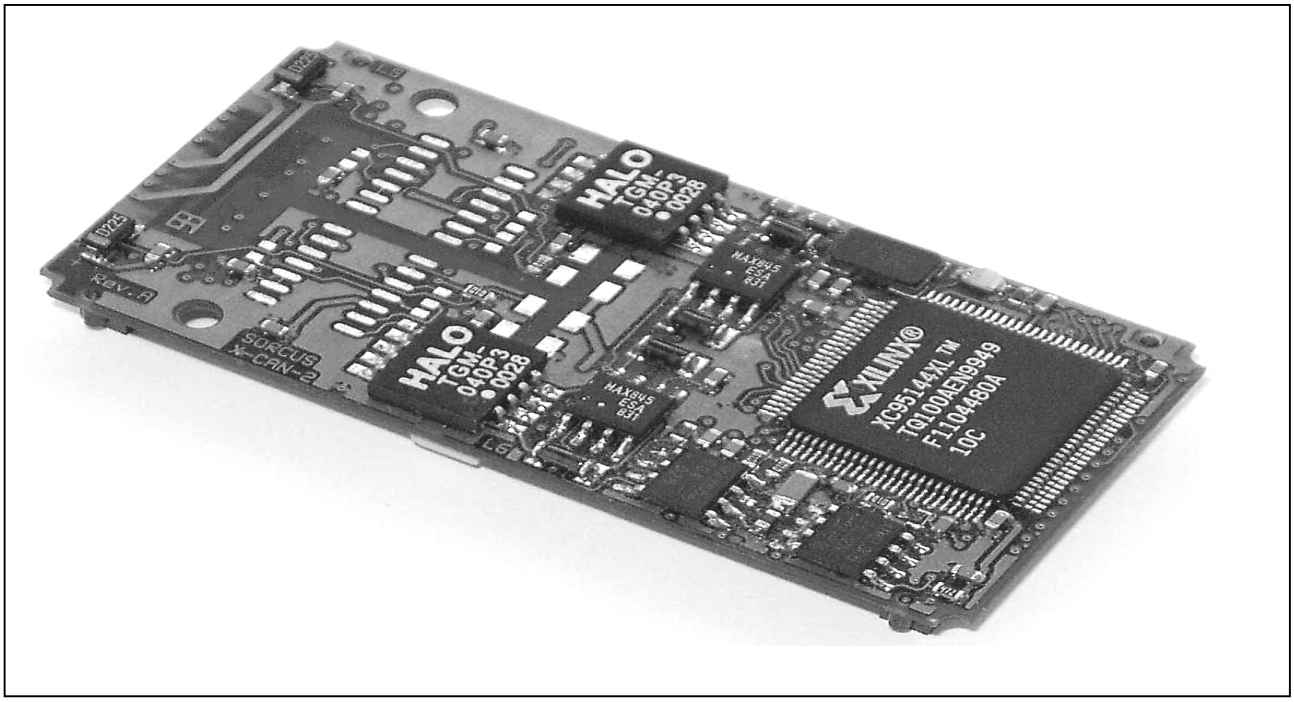
X-CAN-2i

2 unabhängige, isolierte CAN-Bus-Schnittstellen



10.9. X-CAN-2i/H, X-CAN-2i/M, X-CAN-2i/F

Inhaltsverzeichnis

10.9.1. Beschreibung

Das Modul X-CAN-2i bietet zwei unabhängige CAN-Bus-Schnittstellen nach CAN-Spezifikation 2.0 A und 2.0 B (11- und 29-Bit Identifier). Beide Schnittstellen sind sowohl gegenüber der X-Bus-Seite als auch gegeneinander galvanisch getrennt.

Je nach Bestückungsvariante ist die physikalische Busankopplung der beiden Schnittstellen wie folgt ausgeführt:

- zwei High-Speed-Schnittstellen (X-CAN-2i/H = Typ 58, Subtyp 1) nach ISO-DIS 11898 mit Philips 82C250 Transceiver
- zwei fehlertolerante Low-Speed-Schnittstellen (X-CAN-2i/F = Typ 58, Subtyp 2) nach ISO-DIS 11519-1 mit Philips TJA-1054 Transceiver
- je eine High-Speed und eine Low-Speed-Schnittstelle (X-CAN-2i/M = Typ 58, Subtyp 3)

Die maximale Übertragungsgeschwindigkeit beträgt 1 MBit/s bzw. bei der Low-Speed-Schnittstelle 125 kBit/s.

High- und Low-Speed Schnittstellen können nicht im selben Netzwerk betrieben werden!

Die CAN-Bus-Kommunikation wird durch zwei CAN-Bus-Controller INTEL 82527 abgewickelt. Diese sorgen auch für Fehlererkennung und –begrenzung (Bit-Monitoring, Frame- und CRC-Check, Acknowledgement-Überwachung, Überwachung der Bit-Stuffing-Codierungsregel, wiederholtes Senden im Fehlerfall). Durch ihre Full-CAN-Funktionalität reagieren die CAN-Controller nur auf die Bus-Nachrichten, die für die jeweilige Anwendung von Interesse sind.

Bei High-Speed-Schnittstellen kann der Busabschlusswiderstand per Software zugeschaltet werden. Bei Low-Speed-Schnittstellen können die Busabschluss-

Widerstände zwischen den Werten 5,6 kΩ und 510 Ω per Software umgeschaltet werden.

Das Modul ist Interrupt-fähig. Ein Interrupt wird durch eines der folgenden Ereignisse in einer der beiden Schnittstellen ausgelöst:

- Ein CAN-Telegramm ist erfolgreich versendet worden. Das Telegramm muss dazu von mindestens einem anderen Busteilnehmer richtig empfangen worden sein und darf von keinem Busteilnehmer als fehlerhaft erkannt worden sein.
- Ein CAN-Telegramm ist fehlerfrei empfangen worden.
- Ein CAN-Controller erkennt einen stark gestörten Bus bzw. schaltet sich aufgrund gehäufter Übertragungsfehler von der CAN-Bus-Kommunikation ab.
- Der fehlertolerante CAN-Bus-Transceiver erkennt, dass eine der beiden CAN-Leitungen unterbrochen oder mit Masse bzw. der Versorgungsspannung kurzgeschlossen ist oder beide CAN-Leitungen kurzgeschlossen sind (nur für die Low-Speed-Schnittstelle).

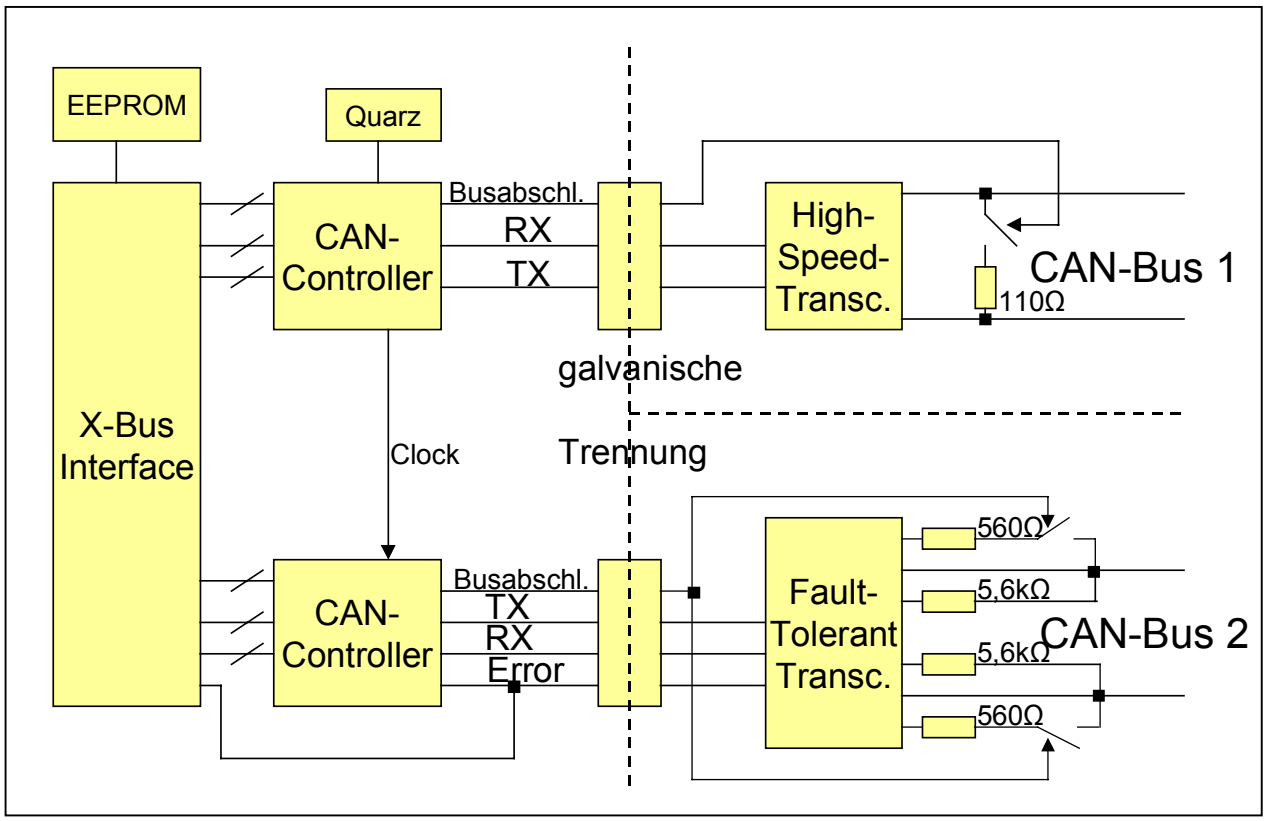
Der Transceiver schaltet bei Erkennen eines Leitungsfehlers automatisch auf eine Ein-Draht-Übertragung (gegen Masse) um.

Bei Abnahme größerer Stückzahlen sind die folgenden weiteren Bestückungsvarianten möglich (für eine oder beide Schnittstellen):

- Bereitstellung der 12 Volt Versorgungsspannung des X-Bus (nicht auf allen Trägerboards vorhanden). Diese ist nicht galvanisch getrennt und wird vom Modul selbst nicht verwendet. Die galvanische Trennung wird dadurch aufgehoben.
- Verbindung der Abschirmung des Bus-Kabels (Shield) mit dem galvanisch getrennten Ground.
- Externe Versorgung (7 – 24 Volt) des galvanisch getrennten Teils der Schnittstelle
- Bestückung nur einer Schnittstelle (X-CAN-1i)

10.9.1.1. Blockschaltbild

Das Blockschaltbild zeigt den Aufbau des X-CAN-2i/M.



10.9.1.2. Der CAN-Bus

Ein CAN-Bus-Netzwerk ist ein Bussystem, bei dem mehrere Teilnehmer den Bus gleichzeitig anfordern können. Ein Teilnehmer sendet eine Nachricht, alle anderen Teilnehmer hören diese Nachricht mit. Mit einer Nachricht können sowohl Daten an andere Teilnehmer verschickt werden als auch Daten eines anderen Teilnehmers angefordert werden.

Die Daten werden hierbei immer zusammen mit einem CAN-Identifier der im folgenden als ID bezeichnet wird, auf dem CAN-Bus verschickt. Die Länge der Daten ist auf maximal 8 Byte pro Telegramm begrenzt. Die ID wird von anderen Teilnehmern als Kriterium verwendet, ob die Nachricht für sie von Interesse ist oder nicht. Weiterhin legt der Wert der ID die Priorität der Nachricht auf dem CAN-Bus fest. Wenn mehrere Nachrichten gleichzeitig gesendet werden, setzt sich im zeitlichen Verlauf die Nachricht mit der kleinsten ID gegenüber den anderen Nachrichten durch.

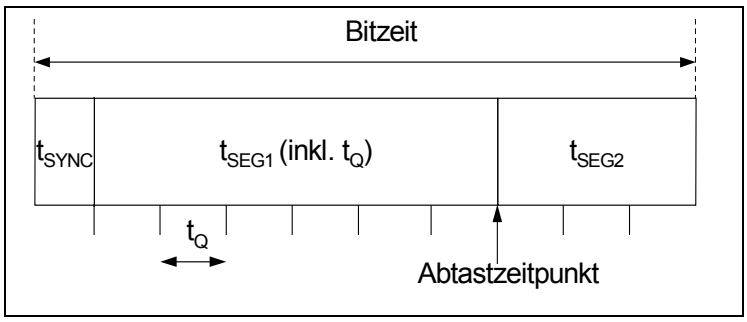
10.9.1.3. Erläuterung der Bit-Timing-Parameter

Laut CAN-Spezifikation ist ein Bit in vier Zeitabschnitte unterteilt:

- Zeitabschnitt t_{SYNC} zur Synchronisation
- Zeitabschnitt t_{PROP} zur Kompensation der physikalischen Verzögerungszeiten
- Zeitabschnitt t_{SEG1} vor dem nominellen Abtastzeitpunkt
- Zeitabschnitt t_{SEG2} nach dem nominellen Abtastzeitpunkt

Die beiden Zeitabschnitte t_{SEG1} und t_{SEG2} dienen der Nachsynchronisation zum Ausgleich der Phasendifferenz zwischen Sende- und Empfangsoszillator. Für die auf dem Modul verwendeten CAN-Controller INTEL 82527 werden t_{PROP} und t_{SEG1} zu einem Segment t_{SEG1} zusammengefasst.

Alle Zeitabschnitte werden in Vielfachen des Time-Quantums t_Q angegeben. Das Time-Quantum t_Q ist immer ein ganzzahliges Vielfaches der Periodendauer des CAN-Controller-Taktes auf dem Modul. Diese beträgt 125 ns.



Die Bitzeit berechnet sich nach der folgenden Formel:

$$\text{Bitzeit} = (t_{SEG1}/t_Q + t_{SEG2}/t_Q + 1) \cdot t_Q$$

Zusätzlich kann die *maximale Synchronisationssprungweite* (Resynchronisation Jump Width t_{SJW}) eingestellt werden. Dies ist die maximale, pro Nachsynchronisation zulässige Verlängerung bzw. Verkürzung von t_{SEG1} und t_{SEG2}. Diese Größe wird ebenfalls als Vielfaches von t_Q angegeben.

Zur Verbesserung der Störsicherheit kann die *Abtastrate pro Bit* (Samples per Bit) auf den Wert = 3 eingestellt werden.

Die Bit-Timing-Parameter sind frei konfigurierbar. Allerdings müssen folgende Bedingungen[2] erfüllt sein:

[2] Der Zeitabschnitt t_{PROP} ist physikalisch vorgegeben: $t_{PROP} = 2 \cdot (t_{PHYS} + t_{DRV} + t_{CC})$. t_{PHYS}, t_{DRV} und t_{CC} sind die maximalen Verzögerungen, entstehend durch die physikalische Signallaufzeit auf dem Bus (berechnet sich aus Materialkonstante (typ < 10 ns/m) · Länge), durch den Ausgangstreiber (max. 80 ns) sowie durch den Eingangskomparator des CAN-Controllers (max. 60 ns).

- $t_{SEG1}/t_Q + t_{SEG2}/t_Q \geq 7$
- $t_{SEG2} \geq t_{SJW}$
- $t_{SEG1} \geq t_{SJW} + t_{PROP}$ für Samples per Bit=1
- $t_{SEG1} \geq t_{SJW} + t_{PROP} + 2 \cdot t_Q$ für Samples per Bit=3

Wenn Sie sich an der CiA-Empfehlung 102, Version 2.0 orientieren wollen, so verwenden Sie eine der in der folgenden Tabelle aufgeführten *Standard-Bitraten* und stellen die dafür angegebenen Bit-Timing-Parameter ein:

Bitrate (kBit/s)	**t_Q (ns)**	**t_{SEG1}/t_Q**	**t_{SEG2}/t_Q**	**t_{SJW}/t_Q**	**Samples per Bit**
10	5000	16	3	2	1
20 [3]	2500	16	3	2	1
50	1000	16	3	2	1
125	500	13	2	1	1
250 [4]	250	13	2	1	1
500 [4]	125	13	2	1	1
800 [4]	125	7	2	1	1
1000 [4]	125	5	2	1	1

Beachten Sie auch, dass das Erzielen hoher Bitraten Auswirkungen auf die maximal zulässige Buslänge hat!

[3] Werkseitig eingetragene Bit-Timing-Parameter.

[4] Nur für High-Speed Schnittstellen

10.9.1.4. Message-Objekte

Nutzdaten und ID werden zu einem Message-Objekt zusammengefasst und vom Modul-Device-Treiber verwaltet. Die eindeutige Kennzeichnung eines Message-Objekts erfolgt durch den Identifier.

Bei der Vergabe der ID an ein Message-Objekt muss folgendes beachtet werden:

- Die ID muss für einen Busteilnehmer eindeutig sein, d.h. er darf sie höchstens einem Nutzdatensatz zuordnen.
- Eine ID darf nur einem Teilnehmer als aktiver Sender zugeordnet sein, d.h. nur dieser eine Teilnehmer darf die der ID zugeordneten Nutzdaten auf dem Bus versenden. Es können mehrere Busteilnehmer als Empfänger der Daten auftreten.
- Für jede der beiden CAN-Schnittstellen muss festgelegt werden, ob sie mit 11- oder 29-Bit-IDs benutzt werden soll (Application Note **AN099** beschreibt, wie über eine Schnittstelle Telegramme sowohl mit 11-Bit als auch mit 29-Bit Identifiern empfangen bzw. versendet werden können.
- Nicht alle auf dem Markt erhältlichen CAN-Controller unterstützen 29-Bit-IDs! Befinden sich solche Busteilnehmer im CAN-Netzwerk, wird es bei der Verwendung von 29-Bit-IDs zu Fehlern kommen. Verwenden Sie in diesem Fall 11-Bit-Identifier.

Message-Objekte werden in zwei Typen unterteilt:

- Sendeobjekte zum Verschicken von Daten
- Empfangsobjekte zum Empfangen von Daten

Ein ***Sendeobjekt*** kann *aktiv* oder *passiv* sein:

- Bei einem aktiven Sendeobjekt geben Sie den Zeitpunkt, wann das Message-Objekt Daten sendet, in Ihrem Anwendungsprogramm durch einen Schreibzugriff auf einen zu dem Sendeobjekt geöffneten Kanal selbst vor. In diesem Falle wird ein *Data-Frame* auf dem CAN-Bus verschickt.
- Bei einem passiven Sendeobjekt wird der Zeitpunkt, wann das Message-Objekt Daten sendet, von einem anderen Busteilnehmer, der von Ihrem Anwendungsprogramm Daten per *Remote-Frame* anfordert, bestimmt. Es werden *automatisch* die zu diesem Zeitpunkt gültigen Daten des passiven Sendeobjekts durch Verschicken eines *Data-Frames* auf den Bus gesendet.

Ein ***Empfangsobjekt*** kann *aktiv* oder *passiv* sein:

- Bei einem aktiven Empfangsobjekt geben Sie den Zeitpunkt, wann das Message-Objekt Daten empfängt, in Ihrem Anwendungsprogramm durch einen Lesezugriff auf einen zu dem Empfangsobjekt geöffneten Kanal selbst vor. In diesem Falle werden Daten durch Senden eines *Remote-Frames* von einem anderen Busteilnehmer angefordert.
- Ein passives Empfangsobjekt empfängt Daten, ohne dass Sie diese in ihrem Anwendungsprogramm speziell anfordern. Die Daten stammen von einem anderen Busteilnehmer, der einen *Data-Frame* verschickt hat.

Die Anzahl der Message-Objekte ist dynamisch. Im Initialisierungszustand sind keine Message-Objekte konfiguriert. Die Konfiguration eines Message-Objekts erfolgt beim Öffnen des ersten Kanals mit einem bestimmten Identifier-Wert.

Werden weitere Kanäle zum selben Message-Objekt geöffnet (d.h. in der CPS ist der selbe Wert im Parameter *ulCanID* angegeben), so wird kein neues Message-Objekt angelegt. Es wird lediglich ein neuer Kanal zum bereits konfigurierten Message-Objekt geöffnet. Dabei ist zu beachten, dass die Eigenschaften eines Message-Objekts bei seiner Konfiguration festgelegt werden. Dieses hat zur Folge, dass sich die CPS-Parameter beim Öffnen eines Kanals zu einem bestehenden Message-Objekt nicht von denen bei seiner Konfiguration unterscheiden dürfen.

10.9.1.5. Verwaltung der Message-Puffer eines CAN-Controllers

Die folgende Erläuterung ist nur dann von Interesse, wenn über eine der beiden CAN-Schnittstellen mehr als 14 verschiedene Message-Objekte kommunizieren sollen.

Damit eine Nachricht gesendet oder empfangen werden kann, muss das zugehörige Message-Objekt in einem der Message-Puffer eines CAN-Controllers auf dem Modul eingetragen sein. Jeder CAN-Controller verfügt hierzu über 14 Message-Puffer (Puffer 1 bis 14), die zum Senden und Empfangen genutzt werden können, und einen speziellen Message-Puffer (Puffer 15), der ausschließlich zum Empfangen genutzt werden kann.

Das Eintragen eines Message-Objekts in einen Message-Puffer nimmt eine gewisse Zeit (einige µs) in Anspruch. Wenn ein Message-Objekt zu dem Zeitpunkt, an dem es z.B. einen Sendewunsch vom Anwenderprogramm erhält, bereits in einem Message-Puffer eingetragen ist, entfällt diese Zeit – es wird schneller reagiert.

Der Modul-Device-Treiber ist darauf ausgerichtet, die Puffer möglichst effektiv nutzen zu können. Hierbei hat das richtige Verständnis zur Pufferverwaltung entscheidenden Einfluss auf die Effektivität Ihres Anwendungsprogramms.

Wenn Sie für eine CAN-Schnittstelle des X-CAN-2i maximal 14 Message-Objekte konfigurieren, dann wird im entsprechenden CAN-Controller jedes Message-Objekt dauerhaft in einen dafür **exklusiv** reservierten **Puffer** (Nr. 1 bis 14) eingetragen. Dieser Fall ist die Optimal-Konfiguration: Weder muss zum aktiven Senden oder Anfordern von Daten die ID in einen Puffer eingetragen werden, noch muss zum Empfangen einer Nachricht die zugehörige ID aus einem Puffer gelesen und ausgewertet werden. Weiterhin wird das zugehörige CPU-Modul minimal belastet, da ausschließlich die gewünschten (d.h. die konfigurierten) Nachrichten empfangen werden.

Mehr als 14 Message-Objekte

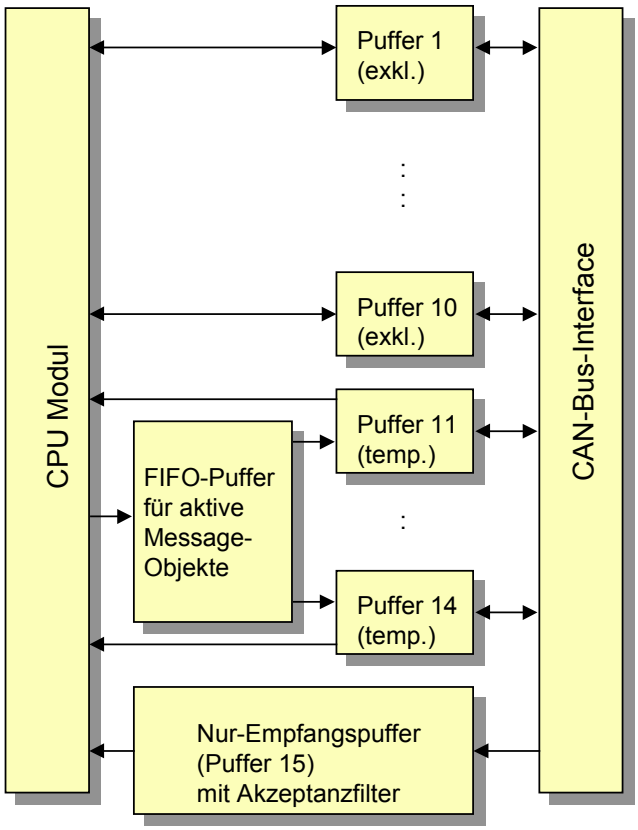
Benötigt eine CAN-Schnittstelle des X-CAN-2i mehr als 14 Message-Objekte, so können zumindest einige davon zwangsläufig nur temporär in einen Puffer des CAN-Controllers eingetragen werden.

Wenn Sie beim Konfigurieren des Message-Objekts das Flag *XCAN_EXCL_ BUFFER* setzen, so zwingen Sie die Pufferverwaltung dazu, einen der Puffer 1 bis 13 abzugeben und ihn exklusiv für dieses Message-Objekt zu reservieren. In diesem Falle wird das Message-Objekt bereits zum Zeitpunkt der Konfiguration dauerhaft in einen der freien Puffer 1 bis 13 eingetragen und der dafür genutzte Puffer exklusiv für dieses Message-Objekt reserviert. Bei einem **exklusiven Puffer** muss weder zum aktiven Senden oder Anfordern von Daten die ID in den Puffer eingetragen werden, noch muss zum Empfangen einer Nachricht die zugehörige ID aus dem Puffer gelesen und ausgewertet werden.

Die von den Puffern 1 bis 13 nicht exklusiv reservierten Puffer sowie Puffer 14 werden von der Pufferverwaltung als **temporäre Puffer** verwendet. Diese Puffer stehen für aktive Message-Objekte zur Verfügung. Ein aktives Empfangsobjekt sendet über einen solchen Puffer einen Remote-Frame, ein aktives Sendeobjekt sendet über einen solchen Puffer einen Data-Frame. Der Modul-Device-Treiber trägt hierzu das Message-Objekt - falls es nicht bereits in einem temporären Puffer gespeichert ist - in einen dieser temporären Puffer ein, sobald es gesendet werden soll. Der so belegte temporäre Puffer ist danach solange nicht verfügbar, bis das gesendete Objekt die Empfangsbestätigung (signalisiert durch Interrupt) erhält. Danach bleibt das Message-Objekt in diesem Puffer solange gespeichert, bis es durch ein anderes Message-Objekt überschrieben wird.

Wenn gerade kein temporärer Puffer verfügbar ist, so wird die Nachricht in einen **FIFO-Puffer** des MDD eingetragen. Wenn einer der temporären Puffer frei wird, so rückt eine Nachricht aus dem FIFO-Puffer nach und wird automatisch gesendet.

Die Abbildung rechts zeigt, wie die Pufferverwaltung des MDD die Message-Puffer nutzt, wenn mehr als 14 Message-Objekte konfiguriert sind und davon 10 Message-Objekte die Eigenschaft X*CAN_EXCL_BUFFER* haben:



Nur-Empfangspuffer

Alle **passiven** Message-Objekte, die nicht in einen exklusiven Puffer eingetragen werden, werden zum Zeitpunkt der Konfiguration in den **Nur-Empfangspuffer** (Puffer 15) eingetragen. Hierbei werden bereits eingetragene Message-Objekte nicht überschrieben. Statt dessen überlagern sich die IDs aller eingetragenen Message-Objekte und bilden ein digitales Akzeptanzfilter, das zur Entlastung des MDD nicht gewünschte Nachrichten per Hardware ausfiltert. Bei den passiven Message-Objekten ist eine durch den CAN-Controller bedingte Einschränkung zu beachten: Alle Message-Objekte, die über den Nur-Empfangspuffer empfangen werden sollen, müssen vom selben Typ sein, d.h. es können darüber entweder nur passive Sendeobjekte oder passive Empfangsobjekte empfangen werden. Die exklusiv genutzten Puffer sind hiervon nicht betroffen. Darin können sowohl passive Sende- als auch Empfangsobjekte eingetragen werden.

Das **digitale Akzeptanzfilter** besteht aus einer ID-Maske und einer Akzeptanzmaske. Beide Masken werden bitweise aus allen in den Nur-Empfangspuffer eingetragenen IDs gebildet. An den Bitpositionen, an denen alle eingetragenen IDs den gleichen Bitwert haben, ist das Bit in der Akzeptanzmaske = 1 und in der ID-Maske gleich dem Bitwert. An den übrigen Bitposition enthalten beide Masken eine 0. Logisch bedeutet dies: Die Akzeptanzmaske entsteht durch eine XNOR-Verknüpfung aller eingetragenen IDs, die ID-Maske entsteht durch eine AND-Verknüpfung aller eingetragenen IDs.

Nur passive Message-Objekte gehen in die Berechnung des Akzeptanzfilters ein!

Eine Nachricht passiert das Akzeptanzfilter nur dann, wenn in der ID der Nachricht an allen Positionen, an denen in der Akzeptanzmaske eine 1 steht, der Bitwert gleich dem Bitwert in der ID-Maske ist.

Beispiel: In den Nur-Empfangspuffer werden die IDs 03h und 07h eingetragen.

IDs:	03h=	0	0	0	0	0	0	0	0	0	1	1
	07h=	0	0	0	0	0	0	0	0	1	1	1
Akzeptanzmaske:		1	1	1	1	1	1	1	1	0	1	1
ID-Maske:		0	0	0	0	0	0	0	0	0	1	1

Es gilt: Akzeptanzmaske = 03h XNOR 07h, ID-Maske = 03h AND 07h. Die IDs, die das Filter passieren können, lassen sich leicht aus der entstehenden ID-Maske herauslesen. Bis auf die Position, die grau unterlegt ist (d.h. das Bit in der Akzeptanzmaske ist = 0), müssen die IDs mit der ID-Maske übereinstimmen. An der grau unterlegten Position spielt der Bitwert der ID keine Rolle. Schlussfolgerung: Das Akzeptanzfilter akzeptiert in diesem Falle tatsächlich nur die IDs 03h und 07h.

Allerdings ist es nicht immer möglich, das Akzeptanzfilter so optimal einzustellen, dass wirklich nur die gewünschten Nachrichten vom CAN-Controller empfangen werden. Bei ungünstiger Vergabe der IDs kann es dazu kommen, dass die Masken des Akzeptanzfilters so ungünstig gebildet werden, dass sehr viele unerwünschte Nachrichten das Filter passieren. Die unerwünschten Nachrichten werden zwar vom MDD verworfen, belasten aber dennoch die CPU!

Beispiel: In den Nur-Empfangspuffer werden die IDs 20h und 07h eingetragen.

IDs:	20h=	0	0	0	0	0	1	0	0	0	0	0
	07h=	0	0	0	0	0	0	0	0	1	1	1
Akzeptanzmaske:		1	1	1	1	1	0	1	1	0	0	0
ID-Maske:		0	0	0	0	0	0	0	0	0	0	0

Im Vergleich zum vorigen Beispiel sind nun mehrere Positionen grau unterlegt (d.h. das Bit in der Akzeptanzmaske ist = 0). Das Filter lässt in diesem Falle die IDs 0h bis 7h und 20h bis 27h passieren. Schlussfolgerung: Das Akzeptanzfilter akzeptiert in diesem Falle außer den gewünschten IDs 20h und 07 auch weitere 14 unerwünschte IDs.

Zusammenfassung

Die Tabelle bietet nochmals eine Übersicht über die unterschiedliche Nutzung der Puffer:

Typ von Message-Objekt	Art der Puffernutzung
Beliebiges Message-Objekt XCAN_EXCL_BUFFER gesetzt	Das Message-Objekt wird zum Zeitpunkt seiner Konfiguration dauerhaft in einen der dafür exklusiv reservierten Puffer eingetragen. Es ist das Senden und Empfangen von Nachrichten, sowohl passiv als auch aktiv, möglich, ohne dass das Message-Objekt vorher nochmals in einen Puffer eingetragen werden muss.
Aktives Message-Objekt XCAN_EXCL_BUFFER nicht gesetzt	Das Message-Objekt wird zum Zeitpunkt, an dem es aktiv Daten sendet oder anfordert, in einen der freien temporären Puffer eingetragen (falls es nicht schon eingetragen ist). Der genutzte Puffer wird vorübergehend gesperrt. Daraufhin wird die Nachricht auf dem CAN-Bus verschickt. Nach der Empfangsbestätigung steht der verwendete temporäre Message-Puffer wieder zur Verfügung.
Passives Sendeobjekt XCAN_EXCL_BUFFER nicht gesetzt	Das Message-Objekt wird zum Zeitpunkt seiner Konfiguration dauerhaft in den Nur-Empfangspuffer eingetragen. Zum Zeitpunkt, an dem es aktiv Daten sendet oder anfordert, wird es in einen der temporären Message-Puffer eingetragen (falls es nicht schon eingetragen ist). Daraufhin wird die Nachricht auf dem CAN-Bus verschickt.
Passives Empfangsobjekt XCAN_EXCL_BUFFER nicht gesetzt	Das Message-Objekt wird zum Zeitpunkt seiner Konfiguration dauerhaft in den Nur-Empfangspuffer eingetragen und wartet darauf, bis es eine Nachricht erhält.

- Um eine optimale Bearbeitungsgeschwindigkeit zu erreichen (d.h. die Anzahl und die Dauer der Zugriffe auf das Modul zu minimieren), legen Sie eine optimale Anzahl an exklusiven Puffern fest und weisen den gewünschten Message-Objekten die Eigenschaft *XCAN_EXCL_BUFFER* zu. Dies sollte für Message-Objekte gelten, die sich durch einen besonders häufigen Zugriff auf den CAN-Bus auszeichnen, oder die sehr zeitkritisch sind.

- Gibt es eine sehr hohe Zahl an Message-Objekten, die alle gleich häufig an der CAN-Bus-Kommunikation teilnehmen, kann es sinnvoll sein, keine exklusiven Puffer zu verwenden, damit der MDD die Message-Objekte möglichst gleichmäßig auf die 14 Puffer verteilen kann. Würden sich alle Message-Objekte z.B. nur einen temporären Puffer teilen, müsste dieser praktisch vor jeder Kommunikation neu mit dem jeweiligen Message-Objekt geladen werden.

- Achten Sie bei der Vergabe der IDs für die Message-Objekte, die in den Nur-Empfangspuffer eingetragen werden, darauf, dass nicht zu viele unerwünschte Nachrichten das Akzeptanzfilter des Nur-Empfangspuffers passieren. Dies erreichen Sie, in dem Sie IDs verwenden, die möglichst ähnliche Bitmuster aufweisen.

- Wenn sie ein Message-Objekt als passives Sendeobjekt konfigurieren, so sollten Sie die Eigenschaft *XCAN_EXCL_BUFFER* zuweisen.

10.9.2. Modul Device Treiber

10.9.2.1. Installation

Der Modul-Device-Treiber für OsX hat die Programmnummer 803Ah und den Dateinamen mxcan2.exe. Der Modul-Device-Treiber für Windows hat den Namen mxcan2.sys. Die Installation aus einem PC-Programm (z.B. für Steckplatz 1, Layer 0):

Error = max_load_mdd (hModul, 1, 0, 0, 0x803A, NULL, &hMDD);

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

MAXLOADMDD slot=1 layer=0 progno=803A

10.9.2.2. Kanaleigenschaftsstruktur CPS_XCAN

Die CPS für das Modul hat den Namen CPS_XCAN.

10.9.2.3. Bus-Steuerung

Das Bus-Steuerungs-Device erlaubt die Einstellung der Bit-Timing-Parameter, das Abschalten einer CAN-Bus-Schnittstelle von der Bus-Kommunikation sowie die Festlegung einer Fehlerbehandlungs-Funktion.

Ein solches Objekt wird mit folgender CPS erzeugt bzw. ein Kanal zu einem solchen Objekt geöffnet:

Strukturelement	**Werte**	**Bedeutung**
.usDevice	*DEVICE_CTRL*	Steuerkanal
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.usIndexFirst	*XCAN_BUS_CTRL_INDEX*	Kanal-Nummer
.usIndexLast	*= .usIndexFirst*	Kanal-Nummer
.usFlags	*0*	Keine Bedeutung
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.

Eingabe- und Ausgabedienst

Mit dem Ausgabedienst wird der CAN-Bus parametriert. Die Bit-Timing-Parameter, der Busabschlußwiderstand und das ID-Format werden für die betreffende Schnittstelle festgelegt. Der Datentyp ist DATA_USHORT. Der Kanal ist rücklesbar.

- **max_write_channel_block**
- **max_read_channel_block** (Zurücklesen der eingestellten Daten)

Eine folgende Datenstruktur vom Typ XCAN_BITPARAMS wird von den Kanaldiensten erwartet:

Name	Typ	Bereich	Bedeutung
Tq	USHORT	125-8000	Time-Quantum in ns (Vielfaches von 125)
Tseg1	USHORT	3-16	Zeit tseg1/Tq vor dem nominellen Abtastzeitpunkt
Tseg2	USHORT	2-8	Zeit tseg2/Tq nach dem nominellen Abtastzeitpunkt
Tsjw	USHORT	1-4	Synchronisationssprungweite tsjw/Tq
SpB	USHORT	1 oder 3	Zahl der Abtastungen pro Bit
Rbus[1]	USHORT	0	Busabschlusswiderstand ausgeschaltet (für High Speed CAN-Bus-Schnittstelle) bzw. Busabschlusswiderstände 5,6 kΩ (für Low-Speed CAN-Bus-Schnittstelle)
		1	Busabschlusswiderstand eingeschaltet (für High Speed CAN-Bus-Schnittstelle) bzw. Busabschlusswiderstände 510 Ω (für Low-Speed CAN-Bus-Schnittstelle)
MsgFormat	USHORT	11 29	Nur 11-Bit CAN-Identifier werden verwendet Nur 29-Bit CAN-Identifier werden verwendet
		XCAN_ MIXED_ FORMAT	Sowohl 11-Bit als auch 29-Bit Identifier können auf der Schnittstelle verwendet werden (s. Application Note AN099).

[1] Bei fehlertoleranten CAN-Bus-Systemen muss der CAN-Bus mit einem Busabschlusswiderstand von 100 Ohm abgeschlossen werden. Dieser ist dabei über das gesamte Netzwerk verteilt, wobei jeder CAN-Knoten nur einen Teil des Gesamtwiderstandes enthält. Der Busabschlusswiderstand berechnet sich aus der Parallelschaltung der Widerstände aller Knoten. Die Widerstände an den einzelnen Bus-Teilnehmern können verschieden sein.
Auf dem X-CAN-2 stehen für die fehlertoleranten Schnittstellen die beiden Widerstandswerte 5,6 kOhm (für Systeme mit vielen Teilnehmern) und 510 Ohm (für Systeme mit wenigen Teilnehmern) zur Auswahl. Dass in Systemen mit weniger als 5 Teilnehmern der optimale Gesamtwiderstand von 100 Ohm nicht erreicht werden kann, stört in Systemen mit geringer Ausdehnung in der Regel nicht. Details dazu finden sich z.B. in den „Application Hints Fault-Tolerant CAN-Transceiver“ von Philips Semiconductors.

Sonderdienste

- **max_channel_control,** Steuerbefehle CMD_START, CMD_STOP: CAN-Bus ein- bzw. ausschalten. Es werden keine Daten übergeben.
- **max_channel_info,** INFO_DEVICE: Liefert den Zustand des CAN-Bus als ULONG (Kodierung s.u.).

Nach dem Öffnen des ersten Kanals zu einem Sende- oder Empfangsobjekt ist der Bus standardmäßig eingeschaltet.

Callback-Funktion = CAN-Bus-Fehlerbehandlungs-Funktion

Durch die Angabe einer Callback-Funktion beim Öffnen eines Kanals zum Bus-Control-Device kann eine Fehlerbehandlungsfunktion installiert werden, die vom MDD automatisch aufgerufen wird, wenn der CAN-Controller einen Fehler signalisiert.

Die Fehlerbehandlungsfunktion bekommt bei ihrem Aufruf einen ULONG-Parameter übergeben. Die Bedeutung dieses Parameters ist wie folgt:

Im Low-Word wird der Zustand des CAN-Bus angegeben. Die folgenden Werte können dabei ODER-verknüpft sein:

- XCAN_BUS_ON Normalbetrieb
- XCAN_BUS_HEAVY Stark gestörter Bus
- XCAN_BUS_OFF Aufgrund schwerer Busstörungen hat der CAN-Controller sich automatisch vom CAN-Bus zurückgezogen
- XCAN_LINE_ERROR Vom Transceiver signalisierter Leitungsfehler (nur für Low-Speed Schnittstelle)

Im High-Word ist nähere Information über den zuletzt aufgetretenen Fehler codiert:

Wert	Bedeutung
1	Stuff Error (mehr als 5 gleiche Bits in einem Telegramm hintereinander)
2	Form Error (Format-Fehler im Telegramm)
3	Ack Error (Gesendetes Telegramm wurde von keinem Teilnehmer quittiert)
4, 5	Bit Error (beim Senden eines Bits wurde der falsche Wert zurückgelesen)
6	CRC-Error (fehlerhafte Checksumme)
übrige	reserviert

Die Fehlererkennung und –signalisierung wird vollständig vom CAN-Controller übernommen. Dazu besitzt jeder CAN-Controller einen Sende- und einen Empfangsfehlerzähler, die er beim Auftreten von Fehlern um einen vom Fehlertyp abhängigen Betrag inkrementiert. Nach jeder erfolgreichen Übertragung wird der Zähler um einen bestimmten Betrag dekrementiert.

In Abhängigkeit dieser beiden Zähler nimmt der Controller einen der drei folgenden Zustände ein:

- Beide Zähler < 128 (Betriebsart 'Fehleraktiv'): Die CAN-Schnittstelle nimmt voll an der Bus-Kommunikation teil und sendet bei der Erkennung von Fehlern automatisch einen aktiven Error-Frame. Die als fehlerhaft erkannte Nachricht wird hierbei durch gezieltes Verletzen des Nachrichtenprotokolls zerstört.
- Einer der beiden Zähler überschreitet den Wert 128 (Betriebsart 'Fehlerpassiv'): Die CAN-Schnittstelle nimmt weiter voll an der Bus-Kommunikation teil, sendet nun aber bei der Erkennung von Fehlern einen passiven Error-Frame, der das Nachrichtenprotokoll nicht zerstört.
- Einer der beiden Zähler überschreitet den Wert 256 (Betriebsart 'Bus Off'): Die CAN-Schnittstelle wird von der Bus-Kommunikation abgeschaltet. Diese Betriebsart kann nur durch die Sonderdienste CMD_STOP und anschließend CMD_START (s.o.) verlassen werden. In diesem Falle werden beide Fehlerzähler wieder zurückgesetzt.

Der CAN-Controller schaltet zwischen den Betriebsarten 'Fehleraktiv' und 'Fehlerpassiv' selbstständig hin und her.

Ein Fehlerstand > 96 wird als 'Stark gestörter Bus' interpretiert und genau wie der Übergang in die Betriebsart 'Bus-Off' mit einem Interrupt signalisiert. In diesen beiden Fällen wird die angegebene Fehler-Funktion aufgerufen.

10.9.2.4. Aktive Sendeobjekte

Aktive Sendeobjekte versenden ihre Daten durch einen (aktiven) Aufruf des Kanal-Ausgabedienstes durch das Anwenderprogramm. Das Datentelegramm wird verschickt, sobald der Bus frei ist. Ein solches Objekt wird mit folgender CPS erzeugt bzw. ein Kanal zu einem solchen Objekt geöffnet:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_BUS_OUT*	Kanal zu einem Sendeobjekt
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.ulCanID	*0 bis 7F0h*	CAN-Bus Identifier für 11-Bit- Identifier
	0 bis 1FFFFFF0h	CAN-Bus Identifier für 29-Bit- Identifier
.usFlags	*_CP_SYNC_CALLBACK*	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
	XCAN_NO_INT	Die Callback-Funktion wird nicht aufgerufen. Statt dessen wird solange im Ausgabedienst gewartet, bis das Telegramm auf den Bus versendet werden kann (max. die durch *usTimeout* gegebene Zeit).
	XCAN_EXCL_BUFFER	Kann zusätzlich gesetzt werden, um einen Puffer des CAN-Controllers für das Objekt zu reservieren.
.usTimeout	*0 bis 65535*	Zeit (in µs), die max. vergehen darf, bis ein Sendewunsch tatsächlich auf den Bus abgesetzt werden kann.

Eingabe- und Ausgabedienst

Der Datentyp ist DATA_UCHAR. Ein Block besteht aus 0 bis 8 Bytes.

- **max_write_channel_block**
- **max_read_channel_block** (Zurücklesen der Daten)

Sonderdienste

- **max_channel_control,** Steuerbefehl CMD_START: die zuvor versendeten Daten nochmals senden. Es werden keine Daten übergeben.
- **max_channel_info** INFO_XCAN_MSG_COUNT: liefert einen Zählerwert (als ULONG). Dieser gibt an, wie viele Telegramme für das Messageobjekt insgesamt versendet worden sind (ab MDD Version 3.C).

Callback-Funktion

Sofern im Element *usFlags* der CPS das Flag *XCAN_NO_INT* nicht gesetzt ist, und beim Öffnen des Kanals eine Callback-Funktion angegeben wurde, wird diese aufgerufen, wenn das Telegramm erfolgreich versendet werden konnte. Der Callback-Funktion werden keine Daten übergeben.

10.9.2.5. Passive Sendeobjekte

Passive Sendeobjekte verschicken ihre Daten nur auf Anforderung (über den CAN-Bus) durch einen anderen Teilnehmer. Sobald ein von einem anderen Teilnehmer gesendeter Remote-Frame empfangen wird, werden die zu dem Zeitpunkt gültigen Daten versendet. Ein Schreibzugriff auf den Kanal stellt dem Sendeobjekt neue Daten zur Verfügung, ohne dass die Daten unmittelbar auf den Bus versendet werden.
Ein solches Objekt wird mit folgender CPS erzeugt bzw. ein Kanal zu einem solchen Objekt geöffnet:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_BUS_OUT*	Kanal zu einem Sendeobjekt
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.ulCanID	*0 bis 7F0h*	CAN-Bus Identifier für 11-Bit- Identifier
	0 bis 1FFFFFF0h	CAN-Bus Identifier für 29-Bit- Identifier
.usFlags	*XCAN_REMOTE*	Muss gesetzt sein.
	XCAN_NO_INT	Das Versenden der Daten erzeugt keinen Interrupt. Die Callback-Funktion wird nicht aufgerufen.
	XCAN_EXCL_BUFFER	Kann zusätzlich gesetzt werden, um einen Puffer des CAN-Controllers für das Objekt zu reservieren.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usTimeout	*0*	reserviert

Eingabe- und Ausgabedienst

Der Datentyp ist DATA_UCHAR. Ein Block besteht aus 0 bis 8 Bytes.

- **max_write_channel_block**
- **max_read_channel_block** (Zurücklesen der Daten)

Sonderdienste

- **max_channel_control,** Steuerbefehl CMD_START: die gerade aktuellen Daten aktiv versenden. Es werden keine Daten übergeben.
- **max_channel_info** INFO_XCAN_MSG_COUNT: liefert einen Zählerwert (als ULONG). Dieser gibt an, wie viele Telegramme für das Messageobjekt insgesamt versendet worden sind (ab MDD Version 3.C).

Callback-Funktion

Sofern im Element *usFlags* der CPS das Flag *XCAN_NO_INT* nicht gesetzt ist, und beim Öffnen des Kanals eine Callback-Funktion angegeben wurde, wird diese aufgerufen, wenn das Telegramm erfolgreich versendet werden konnte. Der Callback-Funktion werden keine Daten übergeben.

10.9.2.6. Aktive Empfangsobjekte

Durch einen Lesezugriff auf ein aktives Empfangsobjekt wird (aktiv) ein Remote-Frame versendet, um von einem anderen Busteilnehmer Daten anzufordern. Im Kanal-Eingabedienst wird solange gewartet, bis die Antwort vom anderen Busteilnehmer eintrifft, jedoch maximal nur die im Parameter *usTimeout* angegebene Zeit. Es wird kein Interrupt ausgelöst.

Ein solches Objekt wird mit folgender CPS erzeugt bzw. ein Kanal zu einem solchen Objekt geöffnet:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_BUS_IN*	Kanal zu einem Empfangsobjekt
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.ulCanID	*0 bis 7F0h*	CAN-Bus Identifier für 11-Bit- Identifier
	0 bis 1FFFFFF0h	CAN-Bus Identifier für 29-Bit- Identifier
.usFlags	*XCAN_REMOTE* + *XCAN_NO_INT*	Müssen gesetzt sein
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
	XCAN_EXCL_BUFFER	Kann zusätzlich gesetzt werden, um einen Puffer des CAN-Controllers für das Objekt zu reservieren.
.usTimeout	*0 bis 65535*	Zeit (in µs), die max. vergehen darf, bis die Antwort empfangen wird.

Eingabedienst

Der Datentyp ist DATA_UCHAR. Ein Block besteht aus 0 bis 8 Bytes.

- **max_read_channel_block**

Sonderdienst

- **max_channel_control,** Steuerbefehl CMD_START: Remote-Telegramm versenden. In diesem Fall wird beim Eintreffen der Antwort die Callback-Funktion aufgerufen. Es werden keine Daten übergeben.

Callback-Funktion

Wenn eine Callback-Funktion angegeben wurde, wird diese nur aufgerufen, wenn über den o.g. Sonderdienst Daten angefordert werden. Die empfangenen Daten (bis zu 8 Bytes) werden der Callback-Funktion übergeben.

10.9.2.7. Passive Empfangsobjekte

Passive Empfangsobjekte empfangen Daten vom Bus nur, wenn ein anderer Teilnehmer ihnen etwas sendet. Sie lösen selbst keine Aktivität auf dem CAN-Bus aus.

Ein solches Objekt wird mit folgender CPS erzeugt bzw. ein Kanal zu einem solchen Objekt geöffnet:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_BUS_IN*	Kanal zu einem Empfangsobjekt
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.ulCanID	*0 bis 7F0h*	CAN-Bus Identifier für 11-Bit- Identifier
	0 bis 1FFFFFF0h	CAN-Bus Identifier für 29-Bit- Identifier
.usFlags	*_CP_SYNC_CALLBACK*	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
	XCAN_EXCL_BUFFER	Kann zusätzlich gesetzt werden, um einen Puffer des CAN-Controllers für das Objekt zu reservieren.
	XCAN_NEW_ONLY	Ist dieses Flag gesetzt, liefert der Dateneingabedienst nur dann einen Wert zurück, wenn seit seinem letzten Aufruf neue Daten für das Empfangsobjekt vom CAN-Bus empfangen worden sind. Dieses ist für den Betrieb ohne Callbacks von Interesse. Liegen keine neuen Daten vor, liefert *max_read_channel_block* den Fehler ERR_DATA_NOT_READY. Dieser muss mit max_clear_error zurückgesetzt werden. Das Flag steht erst ab MDD Version 3.C zur Verfügung.
.usTimeout	*0*	reserviert

Eingabedienst

Die gerade aktuellen Daten eines Empfangsobjekts können jederzeit durch einen Lesezugriff gelesen werden.

Der Datentyp ist DATA_UCHAR. Ein Block besteht aus 0 bis 8 Bytes.

- **max_read_channel_block**

Sonderdienste

- **max_channel_control,** Steuerbefehl CMD_START: Remote-Telegramm für ein passives Empfangsobjekt versenden (dadurch wird es zum sowohl aktiven als auch passiven Empfangsobjekt). Es werden keine Daten übergeben.
- **max_channel_info** INFO_XCAN_MSG_COUNT: liefert einen Zählerwert (als ULONG). Dieser gibt an, wie viele Telegramme für das Messageobjekt insgesamt empfangen worden sind.

Callback-Funktion

Der Empfang der Daten wird dem Anwenderprogramm durch den Aufruf der Callback-Prozedur mitgeteilt, sofern diese beim Öffnen des Kanals angegeben wurde. Dabei werden die empfangenen Daten (max. 8 Bytes) der Callback-Funktion übergeben.

10.9.2.8. Akzeptanzfilter für den Nur-Empfangspuffer

Das Akzeptanzfilter dient dazu, eine Anzahl von CAN-Telegrammen mit unterschiedlichen Identifiern zu empfangen, ohne dass für jeden Identifier ein eigenes Message-Objekt angelegt werden muss.

Soweit Message-Objekte nicht in eigenen Puffern des CAN-Controllers eingetragen sind, geschieht ihr Empfang über den Nur-Empfangspuffer (s. Kap. 10.9.1.5). Für diesen Message-Puffer gibt es außer dem Identifier-Wert noch einen Akzeptanzfilter. Dieses bestimmt, welche Bits der empfangenen ID exakt mit dem abgespeicherten ID-Wert übereinstimmen müssen und welche nicht. Alle konfigurierten Message-Objekte, die über den Nur-Empfangspuffer empfangen werden sollen, gehen automatisch in die Bildung des Wertes des Akzeptanzfilters ein.

Über das Akzeptanzfilter-Device ist es möglich, das Filter manuell zu verändern, so dass das Modul ganze Gruppen von Message-Objekten oder sogar alle Message-Objekte empfangen kann, ohne die Message-Objekte einzeln zu konfigurieren. Dazu ist ein Kanal mit folgender CPS zu öffnen:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_CTRL*	Steuerkanal
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.usIndexFirst	*0 bis 10 bzw.*	Offset des niederwertigsten Bits im 11 bzw.
	0 bis 28 [1]	29 Bit umfassenden Akzeptanzfilter
.usIndexLast	*0 bis 10 bzw.*	Offset des höchstwertigsten Bits im 11 bzw.
	0 bis 28	29 Bit umfassenden Akzeptanzfilter
.ulCanID	*0 bis 7F0h* *0 bis 1FFFFFF0h*	Identifier für den Nur-Empfangspuffer. Dieser Wert ist nur dann wichtig, wenn die über den Ausgabedienst gesetzte Filtermaske nicht 0 ist.
	XCAN_AUTO_ID	Dieser Wert bewirkt, dass der aus allen nicht-exklusiven passiven Message-Objekten automatisch gebildete Identifier für den Nur-Empfangspuffer nicht durch den Akzeptanzfilter-Kanal verändert wird.
.usFlags	*XCAN_REMOTE*	Nur-Empfangspuffer empfängt nur Remote-Frames; wenn das Flag nicht gesetzt ist, empfängt er nur Data-Frames.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_ULONG.

- **max_write_channel_ulong**
- **max_read_channel_ulong** (zum Zurücklesen der eingestellten Maske)

Der geschriebene Wert stellt die Bitmaske für das Akzeptanzfilter dar. An allen Stellen, an denen die Bitmaske =1 ist, wird der Identifier eines empfangenen Telegramms mit dem durch *ulCanID* gegebenen Identifier-Wert verglichen. Das Telegramm wird nur dann empfangen, wenn der Identifier an allen Bit-Positionen, an denen die Bitmaske =1 gesetzt ist, mit dem durch *ulCanID* übergebenen übereinstimmt. Die Bits, an denen die Bitmaske =0 ist, werden nicht verglichen. Sind alle Bits der Bitmaske =0, bedeutet dies folglich, dass alle Identifier-Werte über den Nur-Empfangspuffer empfangen werden.

Passive Message-Objekte, bei denen *XCAN_EXCL_BUFFER* nicht gesetzt ist, werden ebenfalls über den Nur-Empfangspuffer empfangen. Ihre Identifier werden auto-

[1] Zur Zeit muss *usIndexFirst* = *usIndexLast* = 0 gesetzt werden. D.h. das Akzeptanzfilter kann nur als Ganzes verändert werden.

matisch für die Bildung des Identifiers und der Bitmaske des Akzeptanzfilters herangezogen (s. Kap. 10.9.1.5). Wird dann ein Akzeptanzfilter-Kanal geöffnet, gilt die Einschränkung, dass in der automatisch erzeugten Akzeptanzmaske niemals ein 0 gesetztes Bit über das Akzeptanzfilter-Device auf 1 gesetzt werden darf. Ansonsten würden konfigurierte Message-Objekte nicht mehr durch das Akzeptanzfilter gelangen. Der Versuch wird vom MDD mit einer Fehlermeldung quittiert. Im Fall, dass passive Message-Objekte, bei denen *XCAN_EXCL_BUFFER* nicht gesetzt ist, bereits konfiguriert sind, muss *ulCanID=XCAN_AUTO_ID* gesetzt werden. Der Identifier-Wert wird dann durch das Öffnen eines Kanals zum Akzeptanzfilter-Device nicht verändert.

Im Initialisierungszustand (d.h. es sind keine CAN-Kanäle geöffnet und die Akzeptanzmaske ist noch nicht mit Hilfe des Akzeptanzfilter-Devices manuell gesetzt worden), ist der Wert der Akzeptanzmaske = FFFFFFFFh.

Callback-Funktion

Die Callback-Funktion wird immer dann aufgerufen, wenn eine unkonfigurierte Nachricht empfangen wird, d.h. das Akzeptanzfilter passiert hat.

Die Callback-Prozedur bekommt die folgende Datenstruktur vom Typ XCAN_MESSAGE übergeben:

Name	**Typ**	**Bedeutung**
ulCanId	ULONG	Identifier
usDataSize	USHORT	Anzahl der im Array aucData eingetragenen gültigen Daten. Bei der Konfiguration des Nur-Empfangspuffers für den Empfang von Remote-Frames wird hier 0 eingetragen.
usFlags	USHORT	reserviert
aucData	UCHAR	Array aus 8 Byte, in das die empfangenen Daten eingetragen werden.

10.9.2.9. Direktes Versenden von CAN-Telegrammen

Als Gegenstück zum Telegrammempfang ohne konfigurierte Message-Objekte erlaubt es der folgende Kanal CAN-Bus-Telegramme zu versenden, ohne vorher ein entsprechendes Messageobjekt zu konfigurieren (dieser Kanaltyp ist ab **MDD Version 3.C** implementiert). Das Öffnen dieses Kanals belegt einen exklusiven Puffer. Folgende CPS ist zu verwenden:

Strukturelement	Wert	Bedeutung
.usDevice	*DEVICE_BUS_OUT*	Device-Typ
.usChannel	*0 oder 1*	CAN-Schnittstelle auf dem Modul
.ulCanID	*XCAN_GENERAL_ID*	Der CAN-Identifier wird beim Aufruf des Ausgabedienstes mit übergeben. Das Öffnen des Kanals konfiguriert also kein Messageobjekt.
.usFlags	*0*	Der Ausgabedienst trägt die zu sendenden Daten in einen freien Puffer des CAN-Controllers oder den FIFO-Puffer ein. Von dort wird das Telegramm versendet, sobald der Bus frei ist.
	_CP_EXCLUSIVE	Muss gesetzt sein, pro Schnittstelle darf nur ein Kanal mit *ulCanID=XCAN_GENERAL_ID* geöffnet werden.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
	_XCAN_NO_INT	Die Callback-Funktion wird nicht aufgerufen. Statt dessen wird solange im Ausgabedienst gewartet, bis das Telegramm auf den Bus versendet werden kann (max. die durch .usTimeout gegebene Zeit).
.usTimeout	*1 ... 65535*	Zeit (in µs), die max. vergehen darf, bis der Sendewunsch tatsächlich auf den Bus versendet werden kann.

Ausgabedienst

Der Datentyp ist DATA_UCHAR. Dem Ausgabedienst wird eine Datenstruktur vom Typ XCAN_MESSAGE übergeben.

- **max_write_channel_block (hChan, &ulSize, &arcData)**

Die Datenstruktur XCAN_MESSAGE hat folgenden Aufbau:

Name	Typ	Bedeutung
ulCanId	ULONG	Identifier
usData-Size	USHORT	Anzahl der im Array *aucData* eingetragenen gültigen Daten. Bei der Konfiguration des Nur-Empfangspuffers für den Empfang von Remote-Frames wird hier 0 eingetragen.
usFlags	USHORT	Eigenschaften des zu sendenden Telegramms. Bisher sind folgende Flags definiert: XCAN_29BIT_ID: Der Identifier wird im 29-Bit Format versendet, ansonsten im 11-Bit Format. XCAN_REMOTE: Es wird ein Remote-Telegramm versendet. Für den Empfang der Antwort muss ein Kanal zum Akzeptanzfilter-Device mit entsprechend gesetztem Empfangsfilter offen sein, über den die Antwort empfangen wird. Ansonsten wird ein Data-Telegramm versendet.
aucData	UCHAR	Array aus 8 Byte, in das die zu sendenden Daten eingetragen werden.

10.9.3. Pinbelegung

Signal	Modulstecker A	D-SUB-9 Stecker
CAN-Bus 1, GND	1, 21	6
CAN-Bus 1, CAN-L	2, 22	2
CAN-Bus 1, CAN-H	3, 23	7
CAN-Bus 1, GND	4, 24	3
CAN-Bus 1, 12V out [1]	5, 25	8
CAN-Bus 1, 7-24V in [1]	6, 26	4
CAN-Bus 1, 5V out	7, 27	9
CAN-Bus 1, Schirm [1]	8, 28	5
n.c.	9, 29	-
n.c.	10, 30	-
CAN-Bus 2, GND	11, 31	6
CAN-Bus 2, CAN-L	12, 32	2
CAN-Bus 2, CAN-H	13, 33	7
CAN-Bus 2, GND	14, 34	3
CAN-Bus 2, 12V out [1]	15, 35	8
CAN-Bus 2, 7-24V in [1]	16, 36	4
CAN-Bus 2, 5V out	17, 37	9
CAN-Bus 2, Schirm [1]	18, 38	5
n.c.	19, 39	-
n.c.	20, 40	-

[1] Die Pins <12V out>, <7-24V in> und <Schirm> sind nur in speziellen Bestückungsvarianten verfügbar, die erst bei Abnahme einer bestimmten Stückzahl möglich sind. Standardmäßig sind die Pins unbeschaltet.

Die Pinbelegung ist so ausgeführt, dass ein D-SUB-9 Stecker, der mit Modulstecker A verbunden ist, die CiA-Empfehlung 102 einhält. Dabei ist Pin 1 des D-SUB-9 Steckers nicht belegt. Bei der Variante X-CAN-2i/M ist Bus 1 die High-Speed-Schnittstelle und Bus 2 die Low-Speed-Schnittstelle.

10.9.4. Besondere Eigenschaften

Parameter	**Wert**	**Einheit**
CAN-Controller	INTEL 82527 (2 Stück)	-
CAN-Transceiver		
High Speed (X-CAN-2i/H)	Philips 82C250 o.ä.	-
Low Speed (X-CAN-2i/F)	Philips TJA 1054	-
Einstellbare Bitrate		
High Speed	5 bis 1000	kBit/s
Low Speed	5 bis 125	kBit/s
Busabschlusswiderstand, schaltbar		
High Speed typ. (±10%)	110	Ω
Low Speed typ.	510 oder 5600[1]	Ω
Trennspannung	500	V rms
Ausgänge Versorgungsspannung (5V out)		
Spannung min./typ./max.	4,7/5/5,1	V
Strom max. bei funktionierender Bus Kommunikation	15	mA
Temperatur-Bereich, Betrieb	0 bis 70	°C
optional	-40 bis +85	°C
Abmessungen	29x58x8	mm
Gewicht	10	g
Stromaufnahme (3,3V)		
High Speed typ. (X-CAN-2i/H)	510	mA
Low Speed typ. (X-CAN-2i/F)	410	mA
(die X-Bus Spannungen ±12V werden nicht benötigt)		

[1] An beiden CAN-Bus-Leitungen sind Busabschlusswiderstände von je 5,6 kΩ vorhanden. Durch das Einschalten wird ein Widerstand von je 560 Ω dazu parallel geschaltet. Für Netzwerke mit wenig Teilnehmern sollte der niedrige Widerstand gewählt werden, in großen Netzwerken ein hoher Busabschlusswiderstand.
Die CAN-L Leitung wird immer nach VCC (5V) terminiert.