als notwendige Bedingung für das Schreiben in das FIFO 'Data To VME' und somit als Interrupt-Bedingung genutzt werden: Sind die Zellen 'iolev' und 'iovec' des Kommandos 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt, und die Zelle 'XTTID'≠ 0, dann wird beim Eintreten einer gültigen Endebedingung ein VMEbus-Interrupt ausgelöst.

NET LINK/
BUFFER-POINTER... In dieser Zelle trägt die lokale Firmware bei der Verknüpfung zweier CAN-IDs über das Parameter-Kommando 'Link Net...' in vier Bytes die absolute Adresse des jeweiligen Daten-Strukturelementes mit dem es verknüpft ist ein (siehe auch Kap.'Kommandos des Parameter-Buffers').
Wird der Identifier dieses Kontroll-Strukturelementes für CMS-Transfers eingesetzt, so kann in dieser Zelle nach der Buffer-Initialisierung durch 'Init Domain' (siehe auch Kommandoübergabe) die Anfangsadresse des CMS-Buffers gelesen werden. Über diese Startadresse lassen sich die Adressen der relevanten Buffer-Zellen errechnen.

2.3 Rx-Modes

2.3.1 Empfangen von Rx-Daten ohne Interrupt

Bis auf den 'Rx-disable-Mode' der selektierten Identifier ist die CAN2 in allen Betriebsmodes in der Lage, die auf dem CAN-Bus gesendeten Rx-Daten zu empfangen.

Alle empfangenen Daten werden im Daten-Strukturfeld abgelegt und können dort über den VMEbus gelesen werden. Das Daten-Strukturfeld wird ständig mit den eintreffenden Daten aktualisiert. Die gültige Datenlänge in Bytes kann in der Zelle LENGTH des Daten-Strukturelementes nach dem Empfang der Daten gelesen werden. Vor, während und nach der Übertragung kann der aktuelle Status der Übertragung in der Zelle STATUS gelesen werden.

Sollen die Daten einzelner oder mehrerer Identifier nicht abgelegt werden, so kann dies über die Auswahl des Rx-disable-Modes (MODE = 00) für diese Identifier erreicht werden. Der Mode wird über den Parameter-Buffer gesetzt.

Der Aufbau der Elemente des Daten-Strukturfeldes wurde im vorangegangenen Kapitel bereits beschrieben. Das Setzen der Kommandos des Parameter-Buffers wird in dem noch folgenden Kapitel 'Parameter-Buffer' erläutert.

2.3.2 Empfangen von Rx-Daten mit Interrupt-Meldung auf dem VMEbus

Für den Empfang von Rx-Daten mit einer anschließenden Interrupt-Meldung gelten die gleichen Bedingungen wie im vorangegangen Kapitel.

Zusätzlich ist folgendes zu beachten:

1. Generell müssen, um einen VMEbus-Interrupt auslösen zu können, die Zellen 'iolev' (**I**/**O**-Interrupt-**Lev**el) und 'iovec' (**I**/**O**-Interrupt-**Vec**tor) über das Kommando 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sein. Sie initialisieren die Hardware und legen den VMEbus-Interrupt-Level und den Vektor fest. Die Zusammensetzung dieser Zellen und das Setzen der Zellen im Parameter-Buffer wird in dem noch folgenden Kapitel 'Kommandos des Parameter-Buffers' ausführlich erläutert.
2. Im CAN-Daten-Strukturelement des gewünschten Identifiers muß die Zelle 'EVTRIG' gesetzt werden. Alternativ hierzu kann auch die 'XTTID' im Kontroll-Strukturelement auf einen Wert ≠ 0 gesetzt werden. Die Zelle 'XTTID' kann der Anwender beliebig setzen (bis auf '0').
   Die Belegung der Zellen 'EVTRIG' und 'XTTID' ist in den vorangegangenen Kapiteln bereits beschrieben worden.

Mit diesen Voreinstellungen wird der gewünschte VMEbus-Interrupt ausgelöst, wenn eine gültige Endebedingung für den Empfang eintritt. Gültige Endebedingungen sind z.B. 'Daten erfolgreich empfangen' oder 'Daten nicht innerhalb der Time-Out-Zeit empfangen'. Eine Liste der Endebedingungen ist der Beschreibung der Zelle 'EVTRIG' zu entnehmen.

2.3.3 Weitere Optionen für den Empfang von Rx-Daten

Soll eine bestimmte Nachricht auf einem Rx-Identifier innerhalb einer begrenzten Zeit eintreffen, so kann ein Time-Out-Counter gestartet werden, der nach Ablauf einen VMEbus-Interrupt auslösen kann. Um diesen Mode auszuwählen, ist die Time-Out-Zeit (TOUT) im Daten-Strukturelement des Identifiers zu einzutragen. Gestartet wird der Counter durch Setzen der Zelle LENGTH. Der Eintrag der Anzahl Datenbytes in der niederwertigsten HEX-Stelle von LENGTH wird von der Firmware hierbei nicht ausgewertet. (Die Zelle MODE im Parameter-Buffer muß wie bei den vorangegangenen Rx-Modes auf '01' gesetzt sein.) Nach dem Empfang der Daten wird die Zelle LENGTH von der Firmware mit der Länge der gültigen Daten besetzt.

2.4 Tx-Modes

2.4.1 Senden von Tx-Daten ohne Interrupt

Für das Senden von Tx-Daten sollte im Parameter-Buffer für den/die gewünschten Identifier der Mode '00' gewählt werden.

Die zu sendenden Daten werden über den VMEbus in das Daten-Strukturelement des Identifiers eingetragen. Mit dem Setzen der gültigen Datenlänge in die Zelle LENGTH wird die Sendung gestartet. Die in LENGTH eingetragenen Werte haben dabei folgende Bedeutung:

1. Eintrag der Länge als positive Zahl ($0000...$0008):
   Der Eintrag der Anzahl der Bytes als positive Zahl führt nicht zum Start der Abarbeitung des Übertragungsauftrags.
2. Eintrag der Länge als negative Zahl ($FFFF...$FFF7):
   Wird die Anzahl der zu übertragenden Bytes als negative HEX-Zahl eingetragen, so wird die Abarbeitung des Sendeauftrags mit dem Eintrag gestartet.
3. Eintrag der Länge und der Zahl '6' ($0060...$0068):
   Funktion identisch zu 2.

Der aktuelle Status des Sendeauftrags kann in der Zelle STATUS des Daten-Strukturelementes gelesen werden.

2.4.2 Senden von Tx-Daten mit Interrupt-Meldung auf dem VMEbus

Für das Senden von Tx-Daten mit einer anschließenden Interrupt-Meldung gelten die gleichen Bedingungen wie im vorangegangen Kapitel.

Zusätzlich ist folgendes zu beachten:

1. Generell müssen, um einen VMEbus-Interrupt auslösen zu können, die Zellen 'iolev' (**I**/**O**-Interrupt-**Lev**el) und 'iovec' (**I**/**O**-Interrupt-**Vec**tor) über das Kommando 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sein. Sie initialisieren die Hardware und legen den VMEbus-Interrupt-Level und den Vektor fest. Die Zusammensetzung dieser Zellen und das Setzen der Zellen im Parameter-Buffer wird in dem noch folgenden Kapitel 'Kommandos des Parameter-Buffers' ausführlich erläutert.
2. Im CAN-Daten-Strukturelement des gewünschten Identifiers muß die Zelle 'EVTRIG' gesetzt werden. Alternativ hierzu kann auch die 'XTTID' im Kontroll-Strukturelement auf einen Wert ≠ 0 gesetzt werden. Die Zelle 'XTTID' kann der Anwender beliebig setzen (bis auf '0').
   Die Belegung der Zellen 'EVTRIG' und 'XTTID' ist in den vorangegangenen Kapiteln bereits beschrieben worden.

Mit diesen Voreinstellungen wird der gewünschte VMEbus-Interrupt ausgelöst, wenn eine gültige Endebedingung für den Sendeauftrag eintritt. Gültige Endebedingungen sind z.B. 'Daten erfolgreich gesendet' oder 'Daten konnten nicht innerhalb der Time-Out-Zeit auf dem BUS gesendet werden' (Busfehler). Eine Liste der Endebedingungen ist der Beschreibung der Zelle 'EVTRIG' zu entnehmen.

2.4.3 Senden von RTR-Frames

Um bestimmte Rx-Daten anzufordern, kann ein RTR-Frame auf den CAN-Bus gesendet werden. Die Zelle MODE im Parameter-Buffer ist hierzu auf '1' zu setzen. Das RTR wird gesendet, wenn die Zelle LENGTH im Daten-Strukturelement des gewünschten Identifiers entsprechend gesetzt wird (siehe Zelle Beschreibung der LENGTH). Das letzte Nibble von LENGTH enthält dabei die gewünschte Datenlänge der angeforderten Nachricht. Auch bei diesem Transfer kann ein VMEbus-Interrupt ausgelöst werden. Gültige Bedingungen hierzu sind z.B. 'Fehler beim Senden des RTR-Frames' oder 'Empfang der angeforderten Rx-Daten'.

2.4.4 Weitere Optionen für die Sendung von Tx-Daten

Die Sendung kann mit einem Time-Out-Monitor überwacht werden. Erfolgt die Sendung nicht innerhalb der vorgegebenen Zeit erfolgen, so wird ein Time-Out-Fehler in der Zelle STATUS angezeigt. Die Time-Out-Zeit muß im Daten-Strukturelement des gewünschten Identifiers gesetzt werden (siehe dort).

Die CAN2 kann außerdem eine Sendung initiieren, wenn auf einem Identifier ein RTR-Frame empfangen wurde. Hierzu ist MODE im Parameter-Buffer für den entsprechenden Identifier auf '02' zu setzen. Beim Empfang eines RTR-Frames wird das Daten-Strukturelement (0...8 Byte) gesendet. Prinzipiell läuft die Sendung wie bei einem 'Tx-Transfer' in MODE '00' ab. Es gelten die gleichen Endebedingungen.

Unterschiede ergeben sich bei der Interrupt-Erzeugung und der Quelle der Daten, die gesendet werden sollen:

Wird die Zelle 'EVTRIG'≠'0' gesetzt, dann wird nach dem Eintreffen des RTR-Frames ein VMEbus-Interrupt ausgelöst und die zu sendenden Daten müssen vom VMEbus bereitgestellt werden.

Wird die Zelle 'EVTRIG'='0' gesetzt, dann werden die zu sendenden Daten aus dem lokalen Daten-Strukturelement der CAN2 übertragen.

3. Spezielle Funktionen

3.1 Die Monitor-Buffer

3.1.1 Funktion und Aufbau

In den Monitor-Buffern können empfangene Daten sequentiell abgelegt werden, um den zeitlichen Ablauf und die Inhalte der Übertragungen für einen längeren Zeitraum aufzuzeichnen.

Die Auswahl der Identifier die aufgezeichnet werden sollen, wird im CTRL-Strukturfeld des Identifiers durch das Setzen des 'Monitor-Modes' getroffen.

Die Aufzeichnung beginnt mit dem ersten Eintreffen der spezifizierten Trigger-Bedingung. Nach diesem Triggern werden alle folgenden eintreffenden Nachrichten aufgezeichnet, und die Zeit ihres Eintreffens relativ zu dem Trigger-Zeitpunkt mit abgelegt. Die Trigger-Bedingung wird über ein Kommando des Parameter-Buffers, das später noch ausführlich erläutert wird, übergeben.

Die Aufzeichnungen enden, wenn der Monitor-Buffer gefüllt ist.

Die folgende Abbildung zeigt den prinzipiellen Ablauf einer Aufzeichnung.

Initiator	Hardware	Hardware	Firmware	Firmware
Ursache	Senden von Tx-Frames anderer CAN-Teilnehmer	Controller empfängt Frame mit zugelassenem Identifier und löst Rx-IRQ aus	Rx-Interrupt	Eintreffen eines Monitor-Strukturelementes
Wirkung	CAN-Controller empfängt alle gesendeten Frames und filtert Frames mit gewünschtem Identifier heraus	Übertragung der Information an Rx-Software-Filter	Selektieren der empfangenen Strukturen nach Rx-Software-Filter	Sequentielle Ablage der Nachtrichten im Monitor Buffer

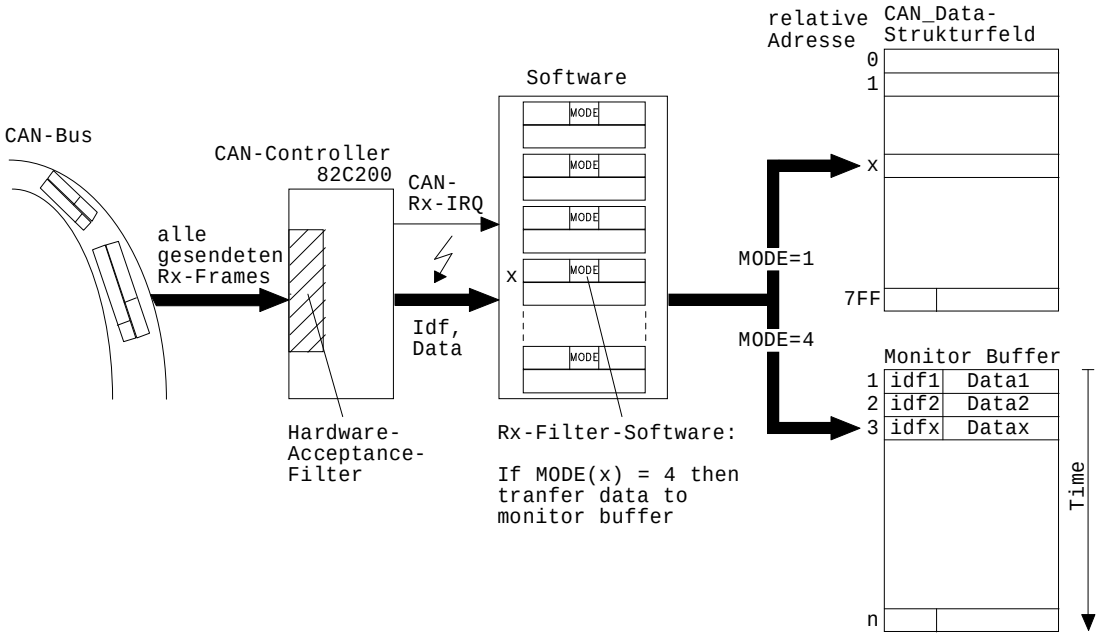


Abb. 3.1.1: Empfang und Ablage von CAN-Bus-Messages im Monitor-Buffer

Jedem Kanal (Net) der CAN2 steht ein Monitor-Buffer zur Verfügung. Die Lage und Größe der Monitor-Buffer ist im Parameter-Buffer abgelegt.

Ein Strukturelement der Monitor-Buffer ist immer 16 Byte lang:

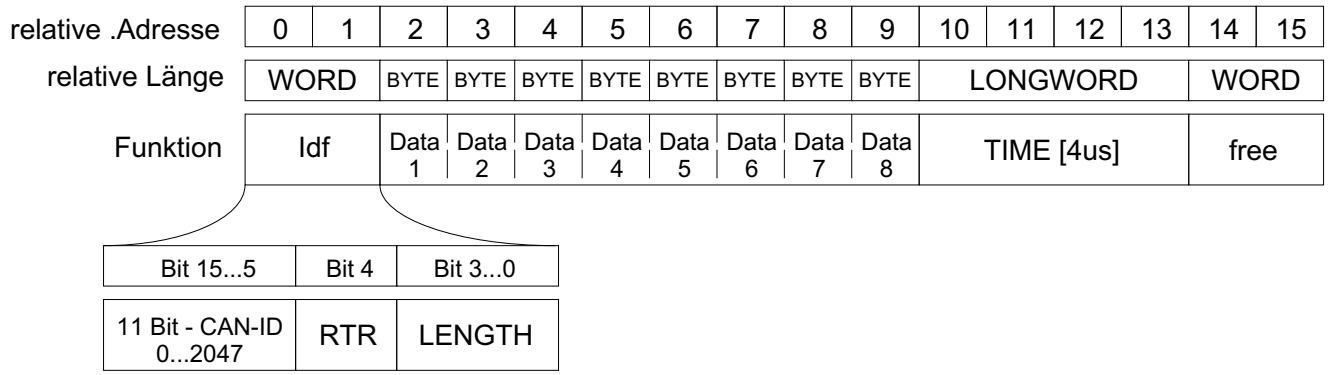


Abb. 3.1.2: Inhalt eines Monitor Strukturelementes

3.1.2 Erklärung der Bytes eines Monitor-Strukturelementes

Im folgenden werden die Bytes eines Monitor-Strukturelementes in aufsteigender Reihenfolge beschrieben.

Idf.....enthält den 11 Bit langen CAN-Id, das RTR-Bit (Remote Transmission Request) und die Anzahl der übertragenen Bytes (LENGTH):

CAN-Id....mögliche Werte: 0...2047
RTR-Bit...zeigt an, ob unter diesem Identifier eine Datenübertragung angefordert wurde.
RTR = '0' ....Data Transfer
RTR = '1' ....Remote Transmission Request
LENGTH....gibt die Anzahl der übertragenen Daten-Bytes an.
Mögliche Werte: 0...8

Data1..Data8....enthalten die übertragenen Daten.
Es können null bis acht Byte übertragen werden.

TIME.....Zeit (in [4µs]-Schritten), die seit dem ersten Eintreffen einer Übertragung in diesem Monitor-Buffer bis zum Eintreffen dieser Daten vergangen ist (='Time-Stamp').

3.2 CAN-Serial-Mode

3.2.1 Funktion

Im CAN-Serial-Mode werden empfangene Daten nicht nur im Strukturfeld abgelegt, sondern außerdem dem VMEbus-Master im FIFO 'Data to VMEbus' zu Verfügung gestellt.

Diese Datenablage bringt dem Master den Vorteil, daß er sequentiell eintreffende Daten eines Identifiers ohne das Risiko des Datenverlustes in der Reihenfolge ihres Eintreffens weiterverarbeiten kann.

Diese Transfermethode wird durch die Nutzung des Pointer-FIFOs (Data to VMEbus) zum Speichern von Daten und das Auslösen eines Interrupts beim Eintreffen des ersten Bytes einer Nachricht ermöglicht. Da bereits abgelegte Daten nicht wie im Daten-Strukturfeld ständig aktualisiert werden, besteht keine Gefahr, daß Daten ungewollt überschrieben werden.

3.2.2 Struktur der CAN-Serial-Data-Blöcke

Der CAN-Serial-Data-Mode wird für den jeweiligen Identifier (Strukturelement) über das zugeordnete Byte 'MODE' im Kontroll-Strukturfeld gewählt. Das Byte 'MODE' muß über den Parameter-Buffer gesetzt werden.

Die lokale Software markiert im FIFO eingetroffene CAN-Serial-Data-Blöcke zur Unterscheidung gegenüber den üblicherweise abgelegten Pointern durch das Setzen einer Marke. Die Marke enthält bereits die Information über das jeweiligen CAN-Netz, auf dem die Daten eingetroffen sind. Die der Marke folgenden Bytes enthalten Statusmeldungen und die empfangenen Daten. Folgen keine weiteren Daten, werden im FIFO nach dem letzten Daten-Byte wieder Pointer oder ein neuer Datenblock, beginnend mit 'Marke', abgelegt.

Beim Empfang eines Remote Requests (RTR) wird der Parameter LENGTH von der Firmware nicht ausgewertet. Es werden keine Daten im FIFO abgelegt. LENGTH kann aber mit einem beliebigen Wert besetzt werden, um z.B. anwenderdefinierte Informationen zu übertragen.

Die folgende Abbildung zeigt als Beispiel die Lage einer Nachricht mit der maximalen Länge von 8 Daten-Bytes im FIFO, der Pointer 'normaler' Transfers vorangehen und folgen.

Es werden nur so viele Bytes in das FIFO geschrieben, wie Daten empfangen worden sind. Es können jedoch immer nur ganze Worte im FIFO abgelegt werden, daher bleibt bei der Übertragung einer ungeraden Anzahl Bytes immer ein Byte (D0...D7) unbenutzt.

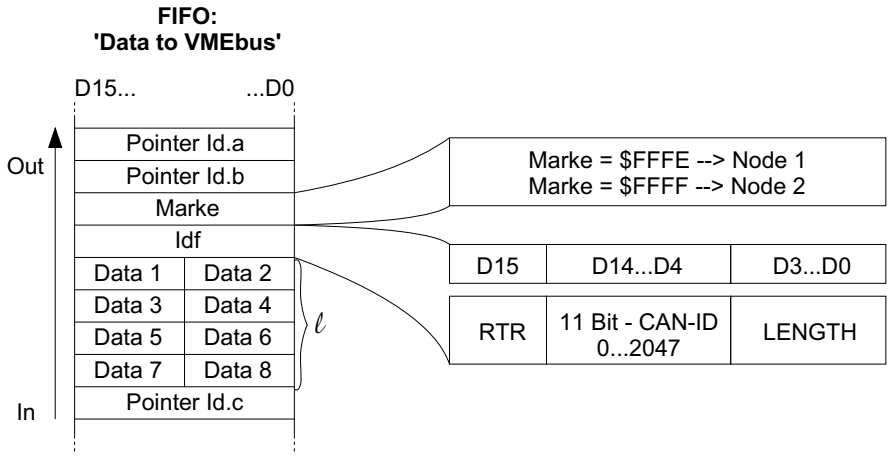


Abb. 3.2.1: Aufbau eines CAN-Serial-Data-Blocks

3.2.3 Erklärung der Bytes eines CAN-Serial-Data-Blocks

```
CAN-Id..........mögliche Werte: 0...2047
RTR-Bit.........zeigt an, ob unter diesem Identifier eine
                Datenübertragung angefordert wurde.
                RTR = '0' ....Data Transfer
                RTR = '1' ....Remote Transmission Request
                              (LENGTH wird nicht ausgewertet)
LENGTH..........gibt die Anzahl der übertragenen Daten-Bytes an.
                Mögliche Werte: 0...8
Data1..Data8....enthalten die übertragenen Daten-Bytes.
                Es können null bis acht Byte übertragen werden.
```

3.2.4 CAN-Serial-Data-Interrupts

Eintreffende CAN-Serial-Data-Blöcke werden dem VMEbus-Master über den CAN-Server-Interrupt angezeigt. Um einen VMEbus-Interrupt auslösen zu können, müssen die Zellen 'iolev' und 'iovec' über das Kommando 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sein.

Im Gegensatz zu den anderen Interrupt-Quellen, die über den CAN-Server-Interrupt auf den VMEbus gegeben werden, wird der CAN-Serial-Interrupt nicht mit EVTRIG oder XTTID maskiert.

Die notwendige Interrupt-Voraussetzung (neben dem Setzen der Zellen 'iolev' und 'iovec' über das Kommando 'Card-Interrupt-Enable' im Parameter-Buffer) ist die Auswahl des Übertragungsmodes CAN-Serial-Data für den gewünschten Identifier. Treffen auf dem Identifier Nachrichten ein, so werden sie im FIFO abgelegt und das FIFO löst den CAN-Server-Interrupt aus.

Die Quittierung des VMEbus-Interrupts wird in dem Kapitel 'CAN-Server-Interrupt' beschrieben.

3.3 Rx-Buffer

3.3.1 Funktion

Im Rx-Buffer können die Daten selektierter Rx-Identifier in der Reihenfolge ihres zeitlichen Eintreffens abgelegt werden.

Wird das Kommando 'Get Rx-Buffer' über den Parameter-Kanal aufgerufen, so wird der Rx-Buffer von der lokalen Firmware eingerichtet. Mit dem Kommando 'Set Mode' können Identifier ausgewählt werden, die in diesem Buffer aufgezeichnet werden sollen. Die Kommandos des Parameter-Buffers werden in einem eigenem Kapitel noch ausführlich beschrieben.

Die Daten der für den Rx-Buffer selektierten Identifier werden nicht im Datenstrukturelement abgelegt.

Die Datenablage im Rx-Buffer erfolgt ähnlich wie im Monitor-Buffer. Die beiden Betriebsarten unterscheiden sich jedoch in einigen wesentlichen Punkten:

	Monitor-Mode	Rx-Buffer
Buffer-Größe	4096 Messages (à max. 16 Byte) (Aufzeichnung endet, wenn Buffer gefüllt)	programmierbar (max. 4096 Messages) (Buffer wird immer wieder neu überschrieben: 'Wrap Around')
Trigger	auf selektierten Identifier	kein Trigger, Aufzeichnung beginnt sofort nach Auswahl des Modes
Time Stamp	4 µs	1024 µs
VMEbus-Interrupt	kein VME-IRQ möglich	VME-IRQ beim Eintreffen der ersten Daten möglich

Tabelle 3.3.1: Gegenüberstellung Monitor-Mode - Rx-Buffer

3.3.2 Struktur des Rx-Buffers

Der Rx-Buffer läßt sich in einen Header- und einen Datenbereich aufteilen. Im Header sind die Pointer und die Anzahl der gültigen Elemente im Buffer abgelegt.

Der Datenbereich beinhaltet neben den empfangenen Daten weitere Parameter wie z.B. Identifier und Net-No..

Relative Adresse [HEX]	Länge	Zugriff	Zelleninhalt
0	WORD	read only	Rx-Write-Pointer (WRP)
2	WORD	read/write	Rx-Read-Pointer (RDP)
4	WORD	read only	Rx-Mask (Anzahl der enthaltenen Elemente)
6 : E	-	-	reserviert
10 (BuffStart) : ??	wird über das Kommando 'Rx-Buffer' programmiert	read only	Datenbereich

Tabelle 3.3.2: Aufteilung des Rx-Buffers

Rx-Write-Pointer... Dieser Pointer zeigt auf die Buffer-Zelle (relativ zu 'BuffStart'), die beim nächsten Empfang eines RxIds beschrieben wird.
Beispiel für die Zählweise der Pointer:
$0000, $0010, $0020,... ..., $0FF0, $0000
Nach einem RESET ist der Pointer auf '0' gesetzt.

Rx-Read-Pointer... Der Read-Pointer zeigt auf die Zelle, die als nächste ausgelesen ist. Nach einem RESET ist der Pointer auf '0' gesetzt.

Der Inhalt des Rx-Write-Pointers und des Rx-Read-Pointers werden als Bedingung zum Schreiben des FIFOs 'Data To VME' und somit auch als Bedingung für das Auslösen eines VMEbus-Interrupts herangezogen: Nach dem Eintreffen neuer Daten im Zustand 'Rx-Read-Pointer = Rx-Write-Pointer' wird das FIFO gesetzt.

Rx-Mask... Dieser Parameter enthält die Größe des Datenbereiches des Rx-Buffers. Rx-Mask wird aus dem Wert 'Buffer-Length', den der Anwender bei der Buffer-Initialisierung im Parameter-Buffer vorgeben kann (siehe auch Beschreibung des Parameter-Buffers), errechnet: Der vorgegebene Wert wird von der Firmware auf Werte von 2^n aufgerundet und in Rx-Mask eingetragen (also 2, 4, 8, 16, 32...).

Datenbereich... Im Datenbereich werden die empfangenen Daten abgelegt. Für jeden Rx-Identifier steht eine Zeile mit 16 Bytes zur Ablage von Statusinformationen und Daten zur Verfügung. Die folgende Abbildung zeigt den Aufbau einer Zeile:

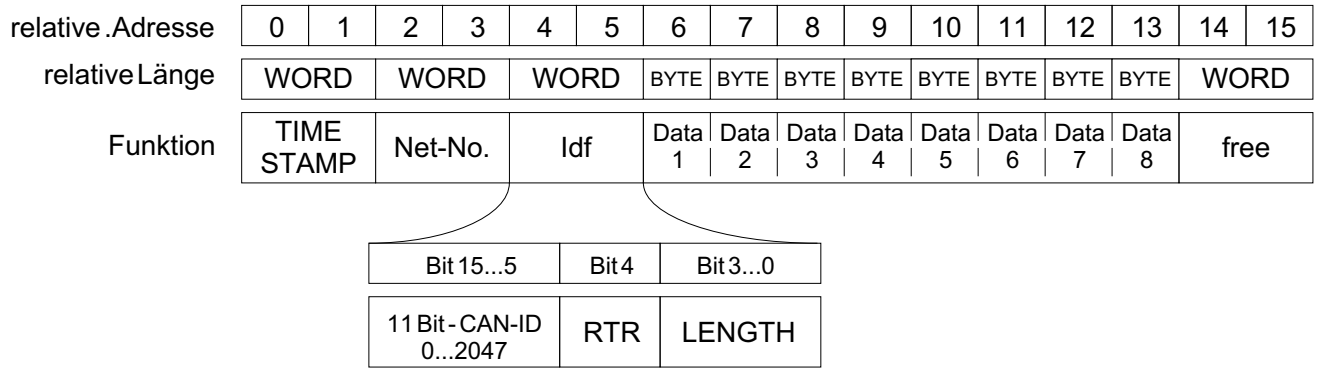


Abb. 3.3.1: Aufbau des Datenbereiches eines Identifiers

TIME STAMP..... Zeit (in [1024µs]-Schritten), die seit dem letzen System-Start (RESET oder Power-On) vergangen ist.

Net-No...... Nummer des CAN-Netzes, auf dem der RxId empfangen wurde (lokale CAN-Schnittstelle 0/1).

Idf..... enthält den 11 Bit langen CAN-Id, das RTR-Bit (Remote Transmission Request) und die Anzahl der übertragenen Bytes (LENGTH):

CAN-Id... mögliche Werte: 0...2047
RTR-Bit... zeigt an, ob unter diesem Identifier eine Datenübertragung angefordert wurde.
RTR = '0'... Data Transfer
RTR = '1'... Remote Transmission Request
LENGTH... gibt die Anzahl der übertragenen Daten-Bytes an. Mögliche Werte: 0...8

Data1..Data8... enthalten die empfangenen Daten.

3.4 Tx-Buffer

3.4.1 Funktion

Im Tx-Buffer können mehrere Sende-Nachrichten abgelegt werden, die nacheinander auf den CAN-Bus gesendet werden sollen. Der Vorteil gegenüber den herkömmlichen Tx-Transfers liegt darin, daß die CAN2 weniger Interrupts auf dem VMEbus auslöst und somit der VMEbus-Master weniger belastet wird:
Bei herkömmlichen Tx-Transfers wird jeder erfolgreiche Transfer durch einen Interrupt angezeigt. Bei Tx-Transfers aus dem Tx-Buffer wird nur *ein* Interrupt nach dem Senden aller Nachrichten des Buffers ausgelöst.

Wird das Kommando 'Get Tx-Buffer' über den Parameter-Kanal aufgerufen, so wird der Tx-Buffer von der lokalen Firmware eingerichtet. Während des Aufrufes wird der CAN-Kanal festgelegt, auf dem gesendet werden soll.

Mit dem Kommando 'Set Mode' werden die Identifier ausgewählt, deren Daten in diesen Buffer eingetragen werden sollen. Die Kommandos des Parameter-Buffers werden in einem eigenen Kapitel noch ausführlich beschrieben.

	Tx-Buffer
Buffer-Größe	programmierbar (max. 4096 Messages) (Buffer wird immer wieder neu überschrieben: 'Wrap Around')
Trigger	Sendung aller Nachrichten des Tx-Buffers, nachdem ein herkömmlicher Tx-Transfer für eine beliebige, im Buffer enthaltene, Nachricht gestartet wurde
VMEbus-Interrupt	VME-IRQ nach dem erfolgreichen Senden aller Nachrichten des Tx-Buffers möglich

Tabelle 3.4.1: Eigenschaften des Tx-Buffers

3.4.2 Struktur des Tx-Buffers

Der Tx-Buffer läßt sich in einen Header- und einen Datenbereich aufteilen. Im Header werden die Pointer und die Anzahl der gültigen Elemente im Buffer abgelegt.

Der Datenbereich beinhaltet neben den empfangenen Daten weitere Parameter wie z.B. Identifier und Net-No..

Relative Adresse [HEX]	Länge	Zugriff	Zelleninhalt
0	WORD	read only	Tx-Write-Pointer (WRP)
2	WORD	read/write	Tx-Read-Pointer (RDP)
4	WORD	read only	Tx-Mask (Anzahl der enthaltenen Elemente)
6 : E	-	-	reserviert
10 (BuffStart) : ??	wird über das Kommando 'Tx-Buffer' programmiert	read only	Datenbereich

Tabelle 3.4.2: Aufteilung des Tx-Buffers

Tx-Write-Pointer... Dieser Pointer zeigt auf die Buffer-Zelle (relativ zu 'BuffStart'), die als nächste beschrieben werden soll. Diese Zelle muß vom Anwender versorgt werden. Beispiel für die Zählweise der Pointer: $0000, $0010, $0020,... ..., $0FF0, $0000 Nach einem RESET ist der Pointer auf '0' gesetzt.

Tx-Read-Pointer... Der Read-Pointer zeigt auf die Zelle, die als nächste gesendet wird. Diese Zelle wird von der Firmware gesetzt. Nach einem RESET ist der Pointer auf '0' gesetzt.

Die Inhalte des Tx-Write-Pointers und des Tx-Read-Pointers werden als Bedingung zum Triggern des VMEbus-Interrupts herangezogen.

Tx-Mask... Dieser Parameter enthält die Größe des Datenbereiches des Tx-Buffers. Tx-Mask wird aus dem Wert 'Buffer-Length', den der Anwender bei der Buffer-Initialisierung im Parameter-Buffer vorgeben kann (siehe auch Beschreibung des Parameter-Buffers), errechnet: Der vorgegebene Wert wird von der Firmware auf Werte von 2^n aufgerundet und in Tx-Mask eingetragen (also 2, 4, 8, 16, 32...).

Datenbereich... Im Datenbereich werden die empfangenen Daten abgelegt. Für jeden Tx-Identifier steht eine Zeile mit 16 Bytes zur Ablage von Statusinformationen und Daten zur Verfügung. Die folgende Abbildung zeigt den Aufbau einer Zeile:

relative Adresse	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
relative Länge	WORD		WORD		BYTE	BYTE	BYTE	BYTE	BYTE	BYTE	BYTE	BYTE	WORD		WORD	
Funktion	11 Bit - CAN-ID 0...2047		LENGTH		Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	reserved		reserved	

Abb. 3.4.1: Aufbau des Datenbereiches eines Identifiers im Tx-Buffer

CAN-Id..... enthält den 11 Bit langen CAN-Id
CAN-Id... mögliche Werte: 0...2047

LENGTH..... gibt die Anzahl der zu übertragenden Daten-Bytes an. Mögliche Werte: 0...8

Data1..Data8... enthalten die zu sendenden Daten.

reserved..... Diese Zellen sind für zukünftige Anwendungen reserviert

3.5 CMS-Implementierung

Die CMS-Implementierung ist an dem Anhang '.cms' an der EPROM-Bezeichnung zu erkennen.

3.5.1 CMS-Domain-Transfers

CMS-Domain-Transfers dienen der Übertragung von Datenblöcken über den CAN-Bus. Für die Übertragung wird ein Rx-Identifier und ein Tx-Identifier benötigt. Der Transfer wird von einem 'Client' eingeleitet und gesteuert und von einem 'Server' bedient. Die Daten können dabei in beide Richtungen übertragen werden.
Die CAN2 kann bei CMS-Domain-Transfers als Client und als Server arbeiten.

Die Auswahl des Netzes und der Identifier für den Domain-Transfers erfolgt über das Kommando 'Init Domain' des Parameter-Buffers. Mit diesem Kommando wird auch die Größe des zur Datenablage und Parameterübergabe benötigten Speicherbereiches (CMS-Buffer) definiert. Eine ausführliche Beschreibung dieses Kommandos ist dem noch folgenden Kapitel 'Kommandos des Parameter-Buffers' zu entnehmen.

Neben dem Parameter-Buffer benötigt der Anwender zur Steuerung der lokalen Firmware weitere Parameter. So ist z.B. in der Zelle NET LINK/BUFFER-POINTER der für den Up- und Download genutzten Identifier die Startadresse des 'CMS-Buffers' abgelegt. Der CMS-Buffer enthält weitere Steuerparameter und dient der Datenablage während der CMS-Transfers. Er wird in dem folgenden Kapitel ausführlich erläutert.

Die Daten-Strukturelemente des Rx- und des Tx-Identifiers werden teilweise wie bei herkömmlichen Übertragungen genutzt. So werden z.B. Tx-Transfers über LENGTH gestartet und der Erfolg einer Übertragung kann nach dem Transfer in der Zelle Status abgefragt werden. In TOUT müssen die entsprechenden Rx- und Tx-Time-Out-Zeiten eingetragen werden.

Die folgende Abbildung zeigt die Strukturelemente und (andeutungsweise) den CMS-Buffer.

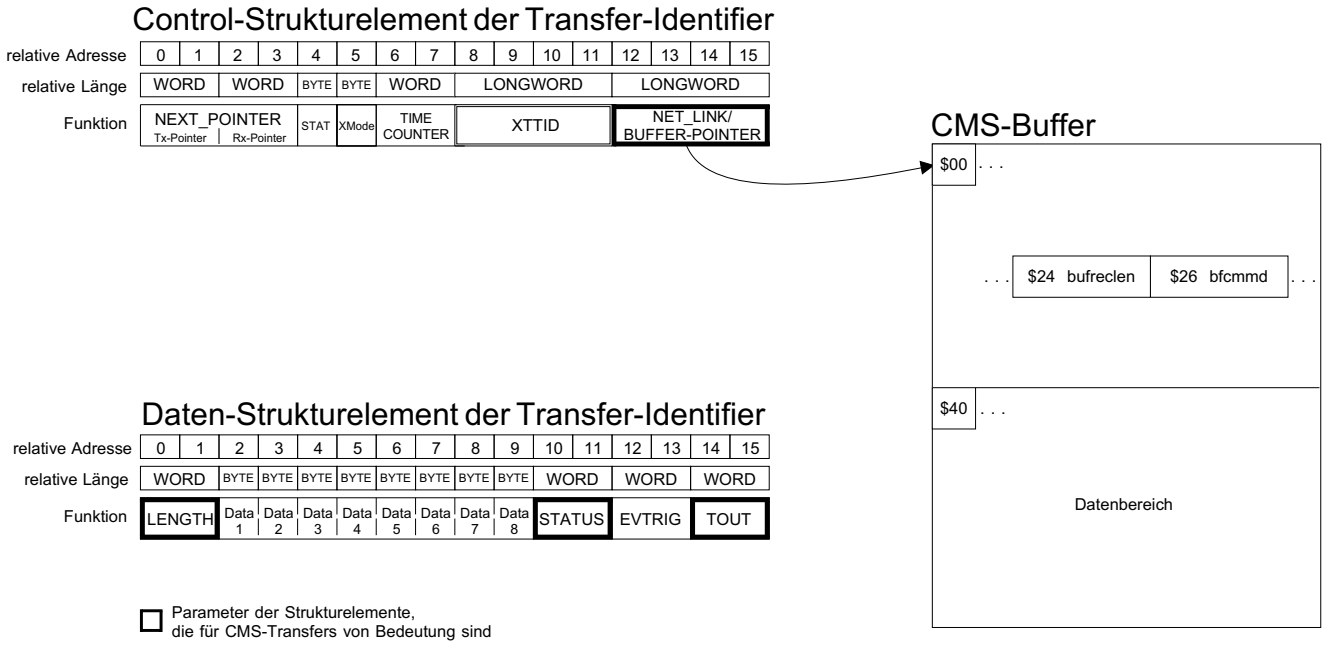


Abb. 3.5.1: Wichtige Parameter-Zellen für CMS-Transfers

3.5.1.1 CMS-Buffer

Der CMS-Buffer dient zur Ablage der zu sendenden oder zu empfangenden Daten und zur Übergabe von Kommandos und Statusmeldungen.

Die folgende Tabelle zeigt den prinzipiellen Aufbau des CMS-Buffers. Die in Klammern () angegebenen Parameter werden von der lokalen Firmware benötigt und sind für den Anwender ohne Bedeutung. Sie dürfen von Anwender nicht verändert werden!

relative Adresse [HEX]	Länge	Zugriff	Zellen-inhalt	Kurzbeschreibung
00 : 1A	-	read only	-	Diese Parameter sind für die lokale Firmware vorgesehen und dürfen von Anwender nicht verändert werden!
1C	LWORD	read only	**buflen**	Maximale Länge des Datenbereichs dieses Buffers in Byte (wird über Init Domain vorgegeben)
20	LWORD	read only	**bufadr**	Absolute Startadresse des Datenbereiches
24	WORD	read/ write	**bfreclen**	Anzahl der gültigen Daten im Buffer Tx -▸ Anzahl der zu sendenden Daten Rx -▸ Anzahl der empfangenen Daten
26	WORD	read/ write	**bfcmmd**	Idle : 0 Client Download : 1 Client Upload : 2 Server-Mode : 3
28	LWORD	read/ write	**bfmux**	Multiplexer-Wert für Init Domain Up/Download ($0000.0000 ... $00FF.FFFF)
2C	WORD	read/ write	**bflast**	Setzen, um beim letzten Datensegment CMS-Ende-Bits (≠ 0 -▸ aktiv) mit zu übertragen, bzw. Lesen um Ende-Bit abzufragen
2E	WORD	read only	**bfsvst**	Server-Status 1: sv down 2: sv up
30	BYTE	read only	(bfstat)	interner Status (≠ 0 : initiated)
31	BYTE		(bfxact)	-
32	WORD		(bfcont)	aktueller Zähler für Datentransfer
34	BYTE		(bftbit)	Toggle-Bit
35	BYTE		(bfssol)	-
36, 37	BYTE		reserved	-
38	LWORD		(bftxid)	Pointer auf Transmit Control Struktur
3C	LWORD		(bfrxid)	Pointer auf Receive Control Struktur
40	wird über Kommando 'Init Domain' vorgege-ben	read/ write	Daten-bereich	Lesen empfangener Daten und Schreiben von Daten, die gesendet werden sollen

Tabelle 3.5.1: Aufbau des CMS-Buffers

3.5.1.2 Ablauf der Transfers

3.5.1.2.1 Zu den folgenden Beschreibungen

In diesem Kapitel wird der Ablauf der Up- und Downloads beschrieben. Um das Verständnis der eingesetzten Firmware und der vom Anwender zur Verfügung zu stellenden Software zu erleichtern sind für alle Betriebsarten Ablaufdiagramme eingefügt.

> Die Ablaufdiagramme der VMEbus-Master-Software, die nicht im Lieferumfang enthalten ist, sind als Anregung und Hilfestellung zu verstehen: Die Bedienung der CAN2-Firmware kann teilweise auch anders realisiert werden.

Die Ablaufdiagramme der lokalen CAN2-Firmware zeigen den prinzipiellen Ablauf der lokalen Programme. Auf eine ausführliche Programmbeschreibung wird aus Gründen der Übersichtlichkeit hier verzichtet.

> Die beschriebenen VMEbus-Interrupts der CAN2 werden nur ausgelöst, wenn die Zellen 'iolev' und 'iovec' im Parameter-Buffer vorher entsprechend gesetzt worden sind!
> In jedem Fall führt die Interrupt-Ursache auch zum Schreiben des betroffenen Identifiers in das FIFO 'Data To VME'.

3.5.1.2.2 Initialisierung des CMS-Transfer-Buffers

Unabhängig von der Art der folgenden Transfers, sind die ersten Schritte immer gleich:
Zu Anfang muß der CMS-Buffer initialisiert werden. Der Buffer wird üblicherweise nur einmal nach dem Systemstart initialisiert und bleibt bis zum nächsten RESET erhalten:

- Aufruf des Kommandos 'Init Domain'
  Mit diesem Kommando des Parameter-Buffers wird eines der zwei Netze der CAN2 ausgewählt und die Größe des Daten-Buffers vorgegeben. Außerdem wird der Rx-Identifier und der Tx-Identifier für den Transfer ausgewählt.

- Nach der Ausführung des Kommandos kann die absolute lokale Startadresse des CMS-Buffers im dem Rückgabeparameter des Parameter-Buffers 'retpar' oder in der Zelle 'NET LINK/BUFFER-POINTER' im Kontroll-Strukturelement der eingesetzten Identifier gelesen werden.

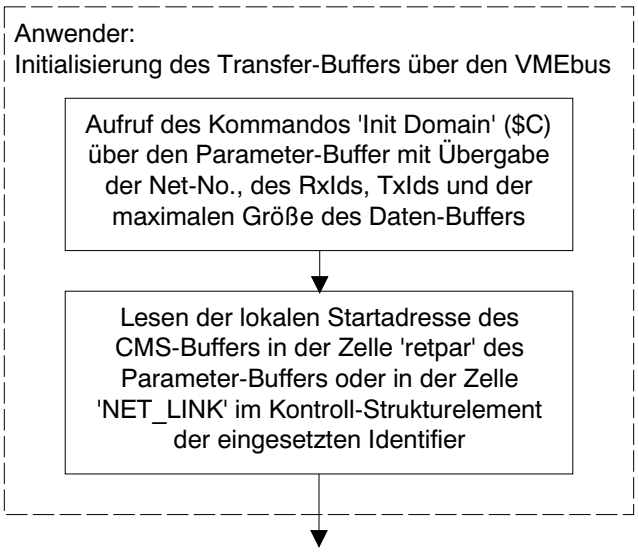


Abb. 3.5.2:
Initialisierung des CMS-Buffers und Lesen der Buffer-Startadresse

3.5.1.2.3 Client Download

Das folgende Bild zeigt den prinzipiellen Ablauf eines Client Downloads.

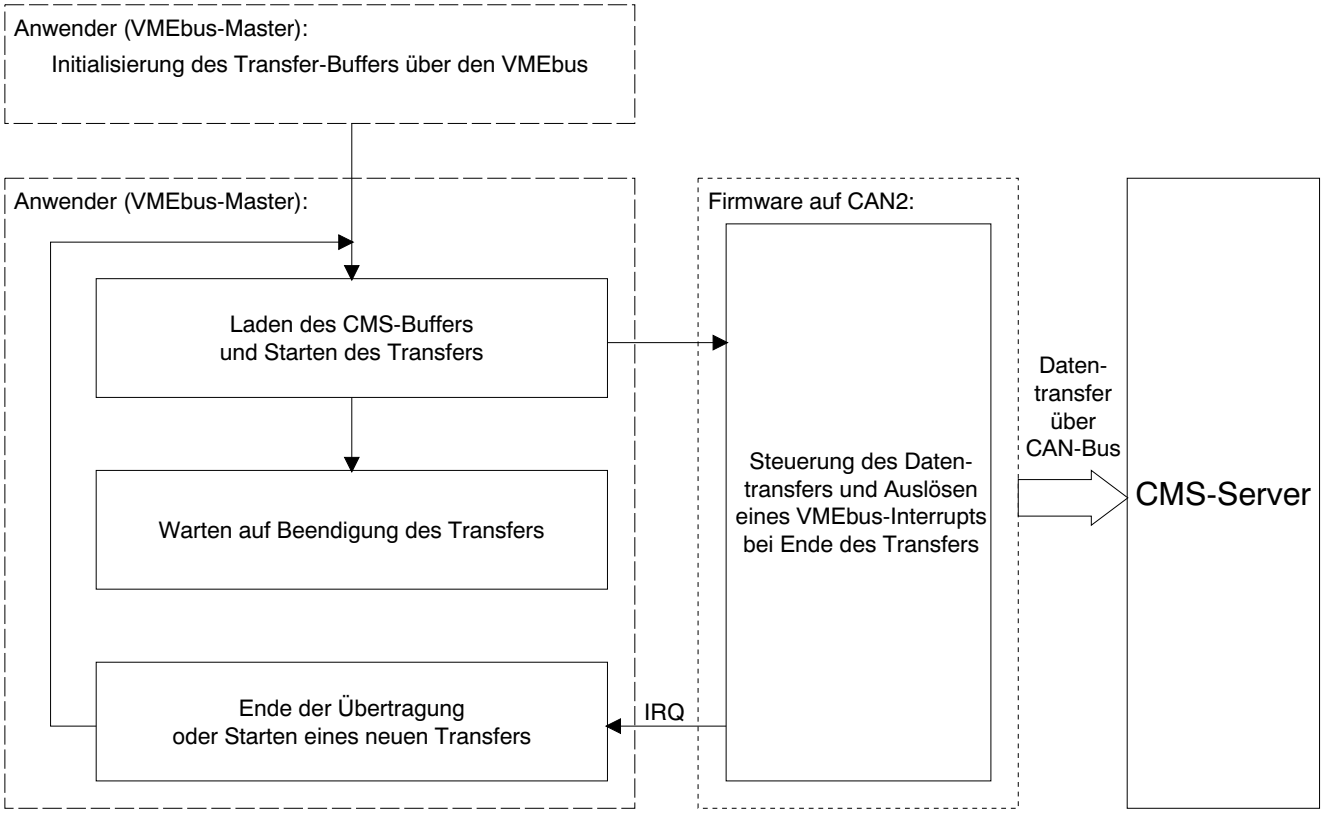


Abb. 3.5.3: Übersicht des Client Downloads

Voraussetzung für einen Client Download ist die Initialisierung des CMS-Buffers.

Der Download erfolgt im wesentlichen in drei Schritten: Über den VMEbus wird der Daten-Buffer auf der CAN2 geladen und der Transfer gestartet. Die CAN2 überträgt die Daten zum CMS-Server und löst einen VMEbus-Interrupt nach Abschluß des Transfers aus. Der Master 'wartet' auf diesen Interrupt, überprüft den Erfolg der Übertragung und startet gegebenenfalls weitere Transfers

Im folgenden werden die einzelnen Blöcke der vorangegangenen Abbildung näher beschrieben:

Die 'Initialisierung des Transfer-Buffers' wurde bereits in dem vorangegangenen Kapitel erläutert.

Die im folgenden dargestellte Abbildung zeigt beispielhaft den Programmablauf für den VMEbus-Master während eines Client Downloads. Dieser Teil ist nicht im Lieferumfang der CAN2 enthalten und muß daher vom Anwender selbst realisiert werden.

Nachdem die Zellen 'bfcmmd' und 'bfmux' gesetzt sind, kann die Übertragung der Daten erfolgen. Da diese Übertragungssequenz auch für den 'Server Upload' eingesetzt werden kann, empfiehlt es sich, diesen Programmteil als Unterprogramm abzulegen. Dieses Modul ist hier 'BLOCK-OUT' betitelt. Das Ablaufdiagramm zu BLOCK-OUT wird im Anschluß an das des VMEbus-Masters besprochen.

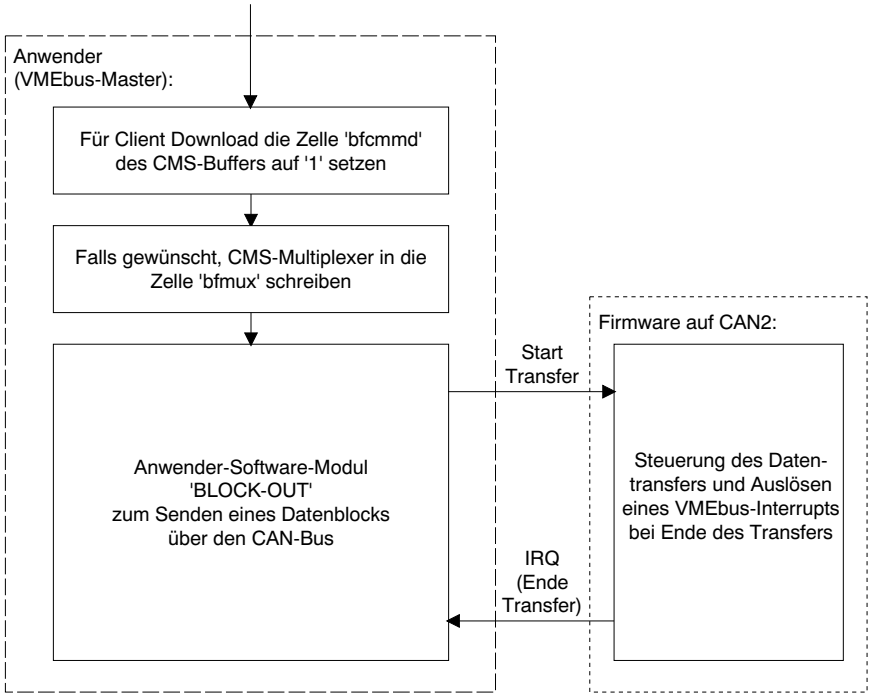


Abb. 3.5.4: VMEbus-Master beim Client Download

Zum Anwender-Software-Modul 'BLOCK-OUT' (folgende Abbildung):

Der hier beispielhaft aufgeführte Programmteil erfüllt folgende Funktionen:

- Laden der zu übertragenden Daten in den Datenbereich des CMS-Buffers und Eintragen der Anzahl der Bytes in die Zelle 'bfreclen' im CMS-Buffer

- Ist dies bereits der letzte Datenblock, so muß die Zelle 'bflast' auf einen Wert ungleich Null gesetzt werden. Bei der Übertragung des letzten Segmentes wird die CAN2-Firmware dann das CMS-Ende-Bit entsprechend setzen.

- Gestartet wird der Download durch das einmalige Triggern eines Tx-Transfers auf dem gewählten Tx-Identifier. Für den Tx-identifier und für die Rückantwort auf dem Rx-Identifier müssen vorher entsprechende Time-Out-Zeiten in den Daten-Strukturelementen gesetzt werden.
  Das Starten des Transfers kann z.B. durch Setzen der Zelle 'LENGTH' im Daten-Strukturelement auf einen negativen Wert oder auf einen Wert zwischen $0060 und $0068 erfolgen. Der absolute Wert von LENGTH und die im Strukturelement eingetragenen Daten dieses TxIds sind in diesem Fall ohne Bedeutung.
  Mit diesen Voreinstellungen erkennt die lokale Software, daß es sich bei dem Transfer um ein CMS-Dienst handelt und nicht um einen Standard-Tx-Transfer und beginnt, den Inhalt des Daten-Buffers gemäß dem CMS-Protokoll zu senden.

- Hat die lokale Firmware der CAN2 den Download der Daten des CMS-Buffers beendet, so löst die CAN2 einen VMEbus-Interrupt aus. Der Download wird beendet, wenn alle Daten übertragen sind, oder wenn während der Übertragung ein Fehler aufgetreten ist. Mögliche Fehler-Ursachen sind z.B. ein Tx-Error oder die fehlende Antwort des CMS-Servers.

  Der VMEbus-Master muß diesen Interrupt bedienen und kann danach die Zelle 'Status' im Daten-Strukturelement lesen, um zu überprüfen, ob die Übertragung erfolgreich war.
  Bei dem Interrupt handelt es sich um einen CAN-Server-Interrupt der CAN2. Im FIFO findet sich der Identifier, der den Interrupt ausgelöst hat. Das Rücksetzen von Interrupts der CAN2 wird in dem noch folgenden Kapitel 'Zusammenfassung der Interrupt-Optionen' beschrieben.Die Zustände der Zelle Status sind der vorangegangenen Beschreibung des Daten-Strukturelementes zu entnehmen. Das Vorgehen bei fehlerhafter Übertragung sei hier dem Anwender überlassen.
- Da die Größe des CMS-Daten-Buffers begrenzt ist, werden u. U. mehrere Transferzyklen benötigt, um die gewünschten Daten zu übertragen. In diesem

Fall kann an den Anfang des Moduls gesprungen werden. Der neue Transfer beginnt wieder mit dem Laden des Buffers.

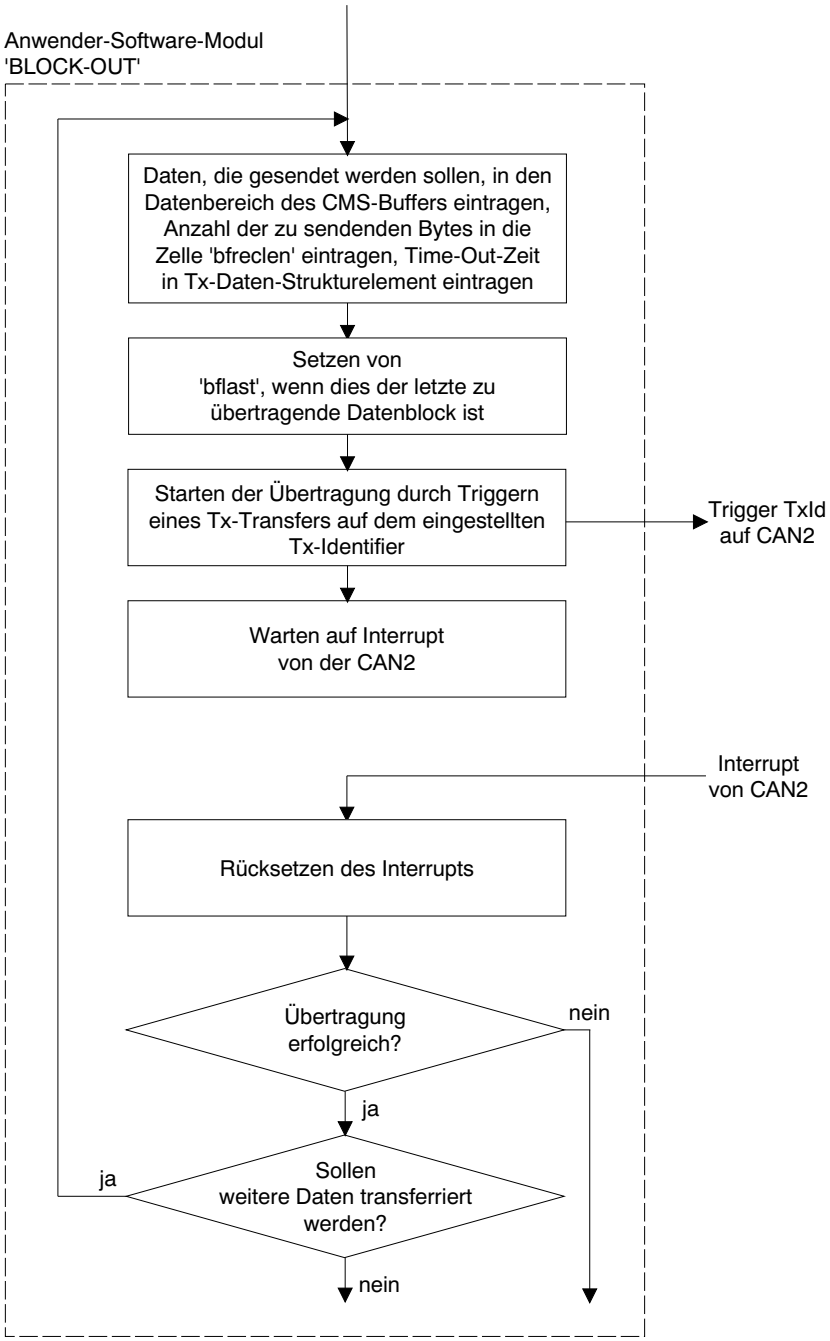


Abb. 3.5.5: Anwender-Software-Modul 'BLOCK-OUT'

In der folgenden Abbildung ist der Ablauf der lokalen Firmware für einen Client Download dargestellt. Die Firmware sendet selbständig die notwendige Init-Sequenz an den angeschlossenen Server und steuert die Übertragung der Daten.

Ist die Zelle 'bflast' des CMS-Buffers vom Anwender gesetzt worden, so wird das Ende-Bit bei der letzten Übertragung gesetzt.

Nach der letzten Übertragung mit dem Ende-Bit, setzt die Firmware die Zelle 'bfcmmd', die vom Anwender auf '1' gesetzt worden war, auf '0' zurück. Die Zelle 'bfcmmd' wird auch zurückgesetzt, wenn die Übertragung wegen eines Fehlers abgebrochen wurde.

Das Ende der Übertragung wird dem Anwender über einen Interrupt mitgeteilt.

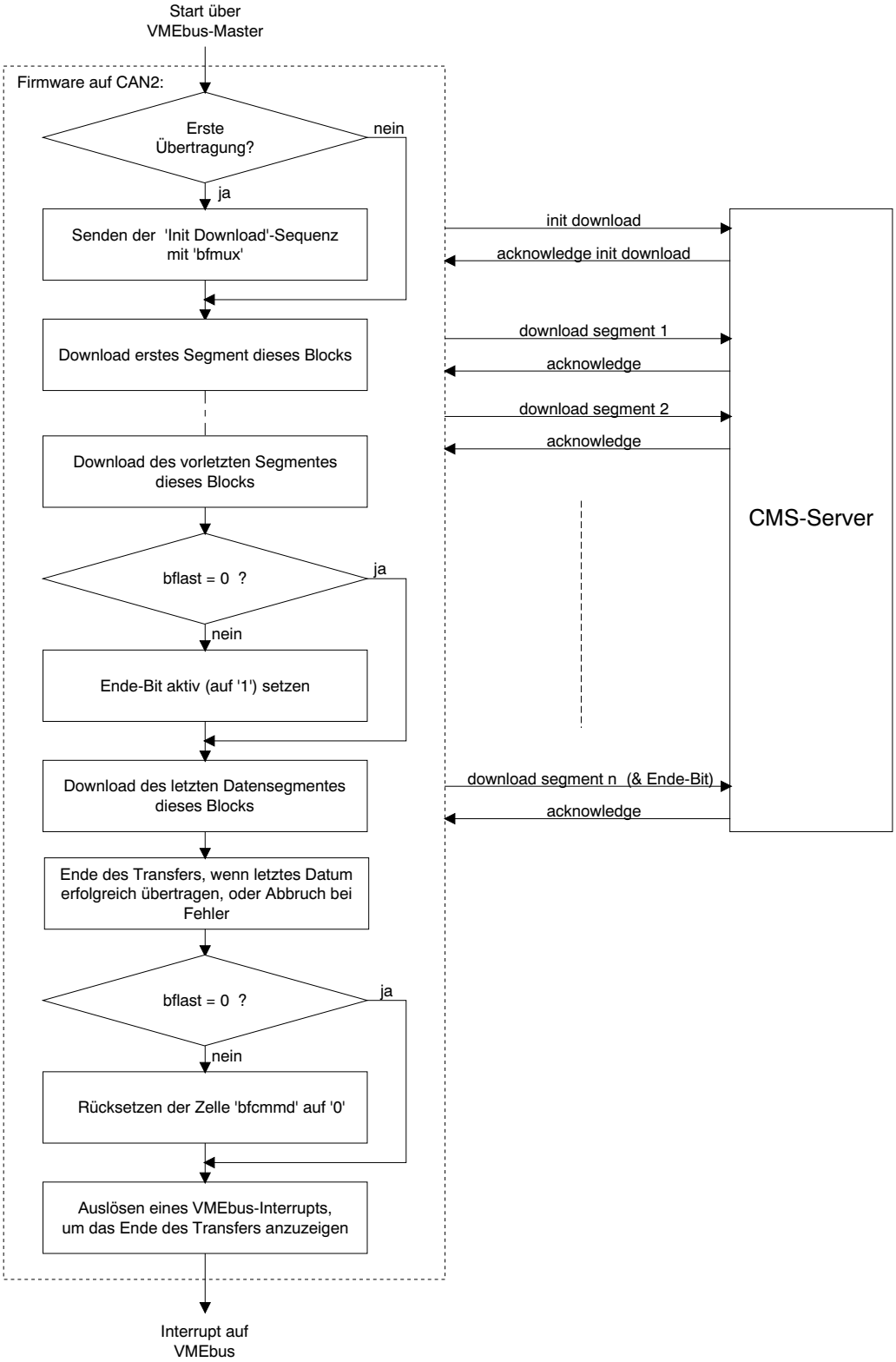


Abb. 3.5.6: Ablauf der lokalen CAN2-Firmware für Client Download-Transfers

3.5.1.2.4 Client Upload

Das folgende Bild zeigt den prinzipiellen Ablauf eines Client Uploads.

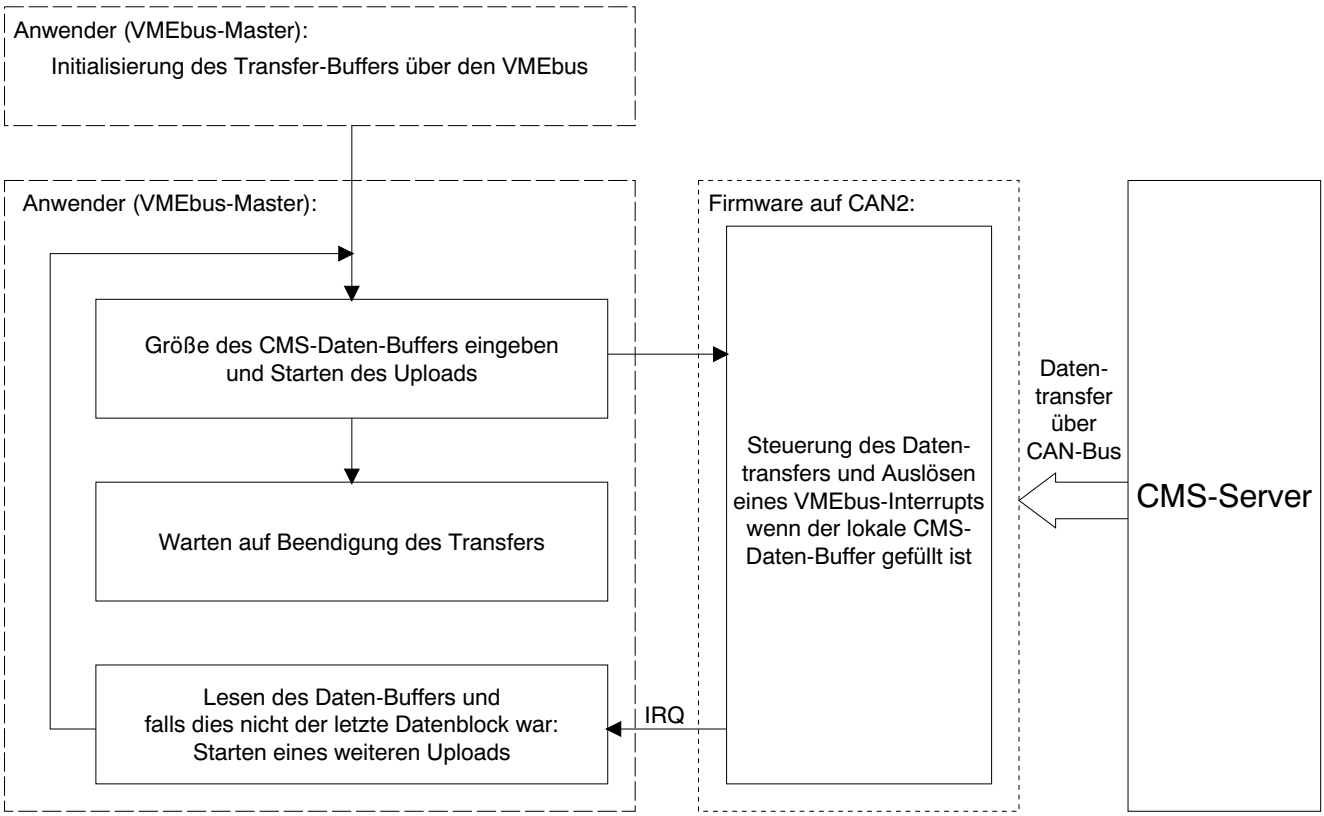


Abb. 3.5.7: Übersicht des Client Uploads

Voraussetzung für einen Client Upload ist die Initialisierung des CMS-Buffers.

Der Upload erfolgt im wesentlichen in drei Schritten: Über den VMEbus wird die Größe des Daten-Buffers auf der CAN2 vorgegeben und der Transfer gestartet. Die CAN2 empfängt die Daten vom CMS-Server und löst einen VMEbus-Interrupt aus, wenn der Buffer gefüllt ist. Der Master 'wartet' auf diesen Interrupt, überprüft den Erfolg der Übertragung, liest den Buffer aus und startet gegebenenfalls weitere Uploads.

Im folgenden werden die einzelnen Blöcke der vorangegangenen Abbildung näher beschrieben:

Die 'Initialisierung des Transfer-Buffers' wurde bereits in einem vorangegangenen Kapitel erläutert.

Die im folgenden dargestellte Abbildung zeigt beispielhaft den Programmablauf für den VMEbus-Master während eines Client Uploads. Dieser Programmteil ist nicht im Lieferumfang der CAN2 enthalten und muß daher vom Anwender selbst realisiert werden.

Nachdem die Zelle 'bfcmmd' gesetzt ist, kann die Übertragung der Daten erfolgen. Da diese Übertragungssequenz auch für den 'Server Download' eingesetzt werden kann, empfiehlt es sich, diesen Programmteil als Unterprogramm abzulegen. Dieses Modul ist hier 'BLOCK-IN' betitelt. Das Ablaufdiagramm zu BLOCK-IN wird im Anschluß an das des VMEbus-Masters besprochen.

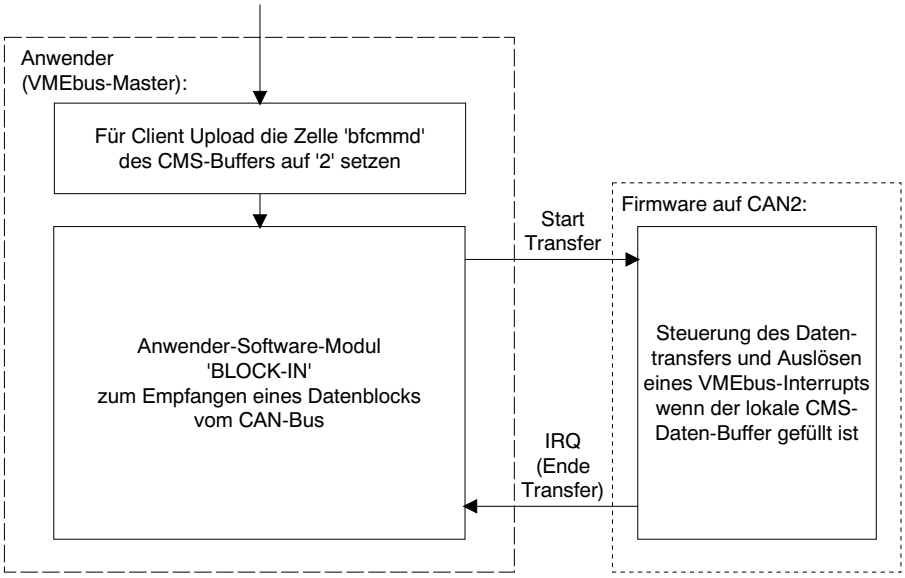


Abb. 3.5.8: VMEbus-Master beim Client Upload

Zum Anwender-Software-Modul 'BLOCK-IN' (folgende Abbildung):

Der hier beispielhaft aufgeführte Programmteil erfüllt folgende Funktionen:

- Zu Anfang wird die maximale Anzahl der zu empfangenden Daten in die Zelle 'bfreclen' des CMS-Buffers eingetragen. Für 'bfreclen' kann der gesamte Inhalt des Datenbereiches genutzt werden. Dieser Wert kann in der Zelle 'bflen' gelesen werden. Wenn die Anzahl der geladenen Daten 'bfrecl - 6' erreicht hat oder das Ende-Bit empfangen wurde, wird die Übertragung beendet. Tritt während des Uploads ein Fehler auf, wird die Übertragung abgebrochen.

- Das Starten des Uploads erfolgt durch das Triggern eines Tx-Transfers. Der Ablauf ist identisch zu dem des Software-Moduls 'BLOCK-OUT und kann dort' nachgelesen werden.

- Nachdem der Daten-Buffer der CAN2 gefüllt ist, meldet sich die Karte mit einem VMEbus-Interrupt zurück. Der Interrupt muß vom VMEbus-Master bedient werden. Die Interrupt-Behandlung auf der CAN2 wird in dem noch folgenden Kapitel 'Zusammenfassung der Interrupt-Optionen' ausführlich beschrieben.

- Durch Abfragen der Zelle 'Status' im Daten-Strukturelement des Rx-Identifiers wird ermittelt, ob die Übertragung erfolgreich war. Die Zustände der Zelle Status sind der vorangegangenen Beschreibung des Daten-Strukturelementes zu entnehmen. Das Vorgehen bei fehlerhafter Übertragung sei hier dem Anwender überlassen.

- Jetzt wird der CMS-Daten-Buffer ausgelesen.

- Die Abfrage der Zelle 'bflast' gibt Auskunft darüber, ob noch weitere Daten folgen sollen, oder ob dies der letzte Datenblock war.

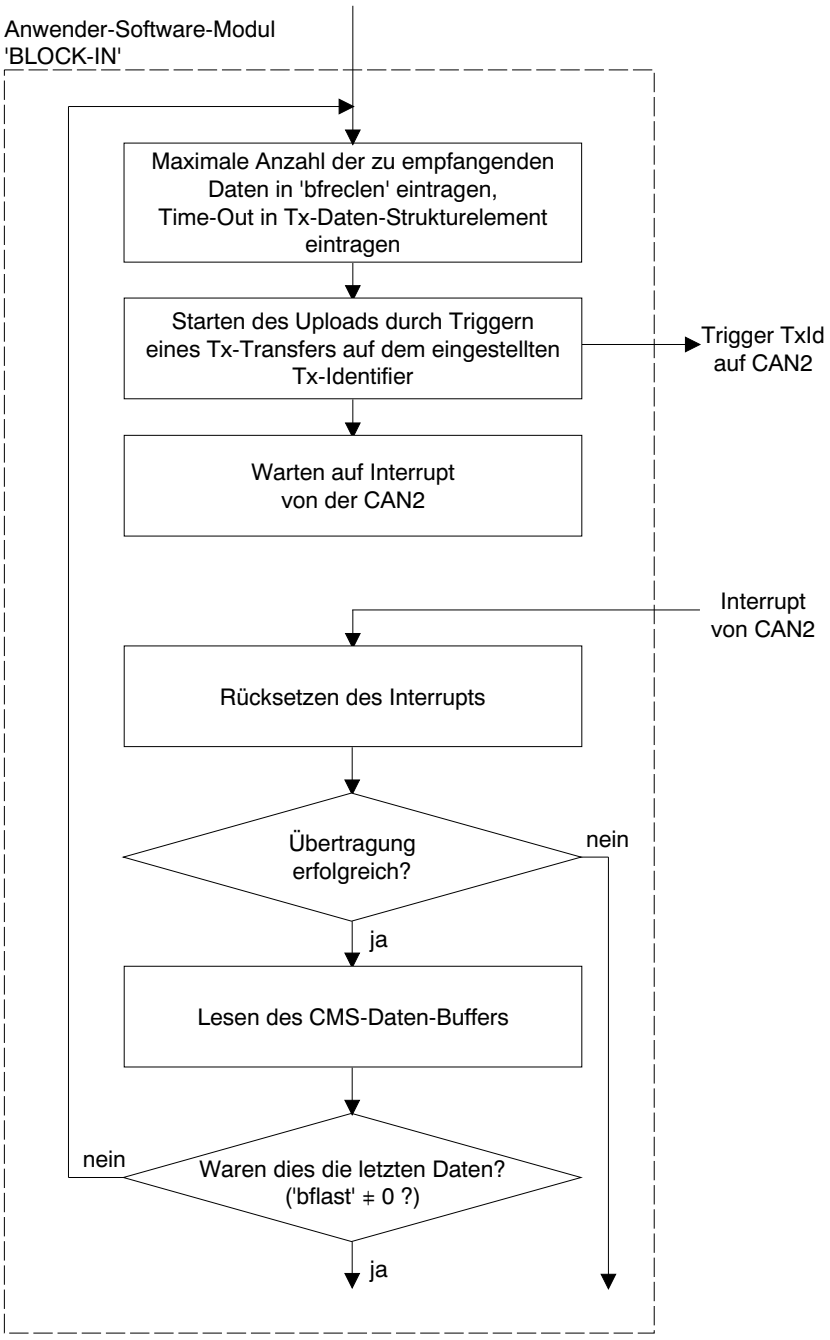


Abb. 3.5.9: Anwender-Software-Modul 'BLOCK-IN'

In der folgenden Abbildung ist der Ablauf der lokalen Firmware für einen Client Upload dargestellt. Die Firmware sendet selbständig die notwendige Init-Sequenz an den angeschlossenen Server und steuert die Übertragung der Daten.

Wird ein aktives Ende-Bit empfangen, so setzt die lokale Firmware die Zelle 'bflast' auf '-1' um dem VMEbus-Master das Ende der Übertragung anzuzeigen. Ebenfalls nach dem Empfangen des Ende-Bits setzt die Firmware die Zelle 'bfcmmd', die vom Anwender auf '2' gesetzt worden war, auf '0' zurück. Die Zelle 'bfcmmd' wird auch zurückgesetzt, wenn die Übertragung aufgrund eines Fehlers abgebrochen wurde.

Das Ende der Übertragung wird dem Anwender über einen Interrupt mitgeteilt.

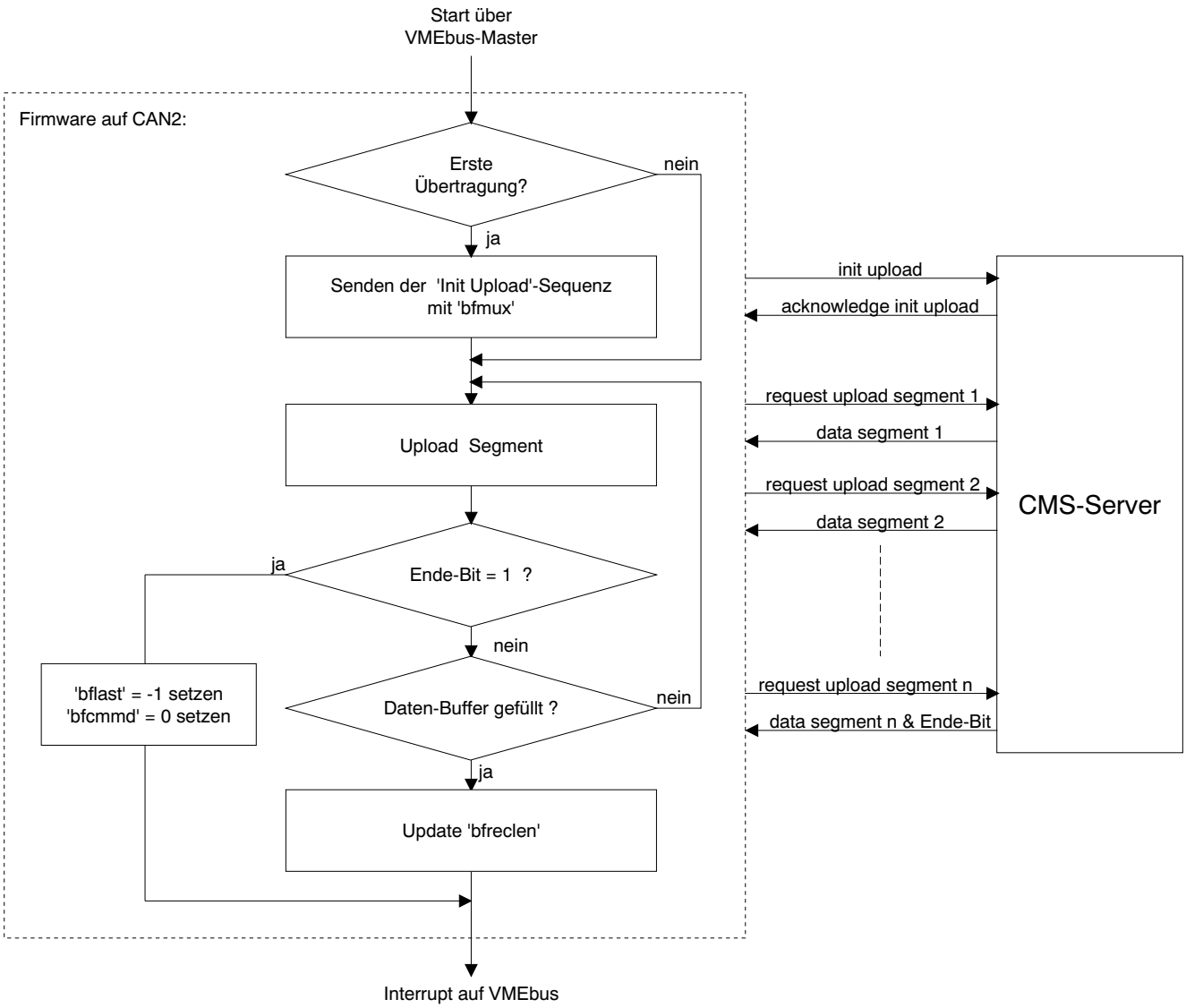


Abb. 3.5.10: Ablauf der lokalen CAN2-Firmware für Client Upload-Transfers

3.5.1.2.5 Server Download

Die CAN2 ist in der Lage, Downloads auch als CMS-Server zu bedienen. In diesem Fall wird der Transfer vom einem anderen CAN-Bus-Partner (Client) gestartet und gesteuert.

Wie bei den bereits beschriebenen Client-Übertragungen wird auf der CAN2 zuerst ein 'Init Domain' ausgeführt und die CMS-Buffer-Adresse ermittelt (siehe vorangegangenes Kapitel 'Initialisierung des CMS-Transfer-Buffers').

Das Setzen der Zelle 'bfcmmd' im CMS-Buffer auf '3' wählt den Server-Mode aus. Aktiviert wird der Mode durch Triggern eines Tx-Transfers. Die CAN2 befindet sich jetzt im Server-Status 'idle'. Der aktuelle Server-Status kann in der Zelle 'bfsvst' des Buffers gelesen werden.
Der Client kann jetzt einen Upload oder einen Download von der CAN2 anfordern. Die CAN2 wird daraufhin einen VMEbus-Interrupt auslösen.

Die folgende Abbildung zeigt vereinfacht den Programmablauf des VMEbus-Masters für die Initialisierung der CAN2 und die Abfrage der Zustände Upload/Download. Nach dem Triggern des Tx-Transfers befindet sich der VMEbus-Master in einer Time-Out-Schleife. Die Schleife kann nur durch einen Interrupt der CAN2 verlassen werden.

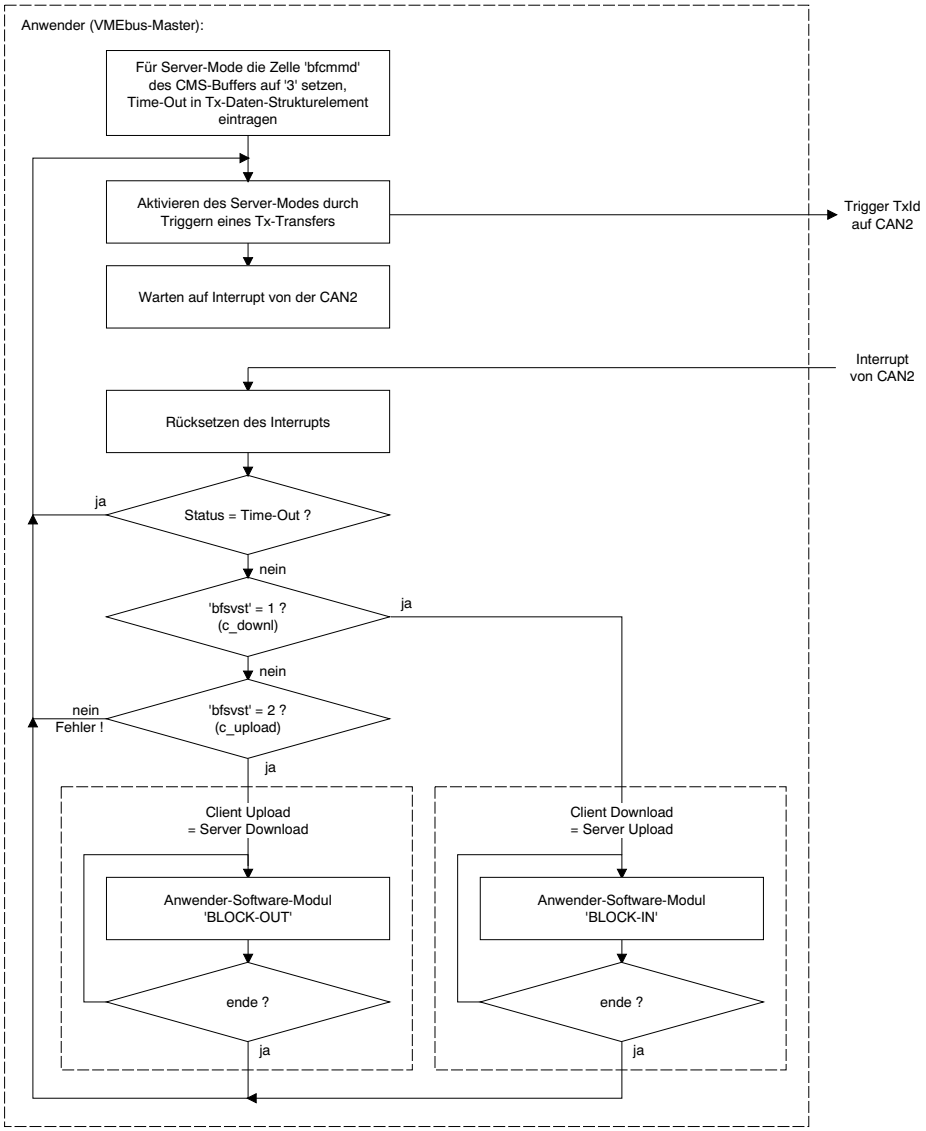


Abb. 3.5.11: Empfohlener Aufbau für die Abfrage der Upload/Download-Anforderung

Das Zusammenspiel von VMEbus-Master, CAN2 und CMS-Client ist in der folgenden Abbildung dargestellt. Voraussetzung für den Transfer ist, wie bei den bisher beschriebenen Betriebsarten, das Vorhandensein eines initialisierten CMS-Buffers.

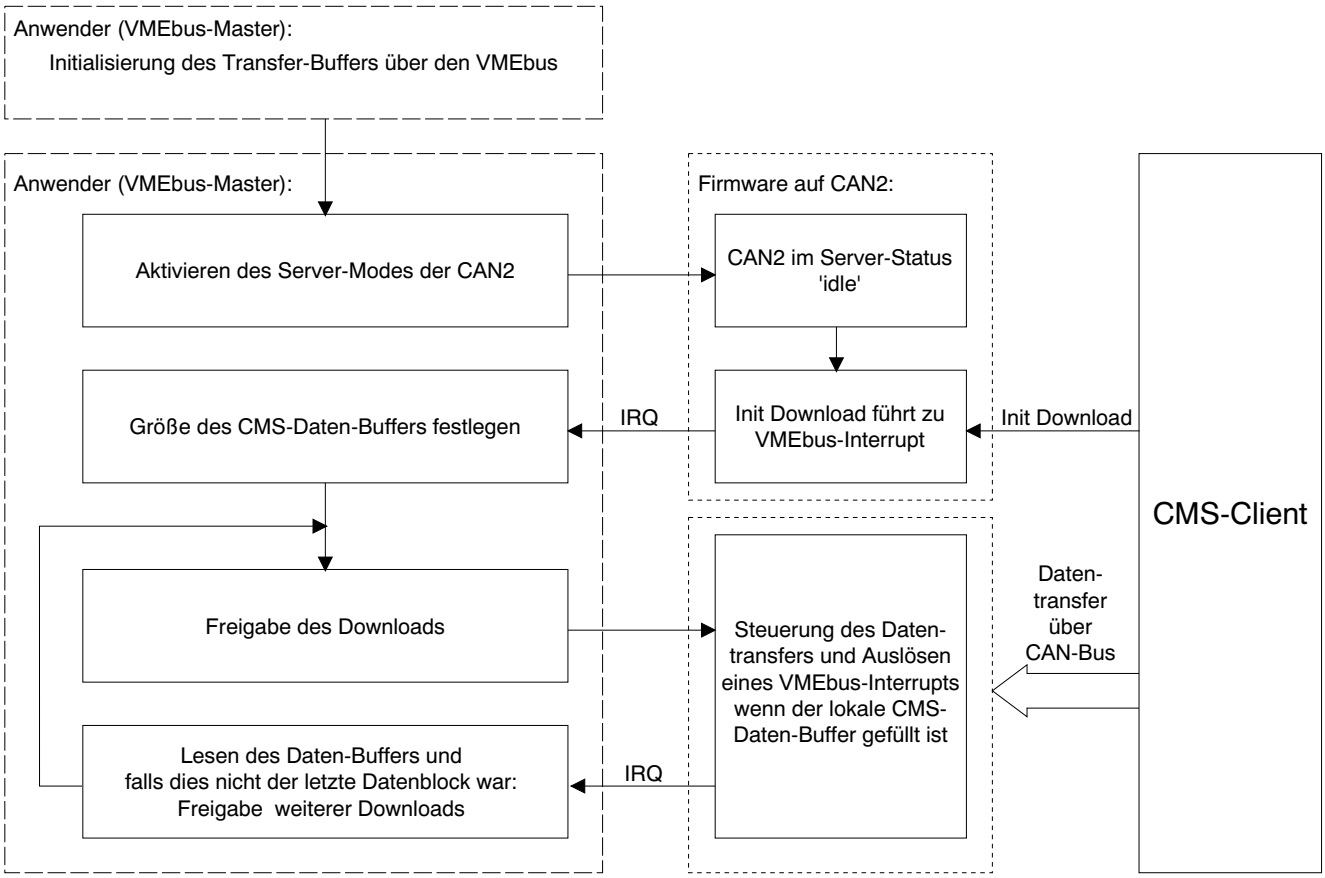


Abb. 3.5.12: Übersicht des Server Downloads

Nach dem Start des Server-Modes befindet sich die CAN2 im Server-Status 'idle'. Empfängt die CAN2 einen 'Init Download' des CMS-Clients löst sie einen VMEbus-Interrupt aus. Der VMEbus-Master muß jetzt die maximale Anzahl der zu lesenden Daten im Buffers der CAN2 festlegen und den Download durch einen Tx-Transfer freigeben.
Die weitere Steuerung des Downloads erfolgt durch den CMS-Client. Ist der Daten-Buffer auf der CAN2 gefüllt, so löst diese wiederum ein Interrupt aus. Der VMEbus-Master muß den Buffer auslesen und kann bei Bedarf weiter Downloads freigeben.

Im folgenden werden die einzelnen Blöcke der vorangegangenen Abbildung näher beschrieben:

Die 'Initialisierung des Transfer-Buffers' wurde bereits in einem vorangegangenen Kapitel erläutert.

Die im folgenden dargestellte Abbildung zeigt beispielhaft den Programmablauf für den VMEbus-Master während eines Server Downloads. Dieser Programmteil ist nicht im Lieferumfang der CAN2 enthalten und muß daher vom Anwender selbst realisiert werden.
Nachdem die Zelle 'bfcmmd' gesetzt ist, wird der Server-Mode durch Triggern eines Tx-Transfers gestartet. Die CAN2 wartet jetzt auf eine Init Download- oder Init Upload-Sequenz vom CMS-Client. In diesem Fall sei ein Init Download angenommen.

Wie bereits besprochen, löst die CAN2 einen VMEbus-Interrupt aus, der vom VMEbus-Master bedient werden muß. Der Master liest die Zelle 'bfsvst', um festzustellen, ob es sich um Upload oder Download handelt und ruft entsprechende Routinen auf. Für einen Server Download ist dies das bereits beim Client Upload eingesetzte Modul 'BLOCK-IN'. Das Modul ist in dem Kapitel Client Upload bereits ausführlich beschrieben worden.

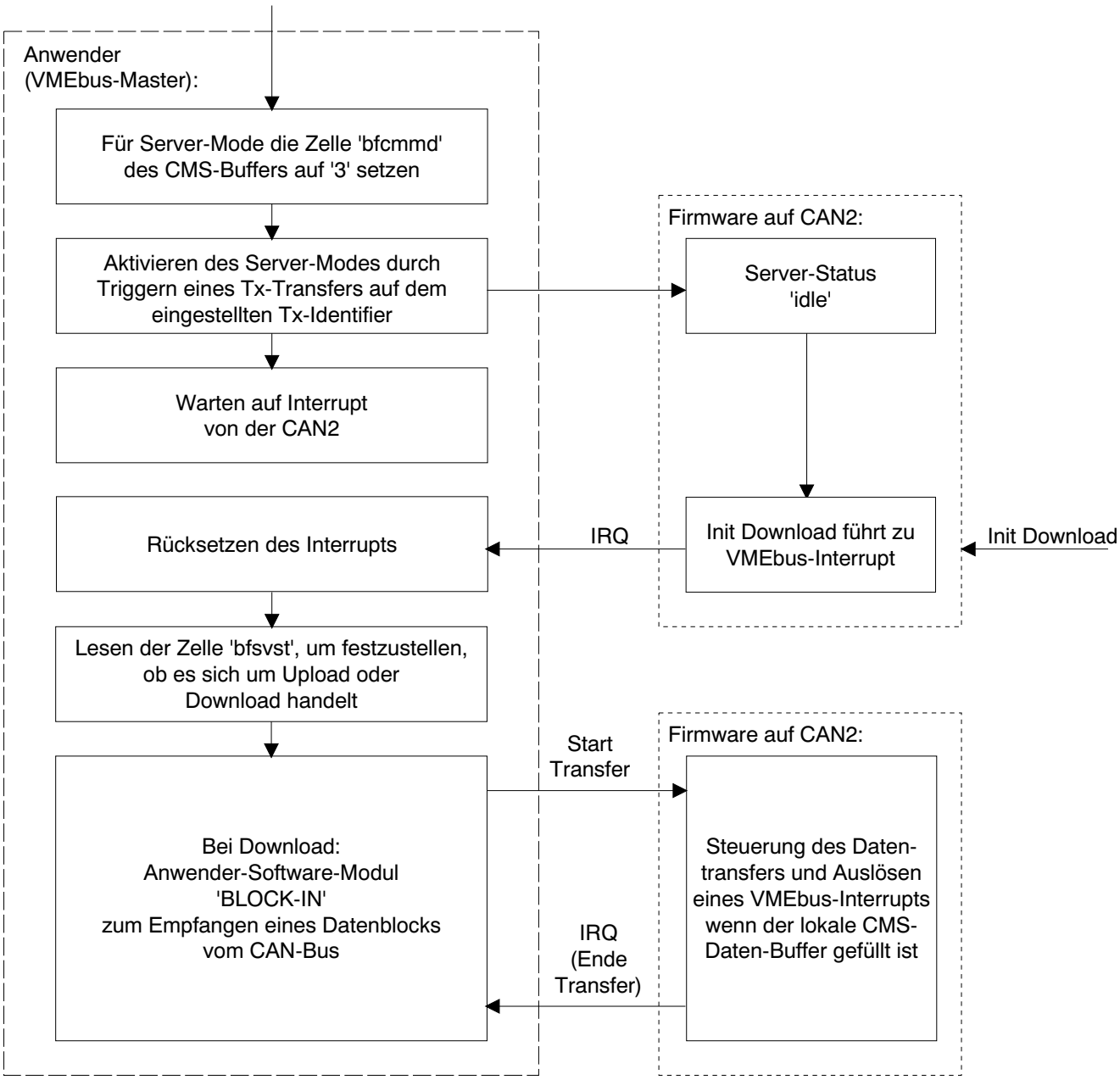


Abb. 3.5.13: VMEbus-Master beim Server Download

In der folgenden Abbildung ist der Ablauf der lokalen Firmware für einen Server Download dargestellt.

Die im ersten Block dargestellte Sequenz wird für jeden Server Download nur einmal durchlaufen.

Nach der Freigabe durch den VMEbus startet die Firmware die Download-Schleife. Sind nach dem Füllen des Daten-Buffers noch nicht alle Daten übertragen, so kann die Download-Schleife vom VMEbus erneut gestartet werden. Das Anstoßen der lokalen Firmware vom VMEbus-Master erfolgt jeweils durch Triggern eines Tx-Transfers auf dem TxId, der für die CMS-Transfers gewählt worden ist.

Wird ein aktives Ende-Bit empfangen, so setzt die lokale Firmware die Zelle 'bflast' auf '-1' um dem VMEbus-Master das Ende der Übertragung anzuzeigen. Ebenfalls nach dem Empfangen des Ende-Bits setzt die Firmware die Zelle 'bfsvst' auf '0' zurück und geht in den Zustand 'idle'. Die Zelle 'bfsvst' wird auch zurückgesetzt, wenn die Übertragung aufgrund eines Fehlers abgebrochen wurde.

Das Ende der Übertragung wird dem Anwender über einen Interrupt mitgeteilt.

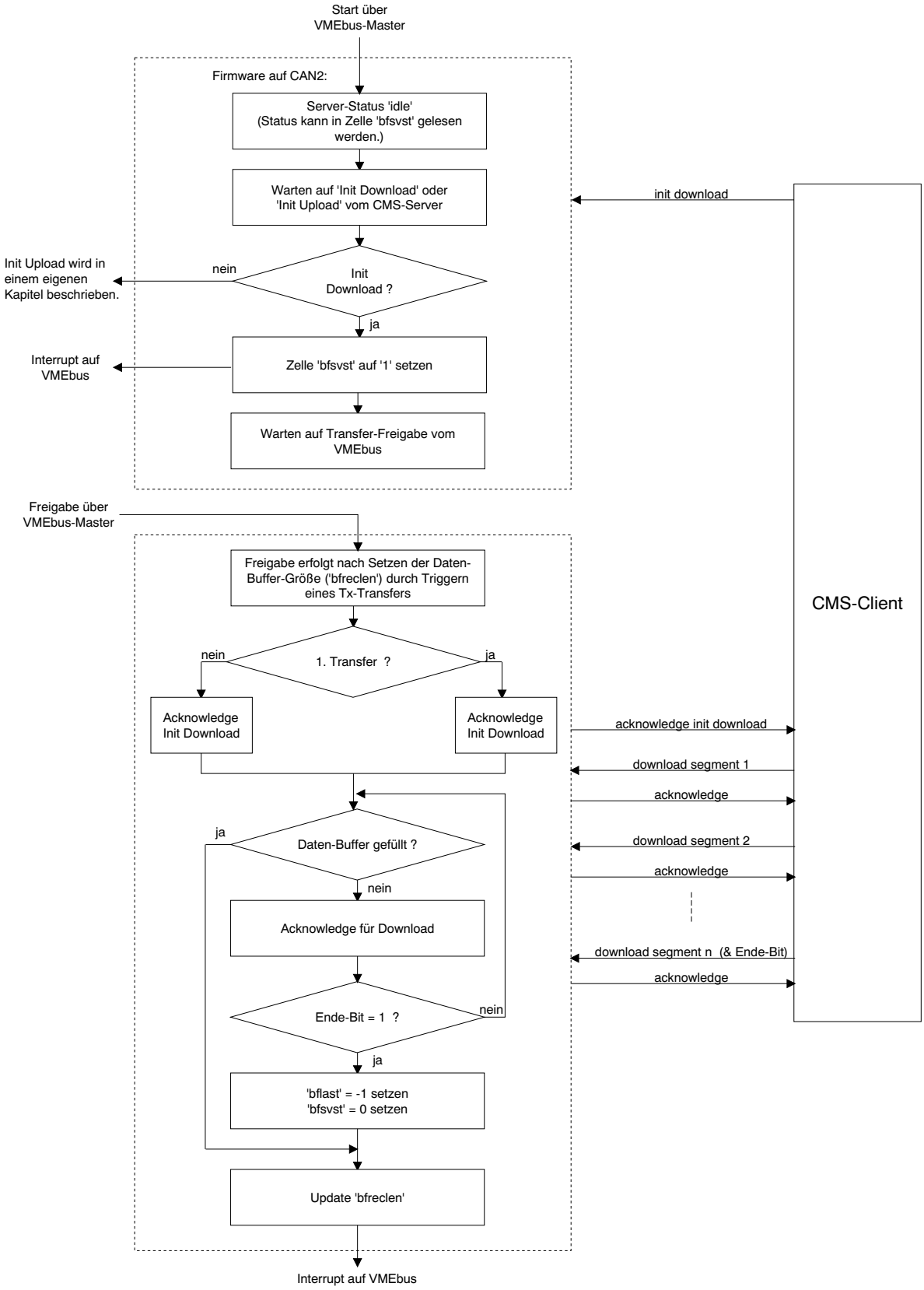


Abb. 3.5.14: Ablauf der lokalen CAN2-Firmware für Server Download-Transfers

3.5.1.2.6 Server Upload

Die CAN2 ist in der Lage, Uploads auch als CMS-Server zu bedienen. In diesem Fall wird der Transfer vom einem anderen CAN-Bus-Partner (Client) gestartet und gesteuert.

Wie bei den bereits beschriebenen Client-Übertragungen wird auf der CAN2 zuerst ein 'Init Domain' ausgeführt und die CMS-Buffer-Adresse ermittelt (siehe vorangegangenes Kapitel 'Initialisierung des CMS-Transfer-Buffers').

Das Setzen der Zelle 'bfcmmd' im CMS-Buffer auf '3' wählt den Server-Mode aus. Aktiviert wird der Mode durch Triggern eines Tx-Transfers. Die CAN2 befindet sich jetzt im Server-Status 'idle'. Der aktuelle Server-Status kann in der Zelle 'bfsvst' des Buffers gelesen werden.
Der Client kann jetzt einen Upload oder einen Download von der CAN2 anfordern. Die CAN2 wird daraufhin einen VMEbus-Interrupt auslösen.

In dem vorangegangenen Kapitel wurde bereits ein vereinfachter Programmablauf des VMEbus-Masters speziell für die Initialisierung der CAN2 und die Abfrage der Zustände Upload/Download dargestellt.

Das Zusammenspiel von VMEbus-Master, CAN2 und CMS-Client ist in der folgenden Abbildung dargestellt. Voraussetzung für den Transfer ist, wie bei den bisher beschriebenen Betriebsarten, das Vorhandensein eines initialisierten CMS-Buffers.

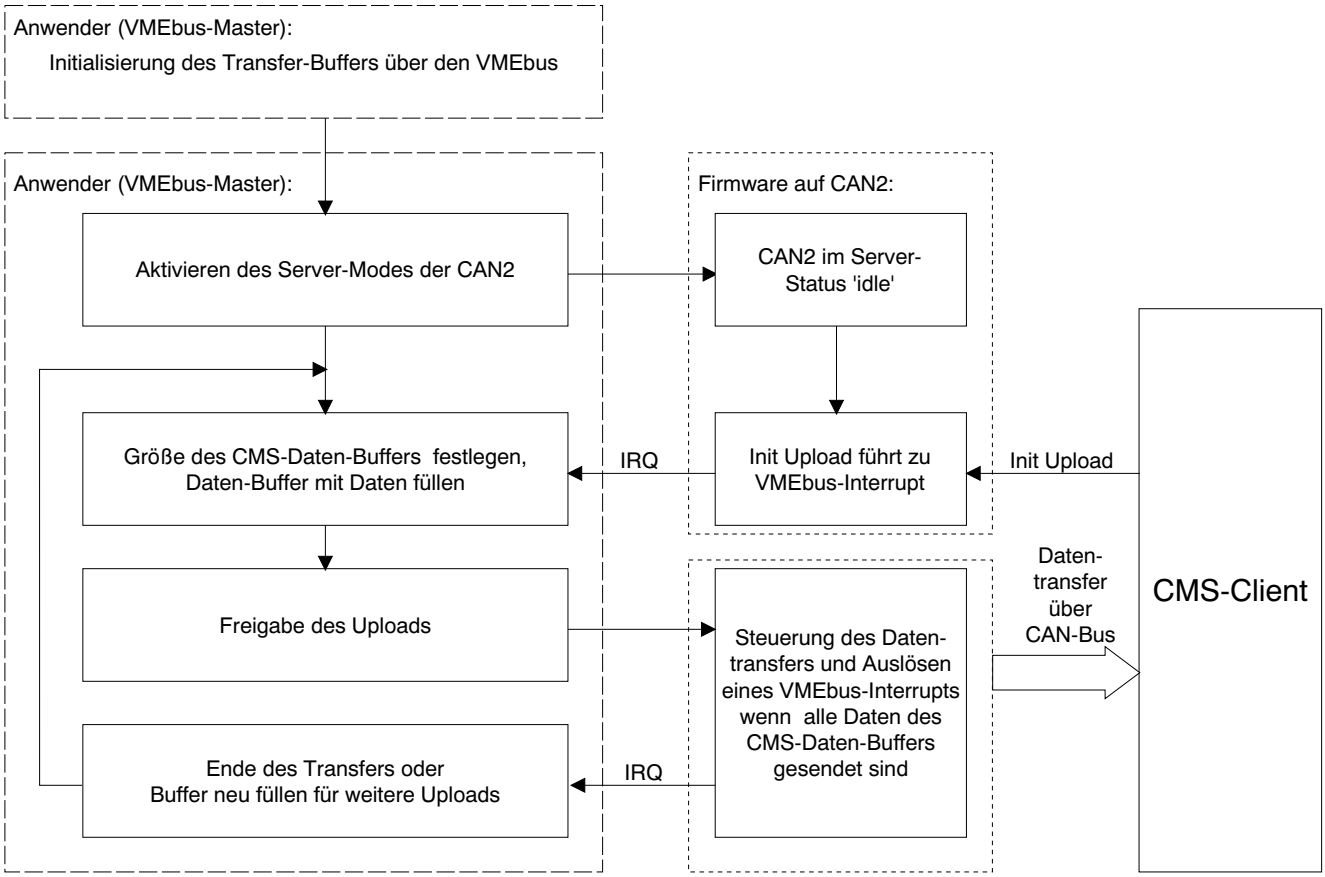


Abb. 3.5.15: Übersicht des Server Uploads

Nach dem Start des Server-Modes befindet sich die CAN2 im Server-Status 'idle'. Empfängt die CAN2 einen 'Init Upload' des CMS-Clients, so löst sie einen VMEbus-Interrupt aus. Der VMEbus-Master muß jetzt die Größe des Daten-Buffers der CAN2 festlegen, den Buffer mit Daten füllen und den Upload durch einen Tx-Transfer freigeben.

Die weitere Steuerung des Uploads erfolgt durch den CMS-Client. Ist der Transfer beendet, so löst die CAN2 ein Interrupt aus. Der VMEbus-Master kann den Erfolg der Übertragung prüfen und gegebenenfalls einen neuen Transfer bedienen.

Im folgenden werden die einzelnen Blöcke der vorangegangenen Abbildung näher beschrieben:

Die 'Initialisierung des Transfer-Buffers' wurde bereits in einem vorangegangenen Kapitel erläutert.

Das folgende Bild zeigt beispielhaft den Programmablauf für den VMEbus-Master während eines Server Uploads. Dieser Programmteil ist nicht im Lieferumfang der CAN2 enthalten und muß daher vom Anwender selbst realisiert werden.

Nachdem die Zelle 'bfcmmd' gesetzt ist, wird der Server-Mode durch Triggern eines Tx-Transfers gestartet. Die CAN2 wartet jetzt auf eine Init Download- oder Init Upload-Sequenz vom CMS-Client. In diesem Fall sei ein Init Upload angenommen.

Wie bereits besprochen, löst die CAN2 einen VMEbus-Interrupt aus, der vom VMEbus-Master bedient werden muß. Der Master liest die Zelle 'bfsvst', um festzustellen, ob es sich um Upload oder Download handelt und ruft entsprechende Routinen auf. Für einen Server Upload ist dies das bereits beim Client Download eingesetzte Modul 'BLOCK-OUT'. Das Modul ist in dem Kapitel Client Download bereits ausführlich beschrieben worden.

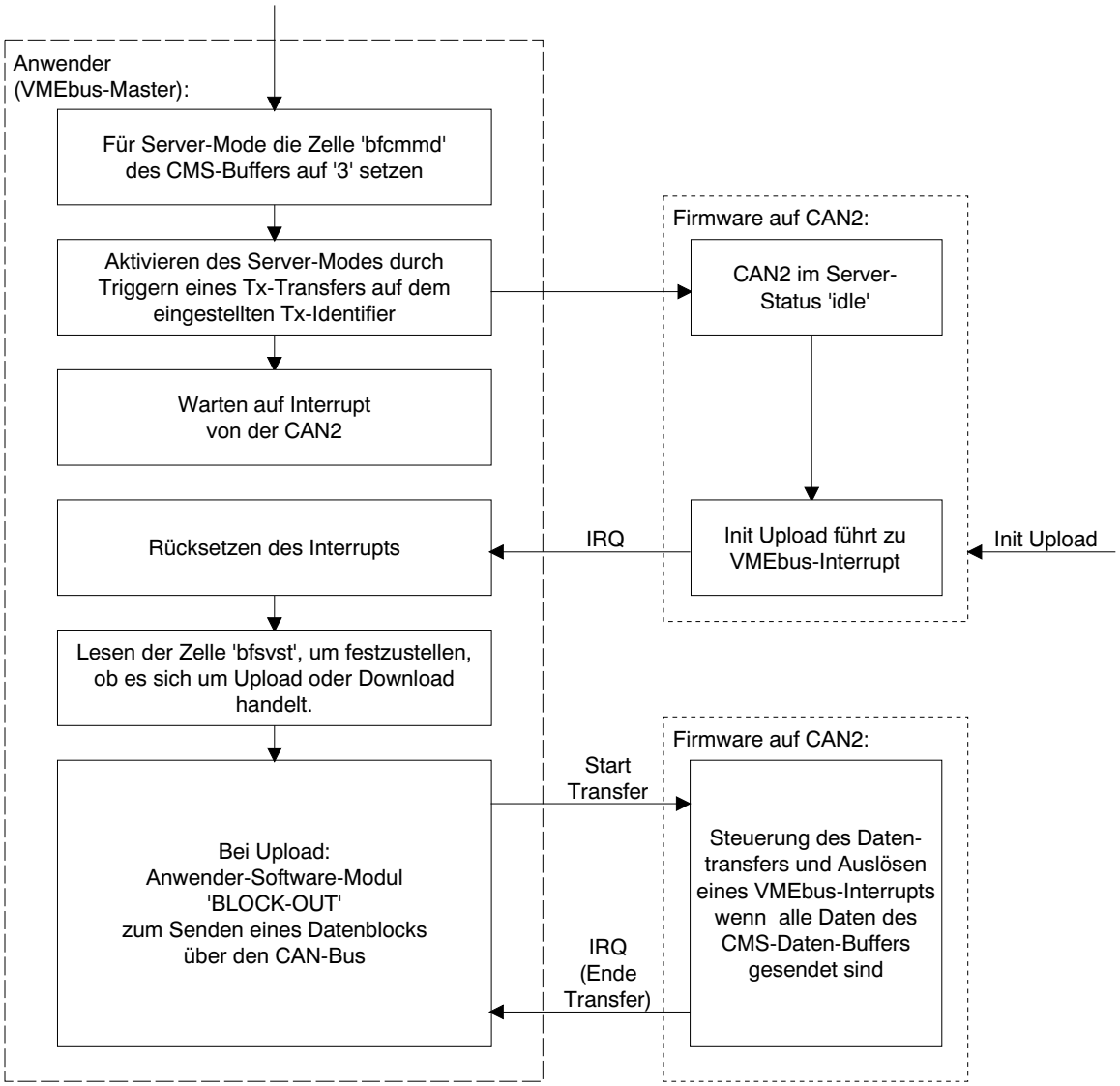


Abb. 3.5.16: VMEbus-Master beim Server Upload

In der folgenden Abbildung ist der Ablauf der lokalen Firmware für einen Server Upload dargestellt:

Die im ersten Block dargestellte Sequenz wird für jeden Server Upload nur einmal durchlaufen.

Nach der Freigabe durch den VMEbus startet die Firmware die Upload-Schleife. Während des Uploads zählt die lokale Firmware die empfangenen Daten mit und löst einen VMEbus-Interrupt aus, wenn der Buffer gefüllt ist. Der VMEbus-Master kann nach diesem Interrupt den Buffer auslesen und dann den Upload wieder starten.

Das Anstoßen der lokalen Firmware vom VMEbus-Master erfolgt jeweils durch Triggern eines Tx-Transfers auf dem TxId, der für die CMS-Transfers gewählt worden ist.

Vor dem Schreiben des letzten Datenblocks in den CMS-Buffer muß die Zelle 'bflast' auf einen Wert ungleich Null gesetzt werden. Daraufhin setzt die lokale Firmware bei der Sendung des letzten Datensegmentes diese Blocks das Ende-Bit, um dem CMS-Server das Ende der Übertragung anzuzeigen. Außerdem wird nach der letzten Übertragung die Zelle 'bfsvst' auf '0' zurückgesetzt.

Die Zelle 'bfsvst' wird auch zurückgesetzt, wenn die Übertragung aufgrund eines Fehlers abgebrochen wurde.
Das Ende der Übertragung wird dem Anwender über einen Interrupt mitgeteilt.

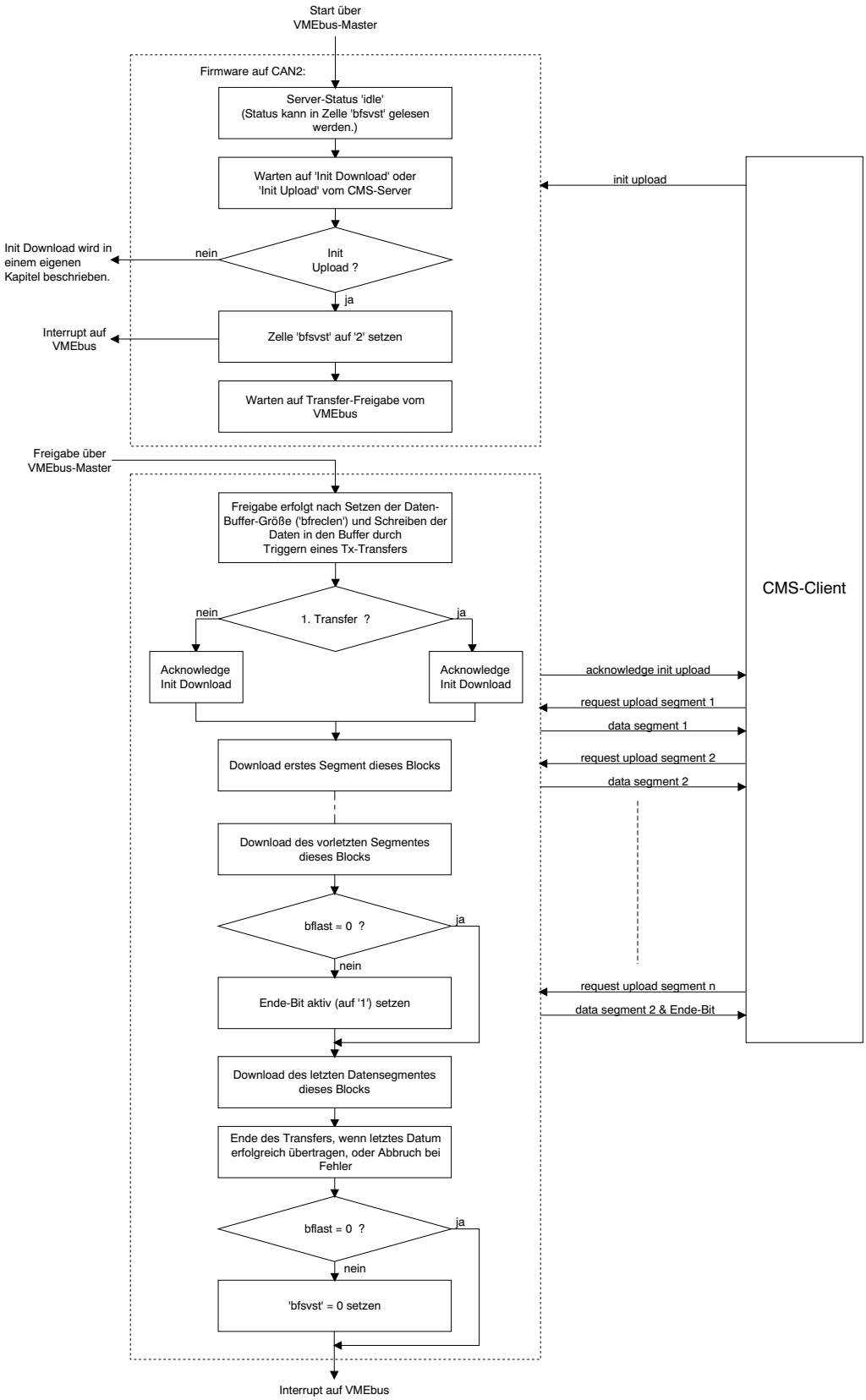


Abb. 3.5.17: Ablauf der lokalen CAN2-Firmware für Server Upload-Transfers

3.5.2 CMS-Watchdog

Der Watchdog Tx-Identifier und das gewünschte CAN-Netz der CAN2 werden über das Kommando 'Activate periodical Tx-Transfers' des Parameter-Buffers ausgewählt. Die Eingabe wird in dem folgenden Kapitel 'Der Parameter-Buffer' weiter beschrieben. Es dürfen nur positive Werte eingegeben werden.

Wesentliche Parameter für den CMS-Watchdog sind im Daten-Strukturelement des Watchdog-Tx-Identifiers abgelegt:

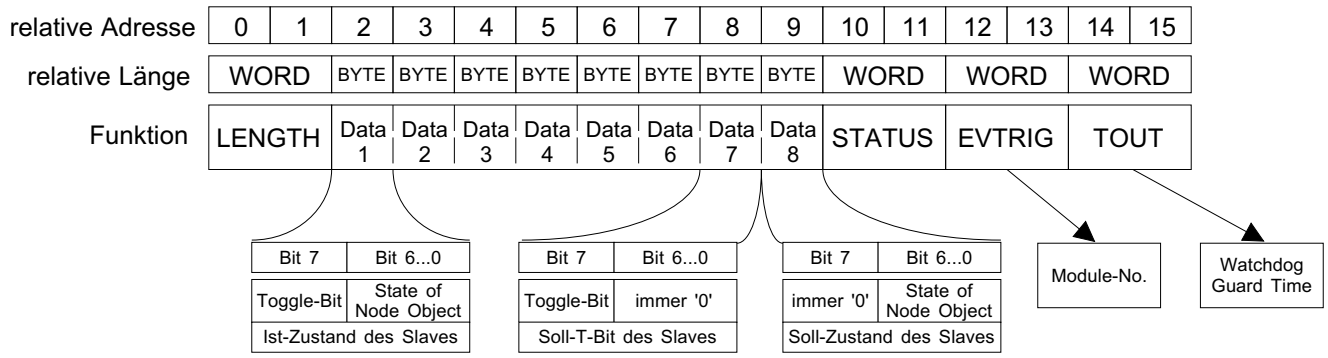


Abb. 3.5.1: Belegung des Datenstrukturelementes des Watchdog Tx-Identifiers

Der Ablauf der Watchdog-Funktion kann der folgenden Abbildung entnommen werden.

Nach dem Start des Watchdog durch den VMEbus-Master erfolgt die Steuerung des Watchdog durch die lokale Firmware. Die CAN2 sendet RTR-Anforderungen auf dem Watchdog-TxId und wertet die Rückmeldungen des angesprochenen Moduls aus.

Tritt ein Fehler auf, so wird in das FIFO 'Data To VME' ein Pointer geschrieben und dadurch, sofern freigegeben, ein VMEbus-Interrupt ausgelöst. Der Interrupt muß vom VMEbus-Master bedient werden. Die Auswertung der Fehlermeldung bleibt dem Anwender überlassen.

Im Fehlerfall wird der Watchdog lokal disabled und muß explizit wieder freigegeben werden.

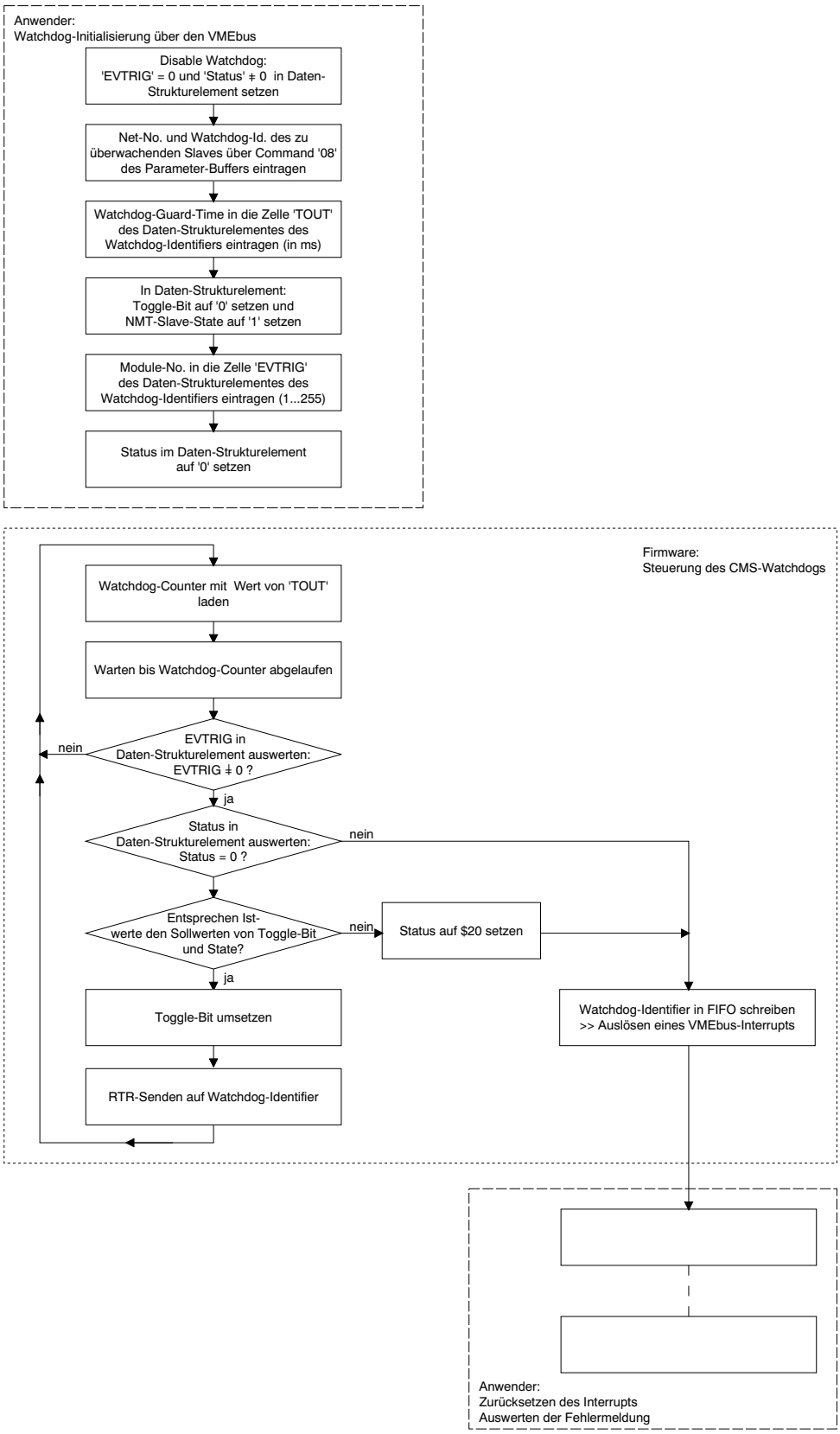


Abb. 3.5.2: Ablauf der Watchdog-Steuerung auf der CAN2

4. Der Parameter-Buffer

4.1 Funktion und Aufbau

Über den Parameter-Buffer werden verschiedene Kommandos, die Parameter der Monitor-Buffer und diverse Parameter der physikalischen CAN-Schnittstelle übergeben.

In der folgenden Abbildung ist die Struktur des Parameter-Buffers mit den für den Anwender relevanten Parametern dargestellt.

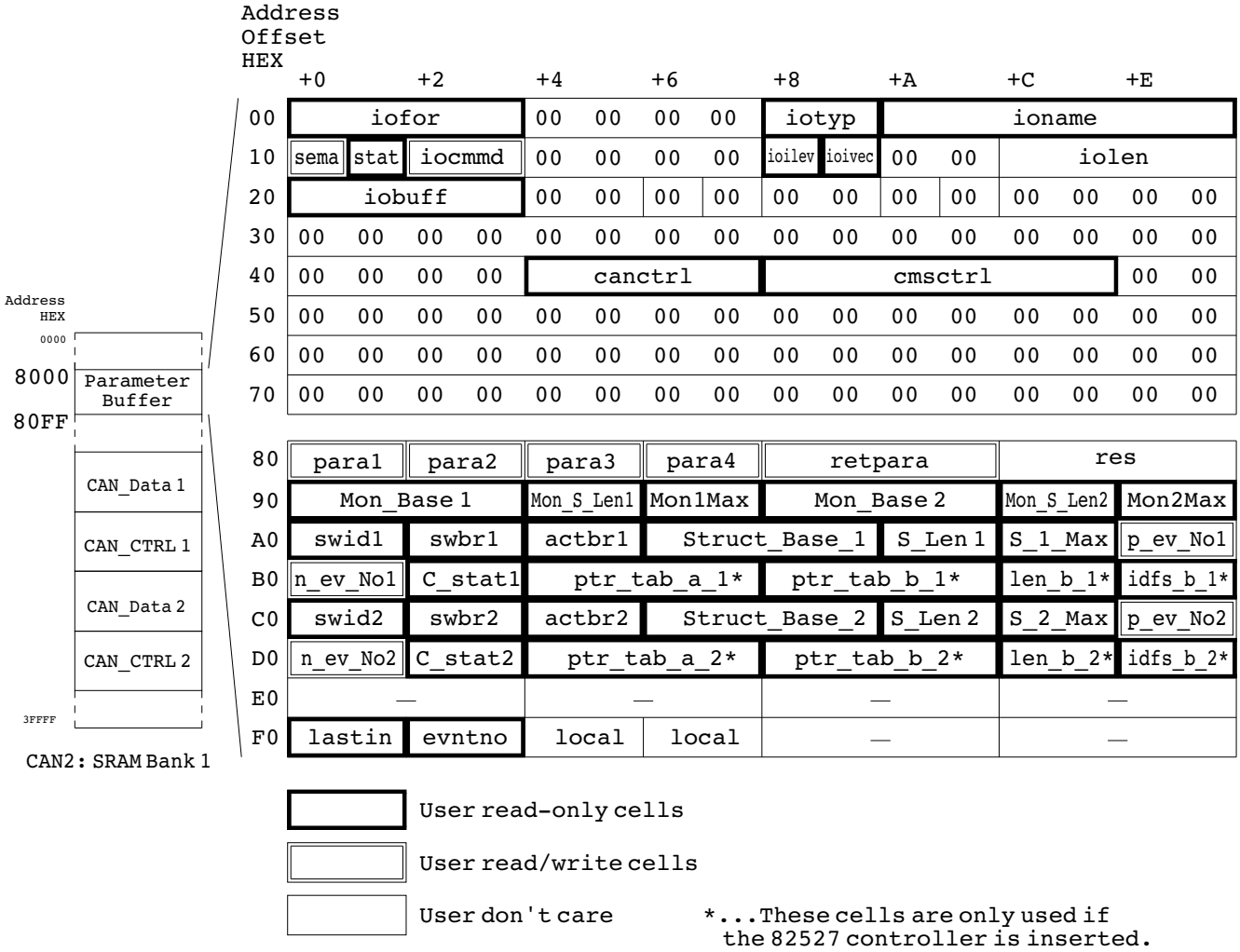


Abb. 4.1.1: Aufteilung des Parameter-Buffers

Die Abbildung auf der folgenden Seite zeigt den Parameter-Buffer mit seinen Default-Einstellungen nach einem RESET der CAN2.

Address Offset HEX	+0		+2		+4		+6		+8		+A		+C		+E	
00	00	00	**80**	00	00	00	00	00	00	**0C**	'CANP' z.B. '5' '8'					
10	sema	stat	**FF**	**FF**	00	00	00	00	00	00	00	00	00	00	00	**80**
20	00	00	**80**	**80**	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	'C200'or'C527'				'___CMS'or'_NoCMS'						00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00	00	00	00	00	res			
90	00	**03**	00	**50**	00	**10**	**10**	00	00	**04**	00	**50**	00	**10**	**10**	00
A0	swid1		swbr1		actbr1		00	**01**	00	00	00	**10**	**08**	00	00	00
B0	00	00	00	**0C**	00	00	00	00	00	00	00	00	00	00	00	00
C0	swid2		swbr2		actbr2		00	**02**	00	00	00	**10**	**08**	00	00	00
D0	00	00	00	**0C**	00	00	00	00	00	00	00	00	00	00	00	00
E0	—				—				—				—			
F0	00	00	00	00	00	00	00	00	—				—			

User read-only cells

User read/write cells

User don't care

Abb. 4.1.2: Besetzung des Parameter-Buffers

4.2 Übergabe von Parametern und Kommandos

Das Setzen von Parametern und Kommandos einer VMEbus MASTER-CPU zur CAN2 muß in folgender Reihenfolge ablaufen:

1. - Auswerten der Zelle 'sema'--> Parameter-Buffer frei?
   - wenn frei, dann Belegen des Parameter-Buffers durch Setzen der Zelle 'sema' ('TAS'-Befehl)

2. - Setzen der Parameter und Kommandos durch Schreiben in die gewünschten Zellen.

3. - Übergabe der Parameter und Kommandos an den CAN2-Slave-Server durch Triggern des lokalen 'GIRL'-Interrupts (siehe auch Kap.'Firmware-definierte Interrupts').

Der GIRL-Interrupt wird durch das Schreiben des Datums $0000 auf die relative I/O-Adresse $00002 mit einem Wort-Zugriff ausgelöst. Die Adreßbelegung des I/O-Bereichs und eine Beschreibung aller GIRL-Interrupts ist in dem Hardware-Handbuch der CAN2 zu finden.

3. - Warten auf Ende der Kommandoabarbeitung: 'Cmmd' = FFFF

5. - Auswerten der Zelle 'stat':
   00 --> OK
   FF --> ERROR

6. - Freigeben der Zelle 'sema'.

4.3 Beschreibung der Parameter

Die Parameter werden unterschieden in 'nur lesbare' und 'schreib- und lesbare' Parameter. In der vorangegangenen Abbildung 4.1.1 sind diese verschiedenen Parametertypen durch unterschiedliche Umrandungen gekennzeichnet.

Neben der Parametrierung der CAN2 können über den Parameter-Buffer diverse Kommandos übergeben werden. Die Kommandoübergabe wird im Kap. 'Kommandoübergabe über den Parameterkanal' beschrieben.

Die Beschreibung der Parameter erfolgt im weiteren in folgender Form:

'parameter'..... (relative lokale Adresse, Länge des Parameters) Erläuterungen zum Parameter

4.3.1 Schreib- und Lesbare Parameter

'sema'..... ($10, Byte) ist mit der Semaphore belegt:

Bit 7 - Semaphore : '0' -> Kanal ist frei
'1' -> Kanal ist belegt
Bit 6..0 - reserviert

'iocmmd'..... ($12, Wort) wird nur bei der Kommandoübergabe benötigt (siehe folgendes Kapitel 'Kommandos des Parameter-Buffers').
Default-Wert nach RESET: $FFFF (= 'letztes Kommando abgearbeitet')

'para1'..... ($80, Wort)
bis
'para4'..... ($86, Wort) sind die für die Kommandoübergabe benötigten Parameter
(siehe folgendes Kapitel 'Kommandos des Parameter-Buffers').
Default-Wert nach RESET: $0000

'retpara'..... ($88, Langwort) enthält Rückgabeparameter nach der Ausführung von Kommandos. Die Rückgabeparameter sind abhängig vom gewählten Kommando.
Ausnahme: Bei der Implementierung der Software für den 82527 wird retpara auch zum Setzen von Parametern durch den Anwender verwendet.

'p ev No1'..... ($AE, Wort),
'p ev No2'..... ($CE, Wort) enthält Information über die Auswertung eines Flankenwechsels des 'NOT-EIN'-Signals (NOT AUS-Eingänge):
Wenn der Inhalt dieser Zellen ungleich '0' ist, wird bei der 'L' ->'H' -Flanke des 'NOT-EIN'-Signals ein VMEbus-Interrupt ausgelöst und die Ev-No. (EVENT-Nummer) in der Zelle 'evntno' abgelegt.

'n ev No1'..... ($B0, Wort),
'n ev No2'..... ($D0, Wort) enthält Information über die Auswertung eines Flankenwechsels des 'NOT-EIN'-Signals (NOT AUS-Eingänge):
Wenn der Inhalt dieser Zellen ungleich '0' ist, wird bei der 'H' ->'L' -Flanke des 'NOT-EIN'-Signals ein VMEbus-Interrupt ausgelöst und die Ev-No. (EVENT-Nummer) in der Zelle 'evntno' abgelegt.

4.3.2 'Nur lesbare' Parameter

'iofor'..... ($00, Langwort) Pointer auf den nächsten gleichartigen Buffer (hier nur ein Buffer, daher 'iofor'=Parameter-Buffer-Adresse [$8000])

'iotyp'..... ($08, Wort) Buffer-Identifier (immer $000C)

'ioname'..... ($0A, 6 Byte ASCII) enthält den Buffer-Identifier als ASCII-String und eine fortlaufende Numerierung (immer 'CANP ').

'stat'..... ($11, Byte) enthält Information über die korrekte Übergabe eines Kommandos (siehe auch folgendes Kapitel 'Kommandos des Parameter-Buffers')

'stat' = $00 --> kein Fehler
'stat' ≠ $00 --> Fehler

Die Kommandoübergabe sollte prinzipiell in folgender Reihenfolge ablaufen:
1. 'sema' belegen
2. Parameter und Kommandos übergeben
3. Trigger IRQ (GIRL-Interrupt)
3. Warten auf 'Cmmd' = FFFF
5. 'stat' abfragen:
   = 00 --> OK
   ≠ 00 --> ERROR
6. 'sema' freigeben

'ioilev'..... ($18, Byte) enthält den über das Kommando 'Card-Interrupt-Enable' gesetzen Wert von 'iolev' (para1)

'ioivec'..... ($19, Byte) enthält den über das Kommando 'Card-Interrupt-Enable' gesetzen Wert von 'iovec' (para2)

'iolen'..... ($1C, Langwort) gibt die Länge des Datenbereiches des Parameter-Buffers an.
Default: 'iolen' = $80 == 128 Byte.

'iobuff'..... ($20, Langwort) ist der Pointer auf den dem Buffer zugeordneten Datenbereich. Als Default zeigt 'iobuff' auf die Zelle 'para 1'.

'canctrl'..... ($44, 4 Bytes) gibt als ASCII-Code den Typ des eingesetzten CAN-Controllers zurück:
'C200' -> 82C200
'C527' -> 82527

'cmsctrl'..... ($48, 6 Bytes) gibt als ASCII-Code an, ob das CMS-Protokoll auf der Karte implementiert ist:
'   CMS' -> CMS-Protokoll implementiert
' NoCMS' -> CMS-Protokoll nicht implementiert

'Mon Base1'.... ($90, Langwort),
'Mon Base2'.... ($98, Langwort) enthält die lokale Adresse der Monitor-Buffer.
Default: 'Mon Base1' - $30050
         'Mon Base2' - $40050

'Mon S Len1'.... ($94, Wort),
'Mon S Len2'.... ($9C, Wort) enthält die Länge eines Monitor-Strukturelementes.
Default: 'Mon S Len1' - $10 (16 Byte)
         'Mon S Len2' - $10 (16 Byte)

'Mon1Max'..... ($96, Wort),
'Mon2Max'..... ($9C, Wort) enthält die maximale Anzahl von Strukturelementen im Monitor-Buffer.
Default: 'Mon1Max' - $1000 (DEZ 4096)
'Mon1Max' - $1000 (DEZ 4096)

'swid1'..... ($A0, Wort),
'swid2'..... ($C0, Wort) enthält nach einem RESET den über die Kodierschalter in der Frontplatte (SW1 und SW3) eingestellten Wert der CAN-Net-No.($0000...$000F).
physik. Kanal 1 - Kodierschalter SW1 - 'swid1'
physik. Kanal 2 - Kodierschalter SW3 - 'swid2'

Die Zuordnung der CAN-Net-No. zu den in der Hardware-Beschreibung benannten 'physikalischen' Kanälen 1 und 2 ist frei wählbar.

'swbr1'..... ($A2, Wort),
'swbr2'..... ($C2, Wort) enthält nach einem RESET die über die Kodierschalter auf der Platine (SW2 und SW4) eingestellten Default-Bitraten-indizes. Die Bitratenindizes entsprechen dem Inhalt der Register BTR0 und BTR1 des Controllers 82C200.

Kanal 1 - Kodierschalter SW2 - 'swbr1'
Kanal 2 - Kodierschalter SW4 - 'swbr2'

Die Bitratenindizes können durch eine Kommandoübergabe des Parameterkanals verändert werden (siehe Kap. 'Kommandoübergabe über den Parameterkanal'). Die aktuellen Bitratenindizes werden in den Zellen 'actbr1' und 'actbr2' abgelegt.

Die unten dargestellte Tabelle zeigt die Einstellung der Bitrate über die Kodierschalter SW2 und SW4 mit Angabe des Register-inhalts (Bitratenindex) des Controllers 82C200 und die jeweils physikalisch maximal mögliche Leitungslänge.

SW2, SW4 [HEX]	Bitrate [kBit/s]	82C200- oder 82527-Register		typische Werte der erreichbaren Leitungslänge l_{max} [m]	minimal erreichbare Leitungslänge l_{min} [m]
		BTR0 [HEX]	BTR1 [HEX]		
0	1000	00	14	37	20
1	666.6	00	18	80	65
2	500	00	1C	130	110
3	333.3	01	18	180	160
4	250	01	1C	270	250
5	166	02	1C	420	400
6	125	03	1C	570	550
7	100	04	1C	710	700
8	66.6	45	2F	1000	980
9	50	09	1C	1400	1400
A	33.3	4B	2F	2000	2000
B	20	18	1C	3600	3600
C	12.5	5F	2F	5400	5400
D	10	31	1C	7300	7300
E	800	00	16	59	42

Die Angaben der Tabelle basieren auf den Grenzwerten des Bit-Timings des CAN-Protokolls, den Laufzeiten des lokalen CAN-Interface und den Laufzeiten des Kabels. Die Laufzeit des Kabels ist mit ca. 5,5ns/m angenommen. In den Angaben sind weitere Einflüsse z.B. durch die Abschlußwiderstände, den spezifischen Widerstand, die Geometrie des Kabels oder äußere Störeinflüsse bei der Übertragung nicht miteinbezogen!

Tabelle 4.3.1: Einstellung der Bitrate

'actbr1'..... ($A4, Wort),
'actrb2'..... ($C4, Wort) enthält den aktuellen Bitratenindex ($0011...$7F7F). Der Bitratenindex kann über eine Kommandoübergabe verändert werden (siehe auch Kap. 'Kommandoübergabe über den Parameter-kanal')
Kanal 1 - 'actbr1'
Kanal 2 - 'actbr2'

'Struct Base1'.. ($A6, Langwort),
'Struct Base2'. ($C6, Langwort) enthält die lokale Basisadresse der CAN-Datenstruktur (CAN Data1, CAN Data2). Die Basisadresse der Kontrollstruktur (CAN CTRL1, CAN CTRL2) ist immer 'Struct Base'+ $8000.

'S Len 1'..... ($AA, Wort),
'S Len 2'..... ($CA, Wort) enthält die Länge eines Daten-Strukturelementes:
Default: 'S Len 1' = $0010 (16Byte)
'S Len 2' = $0010 (16Byte)

'S 1 Max'..... ($AC, Wort),
'S 2 Max'..... ($6C, Wort) enthält die Anzahl von Daten-Strukturelementen der Daten-Strukturfelder (CAN Data1, CAN Data2).
Default: 'S 1 Max' = $0800 (DEZ 2048)
'S 2 Max' = $0800 (DEZ 2048)

'C stat1'..... ($B2, Wort),
'C stat2'..... ($D2, Wort) enthält in Bit 0...7 den Status der CAN-Controller-Bausteine 82C200, bzw. 82527 gemäß dem entsprechenden Handbuch.

'ptr tab a 1*'.. ($B4, Langwort),
'ptr tab a 2*'.. ($D4, Langwort) Diese Zelle ist nur belegt, wenn der Controller 82527 eingesetzt wird.
Sie enthält den Pointer auf die Anfangsadresse der Zuordnungs-tabelle (Tabelle a) zwischen Handle und Identifier. Die Tabelle enthält 2048 Elemente (Handles). Jedem Handle wird ein Identifier und ein Identifier-Mode zugeordnet. Die Zuordnung erfolgt über ein Kommando des Parameter Buffers ('iocmmd' = $010).

'ptr tab b 1*'.. ($B8, Langwort),
'ptr tab b 2*'.. ($D8, Langwort) Diese Zelle ist nur belegt, wenn der Controller 82527 eingesetzt wird.
Sie enthält den Pointer auf eine zweite Tabelle (Tabelle b), in der die gewünschten Identifier von der Firmware so geordnet abgelegt werden, daß sie beim Eintreffen eines Rx-Frames möglichst schnell durchsucht werden können, um festzustellen, ob dieser Rx-Frame für die CAN2B bestimmt war.

'len b 1*'.. ($BC, Wort),
'len b 2*'.. ($DC, Wort) Diese Zelle ist nur belegt, wenn der Controller 82527 eingesetzt wird.
In dieser Zelle ist die Länge der oben beschriebenen Tabelle (Tabelle b) eingetragen. Die Länge errechnet sich folgendermaßen: Länge = 2^{n-1} , wobei das Maximum bei 2047 Einträgen liegt.

'idfs b 1*'.. ($BE, Wort),
'idfs b 2*'.. ($DE, Wort) Diese Zelle ist nur belegt, wenn der Controller 82527 eingesetzt wird.
Sie enthält die Anzahl der in der oben beschriebenen Tabelle (Tabelle b) enthaltenen Identifier.

'Lastin'..... ($F0, Wort) enthält den aktuellen Pegel der NOT-EIN-Eingänge. Bit 0 gibt den Pegel des NOT-EIN-Eingangs von Kanal 1 wieder ('0' oder '1') und Bit 1 den Pegel von Kanal 2. Diese Zelle wird ausgewertet, wenn die Zellen 'p ev No..' oder 'n ev No...' gesetzt sind.

'local'..... ($F4, $F6, Wort) wird nur von der Firmware belegt.

'evntno'..... ($F2, Wort) enthält den aktuellen EVENT des 'NOT-EIN'-Signals.

4.4 Kommandos des Parameter-Buffers

4.4.1 Übersicht der implementierten Kommandos

Der Ablauf der Kommandoübergabe ist bereits im Kapitel 'Übergabe von Parametern und Kommandos' beschrieben worden.

Die Zelle 'stat' im Parameter-Buffer dient der Bestätigung der Kommandoübergabe:

'stat' = $00 --> kein Fehler bei Kommandoübergabe
'stat' ≠ $00 --> Fehler bei Übergabe

In die Zelle 'iocmmd' wird das gewünschte Kommando eingetragen. Die Zellen 'para1' bis 'para4' nehmen die für das entsprechende Kommando benötigten Parameter auf.

Zur Zeit sind 14 verschiedene Kommandos implementiert (bei Verwendung des Controller 82527 kommen die Kommandos 'insert idf' und 'release idf' hinzu):

Cells of parameter buffer							Command
iocmmd [HEX]	Program Label	**para1**	**para2**	**para3**	**para4**	**par32**	
0	svini0	bitrate $0000.. $000F or $0011.. $7F7F	-	-	-	-	init CAN net 1 with bitrate
1	svini1	bitrate $0000.. $000F or $0011.. $7F7F	-	-	-	-	init CAN net 2 with bitrate
4	svmon0	Net 0,1	M code1	M mask1	-	-	trigger monitor 1 or monitor 2
6	nodlk0	Net A 0,1	Idf A 0...2047	Net B 0,1	Idf B 0...2047	-	link Net A/Idf A to Net B/Idf B
7	nodlk1	Net A 0,1	Idf A 0...2047	Net B 0,1	Idf B 0...2047	-	unlink Net A/Idf A from Net B/Idf B
8	autlk0	Net 0,1	Idf 0...2047	-	-	-	activate periodical Tx-transfers (or CMS Watchdog TxId)
9	autlk1	Net 0,1	Idf 0...2047	-	-	-	stop periodical repeat of periodical Tx-transfers (or CMS RTR)
A	iniirq	iolev $0001...$0007	iovec $0000...$00FC	-	-	-	CARD interrupt enable
B	svirq	Net 0,1	Idf start 0...2047	Idf end 0...2047	MODE	-	set MODE
C	domwsp	Net 0,1	client TxId 0...2047	client RxId 0...2047	buffer length 1...32 kBytes	*returns* buffer address	init domain (only CMS transfers)
D	domrbf	Net 0,1	client TxId 0...2047	client RxId 0...2047	-	*returns* buffer address	release CMS buffer
E	grxbuf	handle-no.	buffer length	-	-	*returns* buffer address	get Rx buffer
F	rrxbuf	handle-no.	-	-	-	-	release Rx buffer
10	idfins	Net 0,1	handle 0...2047	ext mode 0,1	dyn mode 0,1,2,3	idf 29/11 0...2947 or $0...2^{29}-1$	only if 29-bit identifier required: Insert identifier
11	idfdel	Net 0,1	handle 0...2047	-	-	-	only if 29-bit identifier required: Release identifier
12 13	-	-	-	-	-	-	reserved
14	gtxbuf	handle-no.	buffer length	Net 0,1	-	*returns* buffer address	get Tx buffer
15	rtxbuf	handle-no.	-	-	-	-	release Tx buffer
FFFF		-	-	-	-	-	'last command done' (Default setting after RESET)

Die mit einem Bindestrich '-' gekennzeichneten Parameter werden nicht ausgewertet.

Tabelle 4.4.2: Kommandos des Parameter-Buffers

4.4.2 Beschreibung der einzelnen Kommandos

Im folgenden werden die einzelnen Kommandos und die dazugehörigen Parameter in aufsteigender Reihenfolge beschrieben.

4.4.2.1 Setzen der Bitrate (Init CAN x mit Bitrate)

Die Bitrate des CAN-Kanals wird neu gesetzt. Die CAN-Kanäle arbeiten mit dieser neuen Bitrate - die über die Kodierschalter voreingestellte Bitrate wird ignoriert. Nach einem RESET arbeitet das System wieder mit den Bitraten der Kodierschalter. Eine Tabelle mit der Zuordnung des Parameters 'bitrate' zur Bitrate findet sich bei der Beschreibung der Buffer-Zellen 'actbr1/2'.

Die Übergabe der Bitrate kann mit den bisher üblichen Werten '0....F' erfolgen, die den Stellungen der Kodierschalter gleichzusetzen sind, oder aber direkt, indem der 16 Bit breite Wert der Register BTR0 + BTR1 des CAN-Controllers eingetragen wird.

Bei laufenden (unvollendeten) Übertragungen würde das Ändern der Bitrate mit großer Wahrscheinlichkeit zu Fehlern führen. Daher wird die Verarbeitung aller Strukturelemente, die sich im Wartezustand befinden, mit der entsprechenden Fehlermeldung abgebrochen.

4.4.2.2 Triggern der Monitor-Funktion (Trigger Monitor x)

Dieses Kommando gibt die Startbedingung für die Aufzeichnung der eintreffenden Daten im Monitor-Buffer. Nach erfolgter Trigger-Bedingung werden alle eintreffenden Nachrichten, für deren Identifier der Monitor-Mode gewählt wurde, mit ihrem Identifier und ihrer 'Ankunftszeit' (nach dem ersten zugelassenen Strukturelement), im Monitor-Buffer abgelegt.

Die Triggerbedingung ist das Eintreffen eines Strukturelementes dessen Eigenschaften in dem durch die Parameter 'M codex' und 'M maskx' (x=1,2) festgelegten Wertebereich liegen.

Die Selektion der eintreffenden Nachrichten über diese Parameter erfolgt über die gleichen Bits, die in der Zelle 'Idf' im Monitor-Strukturelement abgelegt werden.

Bits der Parameter 'M maskx' und 'M codex' (X=1,2)	15...5	4	3...0
Funktion	11Bit-CAN-Id	RTR	LENGTH

Tabelle 4.4.3: Funktion des Parameter 'M codex'

Die Triggerbedingung ist erfüllt, wenn die Bits der eintreffenden Nachrichten folgende Bedingung erfüllen:

```
                                                  !
[Input(CAN-Id., RTR, LENGTH) EXOR M code] AND M mask = $0000
```

Der Parameter 'M maskx' (x=1,2) legt fest, welche Bits des Identifiers auf ihren logischen Zustand untersucht werden sollen.

Der Parameter 'M codex' (x=1,2) legt fest, welchen logischen Zustand die Bits eines Identifiers haben müssen, damit der Empfang dieses Identifiers als Triggerbedingung für den Monitor-Mode anerkannt wird. Es werden nur die Bits ausgewertet, die im Parameter 'M maskx' für die Auswertung ausgewählt wurden.

Übertragen auf eine Wertetabelle heißt das:

M mask	M code	Input-Bit	RESULT	Auswertung
0 0	0 0	0 1	0 0	Input-Bit = don't care
0 0	1 1	0 1	0 0	Input-Bit = don't care
1 1	**0** 0	**0** 1	0 1	Bit = OK Bit = false
1 1	1 **1**	0 **1**	1 0	Bit = false Bit = OK

Tabelle 4.4.4: Funktion des Parameter 'M mask'

Ist ein Bit in der Zelle 'M mask' = '0', so wird das entsprechende Bit des eintreffenden Idf. nicht ausgewertet, d.h. wenn z.B. alle Bits von 'M maskx' (x=1,2) 'null' sind, werden alle eintreffenden Nachrichten als Monitor-Trigger ausgewertet - es findet keine Selektion statt.

Ist ein Bit der Zelle 'M mask' = '1', so muß das Bit des eintreffenden Idf. den gleichen Wert wie das entsprechende Bit in der Zelle 'M codex' haben, um als Triggerbedingung akzeptiert zu werden.

<u>Beispiel</u>:
M mask1 = 0000 000**1** **1**000 0000
M code1 = xxxx xxx**1** **0**xxx xxxx
---
zugelassene = xxxx xxx**1** **0**xxx xxxx
Identifier

x= don't care

Die Aufzeichnung endet, wenn der Monitor-Buffer gefüllt ist. Nach erneutem Start des Triggers werden die alten Daten überschrieben.

4.4.2.3 Verknüpfung von CAN-Identifiern verschiedener Netze
(Link Net A/Idf A to Net B/Idf B)

Über dieses Kommando wird der CAN-Net A mit dem Identifier Idf A bidirektional mit dem CAN-Net B (Idf B) verknüpft (bzw. die Verbindung wieder aufgehoben). D.h., alle auf Net A eintreffenden Daten mit dem Identifier 'A' werden auf Net B mit dem Idf. 'B' gesendet, und alle auf Net B empfangenen Daten werden auf Net A gesendet.

Sowohl Net A, als auch Net B können den 'physikalischen CAN-Kanälen 1 oder 2 zugeordnet werden:

Parameter 'Net A' = 0/1 und
'Net B' = 0/1

Kanal 1 -->'0'
Kanal 2 -->'1'

Neben dem einfachen Weiterleiten von CAN-Frames von einem Netz zum anderen ist es möglich, Nachrichten auf den Netzen zu senden und zu empfangen. Hierbei werden die beiden Identifier der Netze als ein Identifier betrachtet, d.h. wird z.B. eine Nachricht auf Identifier A/Net A gesendet, so wird sie parallel dazu auch auf Identifier B/Net B gesendet.

4.4.2.4 Periodisches Senden von Tx-Nachrichten (Activate periodical Tx-Transfers oder CMS-Watchdog)

Dieses Kommando dient der periodischen Wiederholung von Tx-Transfers, bzw. dem Rücksetzen dieses Modes beim Einsatz der Standard-EPROMs ohne CMS-Implementierung.

CMS-Implementierung
Sind die eingesetzten EPROMs mit einer CMS-Implementierung versehen, so entfällt die Funktion 'Periodisches Senden' und statt dessen wird über dieses Kommando der CMS-Watchdog-Tx-Identifier und die Netz-Nummer übergeben.
Der Watchdog-Identifier wird in der Zelle 'para2' und die Netz-Nr. in der Zelle 'para1' übergeben.

Initialisierung
Die Initialisierung dieses Modes erfolgt durch die Übergabe der Parameter 'iocmmd', 'Net' und 'Idf' an den Parameter-Buffer.

Aktivierung
Das Setzen der Wiederholungsdauer über die Zelle 'TOUT' im Daten-Strukturfeld des Identifiers (negative Werte!; siehe Kap. 'Inhalt eines Daten-Strukturelementes') aktiviert den Mode. Der eingegebene Wert gibt die Zeit zwischen dem Abschluß der letzten Tx-Übertragung und dem Beginn der neuen Übertragung an.

Die über 'TOUT' eingegebene Periodendauer der Tx-Frames auf dem CAN-Bus hat eine zeitliche Unsicherheit von ±1 ms.

Start
Zum Starten der periodischen Transfers ist es notwendig, einmalig einen Tx-Transfer des Strukturelementes erfolgreich zu übertragen. Jeder weitere Tx-Transfer auf diesem Identifier startet den Mode neu.

Ende
Die periodischen Tx-Transfers werden beendet, indem das Kommando im Parameter-Buffer zurückgesetzt, oder die Time-Out-Zeit auf positive Werte gesetzt wird.

4.4.2.5 Freigabe des Karten-Interrupts mit iovec und iolev (CARD-Interrupt-Enable)

Dieses Kommando initialisiert den PIT 68230 zur Erzeugung von VMEbus-Interrupts. Das Kommando ersetzt das in älteren Firmware-Versionen notwendige Setzen der einzelnen PIT-Interrupt-Register durch den Anwender. (Das direkte Setzen der PIT-Register ist weiterhin möglich, wird hier jedoch nicht mehr beschrieben.)

Sind die beiden Zellen 'iolev' und 'iovec' '0', so wird kein VMEbus-Interrupt ausgelöst (Default-Wert nach RESET).

Dem Parameter 'iolev' kann der Anwender die Werte 1...7 zuordnen, die den entsprechenden VMEbus-Interrupt-Level 1...7 festlegen.

Der Parameter 'iovec' legt die obersten sechs Bit des Interrupt-Vektors, der auf den VMEbus gegeben werden soll, fest (Vector-Base). Die untersten zwei Bits (IRQS1, IRQS0) werden von der Hardware entsprechend der lokalen Interrupt-Quelle gesetzt. Sie bestimmen das 'Interrupt-Acknowledge-Bit', das im PIT 68230 gesetzt werden muß, um den VMEbus-Interrupt zurückzunehmen.

Bit	7	6	5	4	3	2	1	0
Funktion	Vector-Base (User defined)						IRQS1	IRQS0

Tabelle 4.4.5: Funktion des Parameters 'iovec'

Erklärung der Bits in der Zelle 'iolvec':

Vector-Base....Diese Bits kann der Anwender frei programmieren.

Bit IRQS1	IRQS0	Interrupt-Quelle
0	0	NOT-EIN-Signal (Flankenwechsel erfolgt)
0	1	Command-Interface (nicht benutzt)
1	0	FIFO (FIFO 'Data from VMEbus' von VMEbus gefüllt)
1	1	CAN-Server (FIFO 'Data to VMEbus' nicht leer)

Tabelle 4.4.6: Funktion der Bits IRQS0 und IRQS1

Der Inhalt der Zellen 'iolev' und 'iovec' wird von der Firmware im Parameter-Buffer in den zellen 'ioilev' und 'ioivec' abgelegt. Die programmierten Werte können dort gelesen werden.

4.4.2.6 Betriebsart (Mode) auswählen (Set Mode)

Mit diesem Kommando wird der Mode für Identifier-Gruppen oder einzelne Identifier gesetzt. Soll für nur einen Identifier der Mode festgelegt werden, so wird in die Zellen 'Idf start' und 'Idf end' der gleiche Wert eingetragen.

Das Kommando 'Set Mode' wird über den Parameter-Buffer aufgerufen, indem in die Zelle iocmmd der Wert $B eingetragen wird. Die Zellen para1...para4 und para32 müssen wie in dem vorangegangenen Kapitel 'Übersicht der implementierten Kommandos' gesetzt werden. In die Zelle para4 wird der Wert für MODE eingetragen. Die folgende Tabelle gibt eine kurze Übersicht über die implementierten Modes:

MODE	ausgewählter Übertragungsmodus
0000	Rx-Transfers disabled (Tx-Only)
0001	abhängig vom Eintrag in Zelle LENGTH im Daten-Strukturelement: Rx-Transfer enabled, RTR senden
0002	Activate Tx-Transfer bei RTR-Empfang
0004	Monitor
0005	CAN-Serial
000F	führt zum Aktivieren des vorherigen Modes
80xy	Rx-Buffer: Handle-Zuordnung xy = 00...FF
81xy	Tx-Buffer: Handle-Zuordnung xy = 00...FF

Tabelle 4.4.7: Übersicht der implementierten Modes

Beschreibung der MODEs:

Das Senden von Tx-Frames ist generell, unabhängig von den hier beschriebenen Modes, immer möglich !

Rx-Transfer... disabled
(Mode 0)

In dieser Betriebsart können nur Daten (0...8 Bytes) auf den CAN-Bus gesendet werden.

In der Zelle 'TOUT' im Daten-Strukturelement kann eine Time-Out-Zeit eingetragen werden.

Das Ende der Übertragung kann dem Anwender über den VMEbus durch einen Interrupt mitgeteilt werden, sofern die Zellen 'iolev' und 'iovec' des Kommandos 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sind, und die Zelle 'EVTRIG' im Daten-Strukturelement ≠ '0' ist oder die Zelle 'XTTID' im Kontroll-Strukturelement ≠ '0' ist.

Gültige Endebedingungen, die in diesem Fall zum Triggern des VMEbus-Interrupts führen:
- Tx-Transfer erfolgreich abgeschlossen
- Tx-Transfer wegen Fehler abgebrochen (z.B. Time-Out)

Für den Eintrag in die Zelle 'LENGTH' gibt es in diesem Mode folgende Alternativen:

1. Eintrag der Länge als positive Zahl (0000...0008):
   Der Eintrag der Anzahl der Bytes als positive Zahl führt nicht zum Start der Abarbeitung des Übertragungsauftrags.
2. Eintrag der Länge als negative Zahl (FFFF...FFF7):
   Wird die Anzahl der zu übertragenden Bytes als negative HEX-Zahl eingetragen, so wird die Abarbeitung des Sendeauftrags mit dem Eintrag gestartet.
3. Eintrag der Länge und der Zahl '6' (0060...0068):
   Funktion identisch zu 2.

Rx-Transfers...
(Mode 1)

Der Anwender kann in diesem Mode die CAN-Daten im Daten-Strukturfeld transparent lesen.

Für den Rx-Mode gilt:
Der Empfang der Daten kann dem Anwender über den VMEbus durch einen Interrupt mitgeteilt werden, sofern die Zellen 'iolev' und 'iovec' des Kommandos 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sind, und die Zelle 'EVTRIG' im Daten-Strukturelement ≠ '0' ist oder die Zelle 'XTTID' im Kontroll-Strukturelement ≠ '0' ist.

Gültige Endebedingungen, die in diesem Fall zum Triggern des VMEbus-Interrupts führen:
- Rx-Transfer erfolgreich abgeschlossen

Für die Rx-Transfers kann auch eine **Time-Out-Zeit** angegeben werden. Die Time-Out-Zeit wird in die Zelle 'TOUT' im Daten-Strukturelement eingetragen. Innerhalb dieser Zeit wird eine eintreffende Nachricht auf diesem identifier erwartet.

Das Setzen der Zelle 'LENGTH' integriert das CTRL-Strukturelement dieses Identifiers in eine Time-Out-Kette, in der die 'Time-Counter' der eingetragenen Strukturelemente heruntergezählt werden.
Das Bit 'WAITR' (wait for Rx) im Status-Byte des CTRL-Strukturelementes wird auf '1' gesetzt, solange keine Nachricht empfangen wurde, oder ein Übertragungsfehler auftritt.
Nach dem Empfang gültiger Daten wird die Zelle 'LENGTH' gesetzt.

Gültige Endebedingungen, die in diesem Fall zum Triggern des VMEbus-Interrupts führen:
- Rx-Transfer erfolgreich abgeschlossen
- Rx-Transfer wegen Time-Out abgebrochen

RTR senden...
(Mode 1)

In diesem Mode wird nach dem Setzen der Zelle 'LENGTH' auf im angegebenen Wertebereich auf dem entsprechenden Identifier ein RTR-Frame gesendet, das die Rücksendung von Daten veranlassen soll. Der Eintrag der Länge, der beim CAN-Bus-Protokoll ohne Bedeutung ist, wird hierbei mit übertragen.

Das Senden des RTR-Frames erfolgt analog zum MODE '0'. Die Auswahl des 'Time-Out-Modes' und das Auslösen eines Interrupts nach Eintreten einer gültigen Endebedingung sind möglich.

Das Ende des RTR-Sendevorgangs wird im 'STATUS'-Byte im Daten-Strukturelement durch den Wert '$FFFE' angezeigt.

Nach erfolgreichem Senden des RTR-Frames wird auf den Empfang von Nachrichten auf diesem Identifier gewartet. Ab diesem Zeitpunkt kann dieser Mode wie ein 'Rx-Transfer mit Time-Out' betrachtet werden.

Gültige Endebedingungen, die in diesem Fall zum Triggern des VMEbus-Interrupts führen:
- Rx-Transfer erfolgreich abgeschlossen
- Rx-Transfer wegen Time-Out abgebrochen

Activate... Tx-Transfer bei RTR-Rx
(Mode 2)

Beim Empfang eines RTR-Frames wird dieses Daten-Strukturelement (0...8 Byte) gesendet. Prinzipiell läuft die Sendung wie in MODE '0' ab.

Unterschiede ergeben sich bei der Interrupt-Erzeugung und der Quelle der Daten, die gesendet werden sollen:

Wird die Zelle 'EVTRIG'≠'0' gesetzt, dann wird nach dem Eintreffen des RTR-Frames ein VMEbus-Interrupt ausgelöst und die zu sendenden Daten müssen vom VMEbus bereitgestellt werden. (Für den VMEbus-IRQ müssen die Zellen 'iolev' und 'iovec' des Kommandos 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt sein!)

Wird die Zelle 'EVTRIG'='0' gesetzt, dann werden die zu sendenden Daten aus dem lokalen Daten-Strukturelement der CAN2 übertragen.

Monitor...
(Mode 4)

Treffen im Monitor-Mode Nachrichten auf diesem Identifier ein, dann werden sie nicht in dem zugeordneten Daten-Strukturelement abgelegt, sondern in der Reihenfolge ihres zeitlichen Eintreffens im Monitor-Buffer gespeichert.

Dieser Mode dient zur Überwachung von Übertragungen ausgewählter CAN-Identifiern. Im Monitor-Buffer werden nicht nur Daten <u>eines</u> Identifiers abgelegt, sondern alle nach der Triggerbedingung des Monitor-Buffers eintreffenden Sendungen, die für diesen Mode in ihrem Strukturelement konfiguriert sind.

Die Aufzeichnung endet, wenn der Monitor-Buffer gefüllt ist. Das Ende der Aufzeichnung kann nicht durch einen VMEbus-Interrupt angezeigt werden.

Weitere Informationen zum Monitor-Mode können auch dem vorangegangenen Kapitel 'Monitor-Buffer' und der Beschreibung des Kommandos 'Trigger Monitor 1 or Monitor 2' entnommen werden.

CAN-Serial...
(Mode 5)

Eintreffende Daten werden nicht nur im Strukturelement, sondern auch im FIFO abgelegt, um dem VMEbus-Master die Möglichkeit zu geben, kontinuierlich eintreffende Daten in ihrer zeitlichen Reihenfolge über einen Interrupt-Zyklus abzuarbeiten.

Weitere Informationen zum CAN-Serial-Mode können auch dem vorangegangenen Kapitel gleichen Namens entnommen werden.

Aktivieren des vorherigen Modes... (Mode F)	Wird für Mode der Wert '$000F' eingestellt, so wird der vorherige Mode wieder aktiviert. Beispiel: Zuerst war der Mode '00' ausgewählt, also Rx-Transfers disabled. Danach wurde auf den Monitor Mode ('04') umgeschaltet. Wird jetzt der Mode auf '$000F' gesetzt, so würde wieder der Mode '00' aktiviert.
Rx-Buffer: Handle-Zuordnung... (Mode 80xy)	Bei der Nutzung des Rx-Buffers werden verschiedenen 'Handle-Nummern' Buffer-Größen zugeordnet. Den Identifiern, die überwacht werden sollen, werden über diese Mode-Zuordnungen wiederum die Handle-Nummern zugewiesen. Weitere Informationen finden sich unter der Beschreibung des Kommandos 'E' (Get Rx-Buffer).
Tx-Buffer: Handle-Zuordnung... (Mode 81xy)	Bei der Nutzung des Tx-Buffers werden verschiedenen 'Handle-Nummern' Buffer-Größen zugeordnet. Den Identifiern, die überwacht werden sollen, werden über diese Mode-Zuordnungen wiederum die Handle-Nummern zugewiesen. Weitere Informationen finden sich unter der Beschreibung des Kommandos '$14' (Get Tx-Buffer).

4.4.2.7 CMS-Initialisierung (Init Domain)

Für den Upload oder Download über das CMS-Protokoll benötigt die lokale Firmware einen Speicherbereich (CMS-Buffer), um Parameter zu übergeben und Daten abzulegen.

Mit dem Kommando 'Init Domain' wird der CMS-Buffer angelegt und die Rx- und Tx-Identifier, über die der Transfer erfolgen soll, werden übergeben.
Die übergebene Buffer-Länge bezieht sich auf die Größe des CMS-Daten-Buffers, der zur Verfügung gestellt werden soll. Für die Größe können Werte zwischen einem Byte und 32 kBytes gewählt werden.

Nach der Ausführung des Kommandos kann die absolute lokale Startadresse des CMS-Buffers im Rückgabeparameter des Parameter-Buffers 'retpar' oder in der Zelle 'NET LINK/BUFFER-POINTER' im Kontroll-Strukturelement der eingesetzten Identifier gelesen werden.

4.4.2.8 Freigabe des CMS-Buffers (Release Buffer)

Wird der CMS-Buffer nicht mehr benötigt, kann der Speicherbereich wieder freigegeben werden. Als Parameter werden hier lediglich die Identifier und die Netz-Nummer übergeben.

4.4.2.9 Rx-Buffer

Die grundlegenden Funktionen der Rx-Buffer wurden bereits in dem vorangegangenen Kapitel gleichen Namens beschrieben.

Vor der Zuordnung der Identifier muß das Kommando 'Get Rx-Buffer' aufgerufen werden. Die lokale Firmware richtet daraufhin einen Rx-Buffer ein. Jeder Rx-Buffer erhält beim Aufruf des Kommandos 'Get Rx-Buffer' eine sogenannte Handle-No..

Dann werden die Identifier ausgewählt, die im Buffer aufgezeichnet werden sollen. Die Zuordnung zwischen Identifiern und Buffer erfolgt über die Handle-Nummern: Einzelnen Identifiern oder Identifier-Blöcken können über die Mode-Zuordnung Handle-Nummern zugeordnet werden.

Identifier mit der gleichen Handle-No. werden also in einem gemeinsamen Rx-Buffer aufgezeichnet.

Die Aufzeichnung der Identifier im Rx-Buffer beginnt mit dem Aufruf dieses Kommandos. Nach dem Kommandoaufruf kann die absolute Startadresse des Rx-Buffers im Rückgabeparameter des Parameter-Buffers 'para32' gelesen werden (Adresse des Parameters siehe vorangegangenes Kapitel 'Funktion und Aufbau).

Über das Kommando 'Release Rx-Buffer' wird der Speicherbereich wieder freigegeben.

4.4.2.10 Insert/Release Identifier (nur mit CAN-Controller 82527)

Diese beiden Kommandos sind nur implementiert, wenn der CAN-Controller 82527 bestückt ist. Wird der 82C200 eingesetzt, so können diese Kommandos nicht ausgeführt werden. Welcher der beiden Controller eingesetzt ist, können Sie in der Zelle 'canctrl' im Prameter-Buffer lesen (ASCII-codiertes Langwort auf Adresse $8044).

Ist der Controller 82527 bestückt, so können Sie die abweichenden Eigenschaften dieses Controllers und der Firmware auf den beigefügten Seiten 'Besonderheiten der Implemetierung des Controllers 82527 auf der CAN2B' entnehmen.

Mit dem Kommando 'Insert Identifier' können insgesamt 2047 verschiedene Identifier zur Bearbeitung über die CAN2B ausgewählt werden. Die Identifier können 11 Bit oder 29 Bit lang sein. Der gewünschte Identifier wird in den vier Bytes der Zelle des Parameters 'para32', der normalerweise nur für Rückgabeparameter vorgesehen ist, übergeben.

Jedem Identifier wird ein *Handle* zugeordnet (nicht zu verwechseln mit der Handle-No. des Rx-Buffers!). Das Handle beschreibt den Speicherplatz in einer Tabelle, in der eingegebene Identifier zusammen mit den Parametern 'ext mode' und 'dyn mode' abgelegt wird.

In dem Parameter *ext mode* (extended/standard-mode) wird festgelegt, ob der Identifier 11 Bit oder 29 Bit lang ist:

```
ext mode = 0            ->    11 Bit Identifier
ext mode = 1            ->    29 Bit Identifier
```

Im Parameter *dyn mode* (dynamisch/resident-mode) wird festgelegt, ob der Identifier innerhalb der 14 Objekte des Controllers 82527 dynamisch verwaltet werden soll, oder resident (fest) einem Object zugeordnet werden soll. Welchem Objekt der Identifier zugeordnet wird bestimmt in beiden Fällen die Firmware.
Rx-Identifer als resident zu definieren empfiehlt sich, wenn bekannt ist, daß auf diese Identifer besoders häufig zugegriffen wird, da so die lokale Firmware entlastet wird. Zwingend notwendig ist es, einen Tx-Identifier als resident zu definieren, wenn auf ein RTR-Frame reagiert werden soll. Dynamische Tx-Identifier reagieren nicht auf eingehende RTRs!

```
dyn mode =  0     ->    dynamischer Rx-Identifier
            1     ->    dynamischer Tx-Identifier
            2     ->    residenter  Rx-Identifier
            3     ->    residenter  Tx-Identifier
```

Achtung! Der dyn-mode '0' (dynamischer Rx-Identifier) bildet einen Sonderfall: Alle für diesen Mode ausgewählten Identifier müssen die gleiche Länge haben! Jeweils der letzte übergebene Eintrag für 'ext mode' ist für alle Identifier des dyn modes '0' gültig.
Bei allen anderen Modes können für Identifier, denen der selbe Mode zugeordnet wurde, unterschiedliche Längen (11 oder 29 Bits) gewählt werden.

4.4.2.11 Tx-Buffer

Die grundlegenden Funktionen der Tx-Buffer wurden bereits in dem vorangegangenen Kapitel gleichen Namens beschrieben.

Vor der Zuordnung der Identifier muß das Kommando 'Get Tx-Buffer' aufgerufen werden. Die lokale Firmware richtet daraufhin einen Tx-Buffer ein. Jeder Tx-Buffer erhält beim Aufruf des Kommandos 'Get Tx-Buffer' eine sogenannte Handle-No..

Dann werden die Identifier ausgewählt, die im Buffer aufgezeichnet werden sollen. Die Zuordnung zwischen Identifiern und Buffer erfolgt über die Handle-Nummern: Einzelnen Identifiern oder Identifier-Blöcken können über die Mode-Zuordnung Handle-Nummern zugeordnet werden.

Identifier mit der gleichen Handle-No. werden also in einem gemeinsamen Tx-Buffer aufgezeichnet.
Über das Kommando 'Release Rx-Buffer' wird der Speicherbereich wieder freigegeben.

5. Zusammenfassung der Interrupt-Optionen

5.1 Übersicht

Dieses Kapitel soll eine zusammenfassende Übersicht über die Interrupt-Erzeugung und Verwaltung in Bezug auf die beiden CAN-Kanäle auf der CAN2 geben. Interrupts, deren Funktion sich auf andere Baugruppen der CAN2 beziehen, sind hier nicht mit aufgeführt.

5.2 Anwenderdefinierte Interrupts/Interrupt Handling

5.2.1 Allgemeines/PIT 68230

Der Anwender kann beim Eintreten verschiedener Ereignisse einen Interrupt auf dem VMEbus auslösen.

Die erste Bedingung für die Freigabe eines Interrupts auf dem VMEbus ist das Initialisieren der Interrupt-Register. In diesen Registern muß der Anwender den gewünschten Interrupt-Level und Interrupt-Vektor eintragen. Die Übergabe der gewünschten Registerinhalte erfolgt über das Kommando 'CARD-Interrupt-Enable' des Parameter-Buffers.

In die Zelle 'iolev' wird der Level des gewünschten Interrupts und in die Zelle 'iovec' die obersten sechs Bits des Interrupt-Vektors, der im Interrupt-Acknowledge-Zyklus auf den VMEbus gegeben werden soll, eingetragen.

Die Bits 2 bis 7 des Interrupt-Vektors in der Zelle 'iovec' kann der Anwender frei programmieren. Die untersten beiden Bits (IRQS0 und IRQS1) enthalten Informationen über die Interrupt-Quelle. Es wird im wesentlichen zwischen dem 'NOT-EIN-Interrupt', dem 'FIFO-VME-to-CAN-Full-Interrupt' und dem 'CAN-Server-Interrupt' unterschieden. Der 'CAN-Server-Interrupt' kann beim Eintreten einer gültigen 'Endebedingung' eines CAN-Identifiers, im CAN-Serial-Mode oder bei CMS-Transfers (sofern implementiert) ausgelöst werden.

Die Zellen 'iolev' und 'iovec' sind bereits im Kapitel 'Kommandos des Parameter-Buffers' beschrieben worden.

Die Initialisierung der PIT-Register auf der CAN2 muß nach jedem RESET des VMEbus-Masters durchgeführt werden.

Der Anwender muß, wenn er die Interrupt-Option gewählt hat, sicherstellen, daß der VMEbus-Interrupt durch eine Interrupt-Routine abgefangen wird!

Um das Rücksetzen des Interrupts zu verdeutlichen, sollen hier kurz die Zusammenhänge mit der Hardware erwähnt werden:

Der VMEbus-Interrupt wird über den Controller PIT 68230 erzeugt. Mit Hilfe des PIT wird auch der Interrupt-Vektor generiert. Der VMEbus-Interrupt wird zurückgenommen, wenn der entsprechende PIT-Interrupt zurückgesetzt wird.

Die Interrupts des PIT werden zurückgesetzt, indem das jeweilige, durch den Interrupt-Vektor bestimmte Bit des PSR (Port Status Register) des PIT auf '1' gesetzt wird.

iovec IRQS1	IRQS0	Interrupt-Quelle	Rücknahme der Interrupt-Ursache
0	0	NOT-EIN-Signal (Flankenwechsel erfolgt)	Bit 0 des PSR auf '1' setzen
0	1	Command-Interface (nicht benutzt)	-
1	0	FIFO (FIFO 'Data from VMEbus' von VMEbus gefüllt)	1. warten bis Bit 6 des PSR des PIT 68230 = '1' (FIFO frei) 2. Bit 2 des PSR auf '1' setzen
1	1	CAN-Server (gültige Endebedingung ein-getreten, CAN-Serial oder CMS-Transfer)	1. Bit 3 des PSR auf '1' setzen 2. Lesen des FIFO 'Data to VMEbus' wortweise auf relativer I/O-Adresse $01602

Tabelle 5.2.1: Rücksetzen des VMEbus-Interrupts über PSR des PIT 68230

Beispiel: Rücksetzen des FIFO 'VME-to-CAN-Full'-Interrupt

```
---   Interrupt-Entry   ---
MOVEB=$04,iack+can2base;      nur dieser Interrupt!
---   weitere Interrupt-Routine ---

'iack'.......Interrupt-Acknowledge-Adresse (incl. Offset des I/O-Bereiches) des
             PIT 68230-PSR: $0601B + $78000
'can2base'...Basisadresse der CAN2-Platine
```

Das Setzen des PSR nimmt nur den VMEbus-Interrupt zurück, nicht aber die lokale Interrupt-Ursache. Die Rücknahme der Interrupt-Ursache wird im folgenden bei den einzelnen Interrupt-Quellen beschrieben.

5.2.2 'NOT-EIN'-Interrupt

Der 'NOT-EIN-Interrupt' teilt dem Anwender mit, daß sich der Pegel des NOT-EIN-Signals geändert hat. Dieser Interrupt wird nur erzeugt, wenn der Anwender die Zellen 'iolev' und 'iovec' gesetzt hat und seine Event-Nr. in eine der Zellen 'p ev No..' oder 'n ev No...' des Parameter-Buffers geschrieben hat. Alle oben aufgeführten Zellen sind in dem Kap. 'Der Parameter-Buffer' beschrieben.

Der Interrupt der 'NOT-EIN'-Eingänge wird nicht über die Hardware ausgelöst. Die lokale Firmware fragt den Zustand dieser Eingänge zyklisch ab und stellt fest, ob sich der Pegel der Eingänge geändert hat. Die max. Zeit zwischen zwei Abfragen beträgt 10 msec.

Der VMEbus-Interrupt wird wie in Abschn. 'Allgemeines' beschrieben, zurückgenommen.

Nach einem NOT-EIN-Interrupt sollte das Setzen der PIT-PSR-Bits so bald wie möglich erfolgen, da auf der CAN2 das Auslösen eines neuen Interrupts so lange verhindert wird, bis der aktuelle Interrupt bestätigt wurde.

5.2.3 FIFO 'VME-to-CAN-Full'-Interrupt

Um diesen Interrupt freizugeben genügt es, die Zellen 'iolev' und 'iovec' zu setzen!

Der 'FIFO-VME-to-CAN-Full-Interrupt' wird von der Hardware ausgelöst, wenn vom VMEbus aus das FIFO 'Data from VMEbus' komplett gefüllt wurde.

Dieses FIFO dient beim Senden von Daten auf den CAN-Bus als Puffer zwischen dem VMEbus und dem CAN-Bus (siehe auch Kap. 'Tx-Transfers'). Sollte es vom VMEbus aus vollständig gefüllt worden sein, dann heißt das, daß der lokale CAN-Server nicht in der Lage war, die Nachrichten in entsprechender Zeit zu verarbeiten.
Das kann z. B. vorkommen, wenn der CAN-Bus ständig mit Tx-Identifiern höherer Priorität belegt ist, und der CAN-Controller daher keine Sendeerlaubnis erhält.

Der Interrupt soll dem Anwender mitteilen, daß er seinen Datenstrom einschränken muß, wenn er keinen Datenverlust riskieren will.

Zurückgenommen wird der Interrupt erst, nachdem wieder eine Zelle des FIFOs frei geworden ist. Dem Anwender wird dies durch den Level von Bit 6 des PSR des PIT 68230 angezeigt. Das FIFO ist gefüllt, wenn Bit 6 den Wert '0' hat. In seiner Interrupt-Routine muß der Anwender also das Bit lesen und warten, bis es den Wert '1' angenommen hat. Jetzt kann der Anwender den PIT-VMEbus-Interrupt, wie im Abschn. 'Allgemeines' beschrieben, zurücknehmen. Erst dann wird die Interrupt-Routine verlassen, und die Übertragung von Daten zur CAN2 wieder aufgenommen.

5.2.4 CAN-Server-Interrupt

Der CAN-Server-Interrupt kann nach dem Eintreten einer gültigen Endebedingung für jeden einzelnen Identifier ausgelöst werden. Gültige Endebedingungen sind z.B. der erfolgreiche Abschluß einer Übertragung oder das Auftreten eines Übertragungs-fehlers (z.B. Time-Out). Er kann außerdem beim Eintreffen von Daten im 'CAN-Serial-Mode' oder während der CMS-Dienste ausgelöst werden.

Um einen VMEbus-Interrupt auslösen zu können, muß neben den Zellen 'iovec' und 'iolev' die Zelle 'EVTRIG' (nur für Interrupt bei Endebedingung) im Daten-Strukturelement des Identifiers, der überwacht werden soll, gesetzt werden . Die Zelle 'EVTRIG' ist im Kap. 'Die Strukturfelder CAN Data1 und CAN Data2' beschrieben.

In den Modes 'Rx-Transfer disabled' (Tx-Only), 'Rx-Transfer mit Time-Out-Option' und 'RTR senden' kann die Zelle 'XTTID' anstelle der Zelle 'EVTRIG' gesetzt werden. Die Zelle 'XTTID' ist im Kap. 'Die Kontroll-Strukturen CAN CTRL1 und CAN CTRL2' beschrieben.

Das Setzen des Modes ist in dem Kapitel 'Parameter- und Kommandoübergabe über den Parameter-Buffer' beschrieben.

Der CAN-Server-Interrupt wird in der Reihenfolge, wie in der vorangegangenen Tabelle bereits beschrieben, zurückgesetzt:

Der Anwender muß den PIT-VMEbus-Interrupt vor dem Lesen des FIFOs zurücknehmen!

Dieses FIFO darf <u>jeweils nur einmal, vom VMEbus aus und nur in der Zugriffsart 'Wort'</u> gelesen werden! Durch jeden Lesezugriff wird der interne FIFO-Counter weitergezählt, d.h. mit einem weiteren Zugriff würde eine andere FIFO-Zelle gelesen, und der FIFO-Counter unzulässigerweise weitergezählt werden!

Diese Reihenfolge ist zwingend notwendig, da während des Lesens bereits weitere Daten eintreffen könnten, deren Interrupt sonst nicht ausgewertet werden könnte. Wird der Interrupt vorher zurückgesetzt, so können neue Daten sofort nachdem der Master die Interrupt-Routine verlassen hat, einen neuen Interrupt auslösen.

Durch das FIFO wird das Strukturelement beschrieben, das den Interrupt ausgelöst hat. Die Endebedingung, die zum Auslösen führte, kann in der Zelle STATUS des Daten-Strukturelementes gelesen werden. In der Beschreibung dieser Zelle im Kap. 'Die Strukturfelder CAN Data1 und CAN Data2' sind die möglichen gültigen Endebedingungen aufgeführt.

Der Inhalt des FIFOs gibt dem Anwender die Information darüber, welcher Identifier auf welchem Kanal der CAN2 den Interrupt ausgelöst hat. Werden die Marken $FFFE ode $FFFF gelesen, so wurde der Interrupt im CAN-Serial-Mode ausgelöst und die folgenden Zellen enthalten einen CAN-Serial-Data-Block (siehe Kap. 'CAN-Serial-Mode').

Bit FIFO ->	15	14 13 ... 6 5 4	3 2 1 0
Idf-Pointer ->	Kanal-Nr.	CAN-Id-No. (11 Bit)	0 0 0 0
Marken für CAN-Serial-Mode	$FFFE -> Node 1 $FFFF -> Node 2		

Tabelle 5.2.2: Inhalt des FIFOs 'Data to VMEbus'

Erklärung der Bits des FIFOs 'Data to VMEbus'

Kanal-Nr..... Dieses Bit gibt den 'physikalischen' CAN-Kanal an, auf dessen Identifier der Interrupt ausgelöst wurde:
Kanal-Nr. = '0' --> Kanal 1
Kanal-Nr. = '1' --> Kanal 2

CAN-Id-No. ..... Diese Bits geben den Identifier (und somit das Strukturelement) an, der den Interrupt ausgelöst hat (0...2047).

Bit 3 bis 0..... Diese Bits werden immer als '0' gelesen.

$FFFE..... Marke für den Beginn eines CAN-Serial-Data-Blocks auf Kanal 1 (Node 0)
$FFFF..... Marke für den Beginn eines CAN-Serial-Data-Blocks auf Kanal 2 (Node 1)

5.2.5 Erkennen der Endebedingung ohne Interrupt (Polling)

Soll beim Eintreten einer Endebedingung kein Interrupt ausgelöst werden, besteht für den Anwender trotzdem die Möglichkeit das FIFO 'Data to VMEbus' zu lesen, um eingetroffene CAN-Bus-Nachrichten auszuwerten. Durch 'pollen' des Bits 7 des PIT 68230-Status-Registers (PSR) kann festgestellt werden, ob sich in dem FIFO Daten befinden.

Das PSR-Bit 7 des PIT 68230 kann mit einem Byte-Zugriff auf der folgenden Adresse gelesen werden:

PIT 68230 Port Status Register: $0601B

Hat Bit 7 des PSR den Wert '1', enthält das FIFO keine Daten, hat das Bit dem Wert '0' enthält das FIFO Daten und kann ausgelesen werden.

Für das Lesen des FIFOs gelten die gleichen Bedingungen die bereits im vorangegangenen Kapitel beschrieben wurden: Es darf jeweils nur einmal gelesen werden, nachdem im PSR angezeigt wurde, daß das FIFO gültige Daten enthält!

Der Inhalt des FIFOs wurde bereits in der vorangegangenen Tabelle beschrieben.

5.3 Firmware-definierte Interrupts

Unter der Bezeichnung 'Firmware-definierte' Interrupts soll hier eine kurze Übersicht über die IRQs gegeben werden, die i.A. im Hintergrund, vom Anwender unbemerkt abgearbeitet werden.

Die Zugriffe des Anwenders auf diese Interrupts sind beschränkt.

Lediglich der sog. 'GIRL'-Interrupt (**G**lobal Access -> **I**nte**r**rupt **L**ocal) wird von dem Anwender gesetzt. Mit ihm teilt der VMEbus-Anwender der lokalen Firmware mit, daß er Zellen im Parameter-Buffer geändert hat, und die neuen Parameter in den Programmablauf integriert werden sollen. Der Einsatz dieses Interrupts ist in dem vorangegangenen Kapitel 'Übergabe von Parametern und Kommandos ' bereits beschrieben worden.

Der Interrupt wird durch das Schreiben des Datums $0000 auf die relative I/O-Adresse $00002 mit einem Wort-Zugriff ausgelöst. Die Adreßbelegung des I/O-Bereichs und eine Beschreibung aller GIRL-Interrupts ist in dem Hardware-Handbuch der CAN2 zu finden (Die Firmware arbeitet zur Parameteraktualisierung nur mit dem o. a. Interrupt!)

<u>Auf die folgenden Interrupts hat der Anwender keinen Einfluß</u>:

Andere im Hintergrund abgearbeitete Interrupts sind z.B. in dem Kap. 'Arbeitsweise der Firmware am Beispiel einfacher Rx- und Tx-Transfers' bereits erwähnt worden. Diese Interrupts kann der Anwender in keiner Weise beeinflussen.

Der 'FIFO-IRQ' wird vom FIFO gesetzt, sobald es nicht mehr leer ist, d.h. der Interrupt ist immer aktiv, sobald sich Strukturelemente im FIFO befinden. Er wird erst zurückgenommen, wenn das FIFO wieder geleert ist. Der FIFO-IRQ teilt der Firmware mit, daß Strukturelemente zum Senden bereitstehen.

Der 'Tx-IRQ' und der 'Rx-IRQ' des CAN-Controllers 82C200 signalisieren die Sende-, bzw. Empfangsbereitschaft des Controllers. Sie werden nur von der Firmware ausgewertet. Der Anwender kann indirekt den Status dieser Interrupts lesen, indem er die Zelle 'STATUS' im Daten-Strukturelement und die Zelle 'STAT' im Kontroll-Strukturelement liest.

6. Leitfaden zu Konfiguration und Betrieb

Die Programmierung der CAN2 kann in drei Abschnitte unterteilt werden:

1. Grundlegende Initialisierung
2. Parametrierung der einzelnen Ids in Daten-Strukturfeld
3. Betrieb der CAN2 (evtl. mit weiteren Änderungen der Initialisierung)

Einige Parameter der CAN2 müssen in der Regel nur einmalig nach einem RESET des Systems gesetzt werden. Hierzu zählt z.B. die Bitrateneinstellung oder die Interrupt-Initialisierung oder auch die MODE-Zuordnung für Identifier, deren Betriebsart während der Systemlaufzeit nicht weiter verändert werden muß.

Die folgende Tabelle soll einen Leitfaden für die Konfiguration bieten. Die Angaben der Tabelle sind nicht als feste Regeln, sondern als Anregung zu interpretieren.

Grundlegende Initialisierung	
1.	Bitrate festlegen, sofern nicht die Default-Bitrate beibehalten wird (-> Parameter-Buffer)
2.	sofern gewünscht: VMEbus-Interrupt initialisieren Der Interrupt-Vektor und der Interrupt-Level werden mit 'iovec' und 'iolev' über das Kommando 'CARD-Interrupt-Enable' im Parameter-Buffer gesetzt. Nicht vergessen: Der VMEbus-Interrupt muß vom Anwender über eine entsprechende Interrupt-Routine versorgt werden!
2a.	nur bei Implementierung der Software für 82527: Eingabe der gewünschten Identifier und ihrer Parameter über das Kommando des Parameter Buffers 'Insert Identifier'
3.	Festlegung der Betriebsart der Identifier: Für die Rx-Identifier des CAN-Bus, die nicht empfangen werden sollen, sollte über das Kommando 'Set Mode' des Parameter-Buffers der MODE '0' gewählt werden. Diese Identifier werden so nicht im Daten-Strukturfeld abgelegt. Dies spart Rechenleistung und entlastet das System. Tx-Transfers sind auf diesen Identifiern weiterhin möglich. Der MODE der Identifier kann auch während des Betriebs verändert werden. Nach einem RESET ist für alle Identifier MODE auf '01' gesetzt.
4.	Falls Identifier der beiden CAN-Netze verknüpft werden sollen, so kann dies über die Kommandos des Parameter-Buffers erfolgen.
5.	sofern gewünscht: Kommando 'Activate periodical Tx-Transfer' im Parameter-buffer für die entsprechenden Identifier aufrufen.
6.	sofern gewünscht: Interrupt-Option für den NOT AUS-Eingang im Parameter-Buffer festlegen
7-n	sofern gewünscht: Rx-Buffer, Init Domain,...

Parametrierung der Ids im Daten-Strukturfeld	
1.	Bei Bedarf die Zelle 'EVTRIG' in den Daten-Strukturelementen der Identifier setzen, die einen VMEbus-Interrupt bei Endebedingung auslösen sollen.
2.	Die Time-Out-Zeiten in den Daten-Strukturelementen der gewünschten Identifier eintragen. (Kann auch während des Betriebs geändert werden.)

Betrieb der CAN2
Nach dieser Konfiguration werden nur noch die Daten der gewünschten Rx-Identifier in den Daten-Strukturelementen aktualisiert. Das Senden von Tx-Frames wird während des Betriebs mit dem Setzen der Zelle LENGTH im Daten-Strukturelement gestartet. Auch die Time-Out-Counter werden durch Setzen von LENGTH aktiviert. Der aktuelle Status der Datentransfers kann jeweils in der Zelle 'STATUS' im Daten-Strukturfeld gelesen werden. Für die selektierten Identifier wird während des Betriebs bei erfüllter Interrupt-bedingung der programmierte VMEbus-Interrupt ausgelöst.

esd gmbh
Abt. Dokumentation
Vahrenwalder Str. 205

D-30165 Hannover

FAX: 0511 - 63 36 50

	Absender (bitte angeben)
	Name
	Firma Dienststelle Abteilung
	Anschrift
❑ Korrekturen ❑ Vorschläge	Telefon FAX

Wir hoffen, daß Ihnen dieses Handbuch bei der Inbetriebnahme und Bedienung von Nutzen war. Sollten Ihnen beim Lesen Fehler aufgefallen sein, so teilen Sie uns dies bitte auf diesem Vordruck mit. Auch wenn Sie Informationen vermissen oder Anregungen zum Handbuch haben, sind wir für eine Mitteilung dankbar.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....