CAN-TALK
Monitor Protokoll für CAN Bus

Protokollbeschreibung

Klassifikation: Nur für NTI internen Gebrauch

Ablage: \UserProducts\Sw\LinMon

Änderungsnachweis:

Dokument	Datum	Autor	Status Bemerkungen
TALK-CAN.DOC	12.6.1996	eb, Rei	V0.5 (preliminary)
Dokumentation_CAN-Talk.doc	16.5.1997	Rei	V1.0
DevDoc_CAN-Talk.doc	13.5.2002	Ro	V1.1

Table of Contents

1 Einleitung

1.1 Überblick

Das CAN-TALK Protokoll wurde an der Abteilung NTI am Technopark zur Kommunikation von einem Bus-Master (Host PC) zu einem oder mehren Slave Prozessor Systemen (Zielsysteme) entwickelt. Das Protokoll ermöglicht sowohl die Steuerung der Zielsysteme via Bus-Master, wie auch den Datenaustausch zwischen dem Bus-Master und den Zielsystemen.

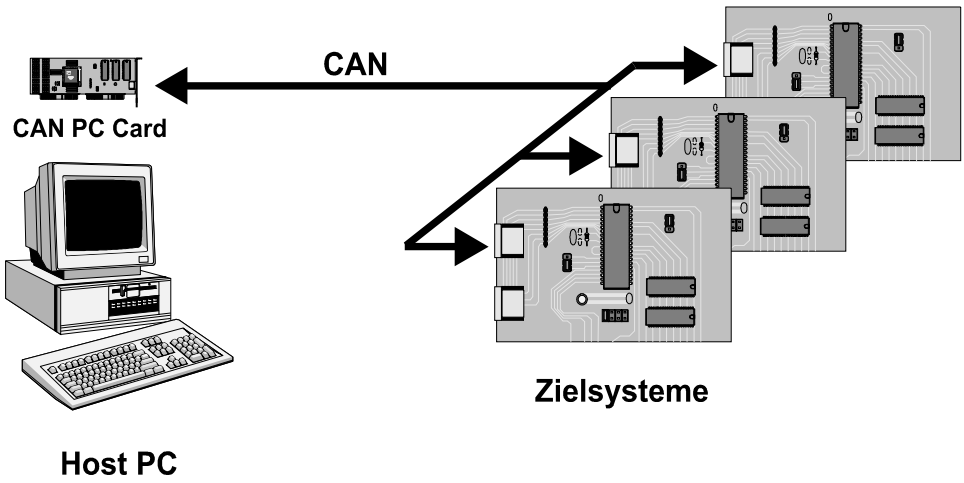

Abb. 1.1: Kommunikation mit Host PC

Für den CAN Bus wird eine CAN-Schnittstellenkarte im Host PC benötigt. Damit lässt sich mittels CAN-TALK Protokoll auf einfachste Art mit einem oder mehreren Zielsystemen kommunizieren.

1.2 Ziele und Vorgaben

Das Protokoll soll folgende Vorgaben erfüllen:

- Datenaustausch über die CAN Schnittstelle des Host-PC in beide Richtungen, wobei ein Datentransfer nur vom Master (Host) PC ausgelöst werden kann.
- Die Zielsysteme sollen die Möglichkeit haben, spontane Meldungen an den Bus-Master zu senden.
- Das Protokoll soll nicht nur den Datenaustausch zwischen Master (Host PC) und Zielsystem zulassen, sondern gleichzeitig die Steuerung des Zielsystems ermöglichen
- Übertragung von einzelnen Datenwörtern (16 Bit) von bzw. auf beliebige Adressen des Zielsystems.
- Übertragung von grösseren Datenblöcken für das Laden von Programmen und Daten (z.B. Tabellen, Sollwertkurven, etc.)
- Das Protokoll soll eine möglichst einfache Auswertung auf dem Host PC und dem Zielsystem ermöglichen, um die Rechnerbelastung so tief wie möglich zu halten.

Eine sehr hohe Störsicherheit wird durch den redundanten Aufbau der CAN-Nachrichten bereits durch die Hardware gewährleistet.

1.3 Referenzen

Ref	Titel	Autor	Datum	ID	Bezugsquelle
[1]	Selectron MAS-DP Systemhandbuch SeleCAN		19.09.96	393.0060	Selectron AG CH-3250 Lyss
[2]	RS232-Talk				
[3]					
[4]					

1.4 Definitionen, Begriffe, Abkürzungen

Kurzzeichen	Bedeutung

2 CAN Schnittstelle

2.1 Schnittstellen Spezifikation

Die CAN Schnittstelle für die Datenübertragung zwischen Host PC und den Zielsystemen ist wie folgt spezifiziert:

Baud Rate: Max: 1 MBaud

Datenformat: CAN 2.0A (11 Bit Identifier)

1	Start Bit
11	Identifier Bits
1	RTR Bit
1	IDE Bit
1	r0 Bit
4	DLC Bits
0…8*8	0 bis 8 Data Byte
15	CRC Bits
2	ACK Bits
10	EOF + IFS Bits

3 Aufbau einer CAN Meldung

3.1 Aufbau des CAN-Identifiers

Der CAN-Identifier besteht aus 11 Bits. Im CAN-Identifier wird die Adresse des CAN-Knotens sowie die Priorität einer Nachricht bestimmt. Der Identifier für das CAN-Talk Protokoll wurde folgendermassen spezifiziert:

ID 10	ID 9	ID 8	ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0
Res	Dir	Adr 5	Adr 4	Adr 3	Adr 2	Adr 1	Adr 0	Spez 2	Spez 1	Spez 0

Res — Das erste Bit des Identifiers ist Reserviert für zukünftige Anwendungen und Erweiterungen des CAN-Talk Protokolls. Solange dieses Bit nicht gebraucht wird, muss es den Wert 1 haben. Damit bleibt die Möglichkeit offen höher priorisierte Botschaften zu versenden.

Dir — Das Direction-Bit bestimmt die Richtung einer CAN-Meldung:
0 Master - Slave Meldung (hohe Priorität)
1 Slave - Master Meldung (tiefe Priorität)

Adr 0 - Adr 5 — Die fünf Adressbits bestimmen die Adresse des Senders bzw. Empfängers (siehe Kapitel 3.3). Mit diesen fünf Bits lassen sich bis zu 64 Knoten auf einem CAN-Bus adressieren.

Spez 0 - Spez 2 — Diese drei Bits haben im CAN-Talk Protokoll immer den Wert 110. Damit ist der gleichzeitigen Betrieb des CAN-Talk Protokolls und des Selectron CAN Protokolls [1] möglich. Dabei gilt es zu beachten, dass beim gleichzeitigen Betrieb der zwei Protokolle die System-Kommunikation Grp. 2 des Selectron Protokolls nicht verwendet werden darf.

3.2 Aufbau der CAN-Nutzdaten

Die CAN Nutzdaten bestehen aus bis zu acht Datenbytes:

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service ID	Data	Data	Data	Data	Data	Data	Data

Das erste Byte des CAN Nutzdatenfeldes dient als Service ID. Die Service ID spezifiziert das Kommando, welches auf dem Zielsystem ausgeführt werden soll. Sie ist vergleichbar mit dem Control-Byte beim RS232-Talk Protokoll [2]. Die restlichen Datenbytes enthalten die zur Ausführung des Kommandos benötigten Daten.

3.3 Knotenadressen

Die Knotenadresse ist die physikalische Adresse eines Busteilnehmers (Knoten). Sie lässt sich durch Schalter hardwaremässig einstellen und bestimmt unter anderem die Priorität des Knotens. Das CAN-TALK Protokoll erlaubt bis zu 64 Busteilnehmer, für die folgende Adressen vergeben werden können:

Knotenadresse	**CAN-Identifier (Adr 5 … Adr 0)**	
Knotenadresse 0	00000	(Knoten mit höchster Priorität)
Knotenadresse 1	00001	
Knotenadresse 2	00010	
:		
Knotenadresse 63	11111	(Knoten mit tiefster Priorität)

Es ist darauf zu achten, dass jede Adresse in einem Bussystem nur einmal vergeben wird.

Die im CAN-Header enthaltene Adresse spezifiziert je nach Richtung der Meldung die Adresse des Senders bzw. des Empfängers:

- Meldungen mit der Richtung Host - Knoten beinhalten die Empfängeradresse
  Sendet der Host die Adresse 0000h, so handelt es sich um eine Broadcast-Nachricht, die an alle Bussteilnehmer gerichtet ist.
- Meldungen mit der Richtung Knoten - Host beinhalten die Senderadresse.

Ein Host Knoten muss nicht zwingend die Addresse 0000h aufweisen. Ein Host empfängt generell sämtliche auf dem Netz vorhandenen Meldungen und wertet diese gemäss seinem Programm aus. Das Richtungsbit im CAN-Identifier enthält die Information, ob es sich bei der aktuellen Meldung um ein Host - Knoten oder Knoten - Host Kommunikation handelt. Falls der Host nicht die Adresse 0000h besitzt, sollte diese Addresse keinem andern Knoten zugeteilt werden, da dieser Knoten dann nur über Broadcast-Nachrichten angesprochen werden könnte.

4 CAN-Talk Protokoll

Das CAN-Talk Protokoll sieht zwei Kommunikationsmodelle für den Datenaustausch zwischen Host und Zielsystemen vor, die einen Datenaustausch von einem 16 Bit Word oder von ganzen Datenblöcken ermöglichen.

- Standard Transfer
- Block Transfer Host - Knoten

Der Standard-Transfer ermöglicht das Übertragen von einzelnen 16-Bit Werten. Der Blocktransfer ermöglicht hingegen das Übertragen von grösseren Blöcken und wird beim Download von Programmen gebraucht.

4.1 Command Overview

The following table lists all the CAN-Talk commands. Afterwards all commands are described with more details.

Command ID	Description
10h	Read Memory Word
11h	Write Memory Word
12h	Write Flash Word
20h	Start Program
21h	Stop Program
22h	Reset Device
23h	Erase Flash Sector
24h	Broadcast Enable/Disable
30h	Read Memory Block
31h	Write Memory Block
32h	Download to Flash

4.2 Standard Transfer

Der Standardtransfer ermöglicht das Schreiben und Lesen von 16 Bit Wörtern aus dem ganzen Speicherbereichen des Zielsystems, sowie die Steuerung des Monitors (Starten und Stoppen von Anwenderprogrammen, Löschen von FLASH EPROM Speicher, etc.)

Beim Standard Transfer sendet der Host eine REQUEST Nachricht an einen Knoten (Knotenadresse in der CAN-ID). Der betreffende Knoten sendet eine RESPONSE zurück, welche allfällige Fehlermeldungen und je nach Transferart die gewünschten Daten enthält.

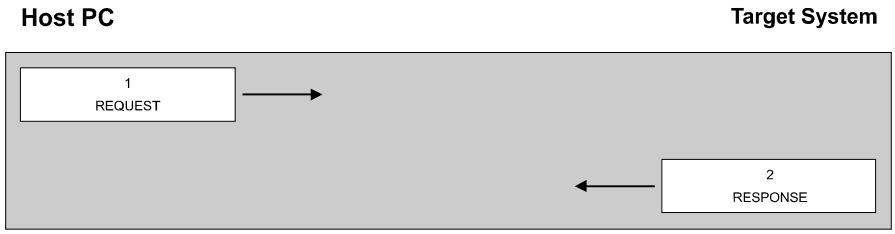

Abb. 4.1: Standard Transfer

Der Standard Transfer gestattet auch das Senden von Broadcast Nachrichten an alle Bussteilnehmer (Adresse 0000h in der CAN-ID). Nach dem Empfang der REQUEST Nachricht senden alle Bussteilnehmer die RESPONSE an den Host zurück.

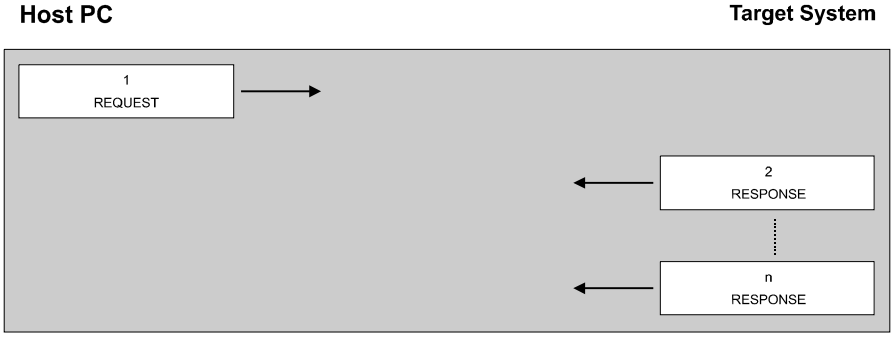

Abb. 4.2: Standard Transfer Broadcast

Versuchen zwei Knoten zur gleichen Zeit ihre RESPONSE Nachricht zu senden, wird der Knoten mit der tieferen Adresse priorisiert.

4.2.1 Read Memory

Liest ein 16 Bit Wort von der gewünschten Adresse.

Read Memory, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=10h**	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	-	-	-	-

Read Memory, RESPONSE							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=10h**	Result Ok = 00	Data Bit 0-7	Data Bit 8-15	-	-	-	-

Result	**Description**
00	Ok
10	Error

4.2.2 Write Memory

Schreibt ein 16 Bit Wort auf die gewünschte Adresse.

Write Memory, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=11h**	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	Data Bit 0-7	Data Bit 8-15	-	-

Write Memory, RESPONSE							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=11h**	Result Ok = 00			-	-	-	-

Result	**Description**
00	Ok
10	Error

4.2.3 Write Flash

Schreibt ein 16 Bit Wort ins FLASH EPROM

Write Flash, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID =12h**	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	Data Bit 0-7	Data Bit 8-15	-	-

Write Flash, RESPONSE							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=12h**	Result Ok = 00			-	-	-	-

Result	**Description**
00	Ok
01	Acknowledge
40	Flash error
41	Flash timeout
42	Flash not empty
43	Flash protected

4.2.4 Start Program

Startet ein Anwenderprogramm

Start Program, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=20h**	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	-	-	-	-

Start Program, RESPONSE							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=20h**	Result Ok = 00			-	-	-	-

Result	**Description**
00	Ok
10	Error

4.2.5 Stop Program

Stopt ein Anwenderprogramm

Stop Program, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=21h**	-	-	-	-	-	-	-

Stop Program, RESPONSE							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=21h**	Result Ok = 00			-	-	-	-

Result	**Description**
00	Ok
10	Error

4.2.6 Reset Device

Löst einen Software Reset auf dem Zielsystem aus. Dieser Befehl funktioniert nur, wenn der Monitor auf dem Zielsystem noch angesprochen werden kann (bei einem Absturz des Anwenderprogramms muss auf dem Zielsystem ein Harware Reset ausgelöst werden).

Reset, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=22h**	-	-	-	-	-	-	-

Reset, RESPONSE							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID=22h**	Result Ok = 00			-	-	-	-

Result	**Description**
00	Ok
10	Error

4.2.7 Erase Flash

Löscht den Block im FLASH EPROM, in dem die Adresse (REQUEST Byte 1 bis Byte 3) liegt.

Erase Flash, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 23**	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	-	-	-	-

Erase Flash, RESPONSE 1 (enter sector erase)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 23**	Result Ok = 00			-	-	-	-

Erase Flash, RESPONSE 2							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 23**	Result Ok = 00			-	-	-	-

Result	**Description**
00	Ok
01	Acknowledge
40	Flash error
41	Flash timeout
43	Flash protected

4.2.8 Broadcast Enable/Disable

Aktiviert oder deaktiviert das Empfangen von Broadcastmeldungen. Auf eine Broadcastmeldung “Broadcast Enable” muss auch bei deaktiviertem Broadcastempfang reagiert werden. Somit können unabhängig vom Broadcast-Zustand alle Knoten für Broadcast aktiviert werden.

Broadcast Disable, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 24**	00	-	-	-	-	-	-

Broadcast Enable, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 24**	01	-	-	-	-	-	-

4.3 Block Transfer Host - Knoten

Der Blocktransfer Host - Knoten ermöglicht das Schreiben (Laden) von grösseren Datenmengen (Programme, Daten, etc.) vom Host auf einen oder alle Knoten.

Beim Block Transfer sendet der Host ein REQUEST TASK 1 an einen Knoten (Knotenadresse in der CAN-ID), in dem die Anfangsadresse des Zielbereichs steht. Danach werden die Daten (jeweils drei Words) mittels REQUEST TASK 2 Meldungen übermittelt. Der REQUEST TASK 3 schliesst den Blocktransfer ab. Nachdem der betreffende Knoten den REQUEST TASK 3 empfangen hat, sendet er eine RESPONSE zurück, welche eine allfällige Fehlermeldung enthält.

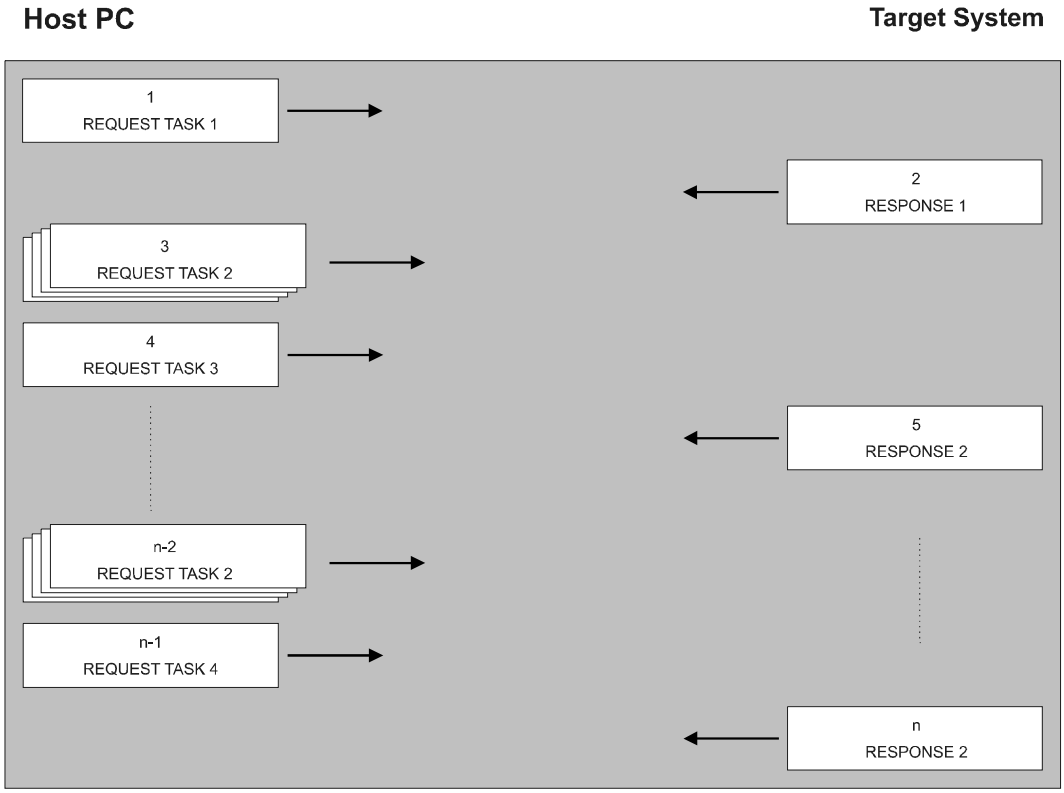

Abb. 4.1: Block Transfer Host Knoten

Der Blocktransfer Host - Knoten Transfer gestattet auch das Senden von Broadcast Datenblocks an alle Bussteilnehmer (Adresse 0000h in der CAN-ID). Nach dem Empfang des REQUEST TASK 3 senden alle Bussteilnehmer die RESPONSE an den Host zurück.

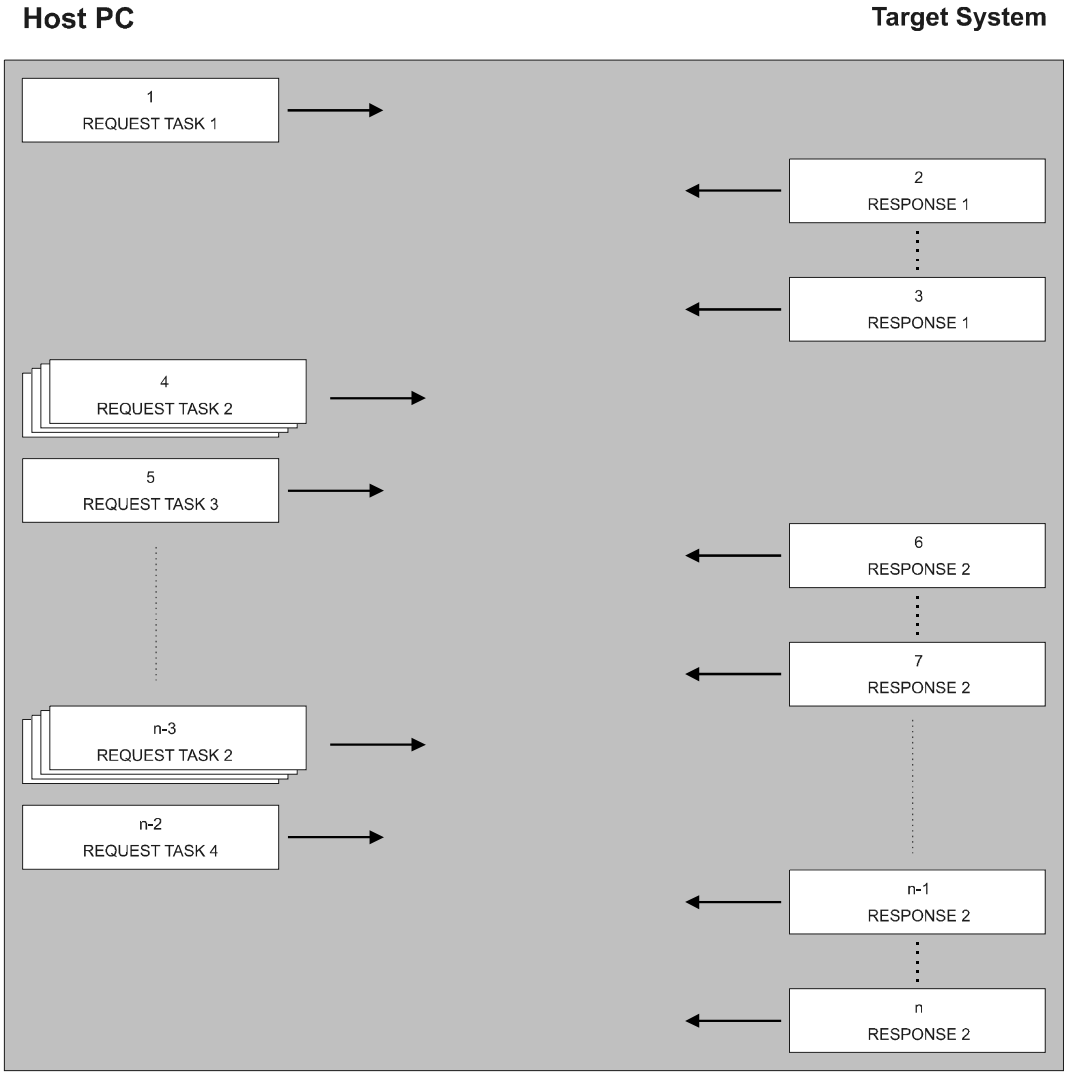

Abb. 4.2: Standard Transfer Host Knoten Broadcast

Versuchen zwei Knoten zur gleichen Zeit ihre RESPONSE Nachricht zu senden, wird der Knoten mit der tieferen Adresse priorisiert.

4.3.1 Read Memory Block

This command is used to read whole memory blocks and needs much less communication overhead than the word wise access. This command is available since monitor version 1.7.2.

Read Memory Block, REQUEST							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $30**	Request = 00	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	Size in Bytes	-	-

Read Memory Block, Response 1							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $30**	Result Ok = 01	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5

Read Memory Block, Response n (n > 1)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $30**	Result Ok = n	Data Byte 6*(n-1)	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1

Result	**Description**
01	Ok
10	Error

Notes: The Address and the Size have to be an even number.
Make sure this command is completely finished (request and all responses) before starting the next command.

4.3.2 Write Memory Block

This command is used to write whole memory blocks into RAM location. This command needs much less communication overhead than the word wise access. This command is available since monitor version 1.7.2.

Write Memory Block, Request 0 (set address and size)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $31**	Request = 00	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	Size in Bytes	-	-

Write Memory Block, Response 0 (answer to request 0)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $31**	Result Ok = 00	-	-	-	-	-	-

Write Memory Block, Request n (n > 0)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $31**	Request = n	Data Byte 6*(n-1)	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1	Data Byte 6*(n-1)+1

Write Memory Block, Response 1 (answer to request 0)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = $31**	Result Ok = 01	-	-	-	-	-	-

Result	**Description**
00	Ok
01	Ok, last telegram received
10	Error

Notes: The Address and the Size have to be an even number.
Make sure this command is completely finished (all requests and responses) before starting the next command.

4.3.3 Download to Flash

Dient zum Laden von Datenblöcken ins FLASH EPROM. Falls bei einem Zielsystem ein Fehler bei der Programmierung des FLASH EPROM auftritt, werden die nachfolgenden Daten nicht mehr ins FLASH EPROM programmiert.

Download to Flash, REQUEST TASK 1 (set address & enter blockwrite)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 32**	Task **01**	Address Bit 0-7	Address Bit 8-15	Address Bit 16-23	-	-	-

Download to Flash, REQUEST TASK 2 (add data to buffer)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 32**	Task **02**	Data 0 Bit 0-7	Data 0 Bit 8-15	[Data 1] Bit 0-7	[Data 1] Bit 8-15	[Data 2] Bit 0-7	[Data 2] Bit 8-15

Download to Flash, REQUEST TASK 3 (add data to buffer and write buffer)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 32**	Task **03**	Data 0 Bit 0-7	Data 0 Bit 8-15	[Data 1] Bit 0-7	[Data 1] Bit 8-15	[Data 2] Bit 0-7	[Data 2] Bit 8-15

Download to Flash, REQUEST TASK 4 (add data, write buffer and exit)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 32**	Task **04**	Data 0 Bit 0-7	Data 0 Bit 8-15	[Data 1] Bit 0-7	[Data 1] Bit 8-15	[Data 2] Bit 0-7	[Data 2] Bit 8-15

Download to Flash, RESPONSE 1 (to set address & enter blockwrite)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 32**	Result ok = 00	Length of buffer in bytes		-	-	-	-

Download to Flash, RESPONSE 2 (send result)							
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Service **ID = 32**	Result			-	-	-	-

Result	**Description**
00	Ok
10	Device busy
11	Service protected
20	Protocol error
22	Frame size error
24	Buffer overflow
31	Unknown service
40	Flash error
41	Flash timeout
42	Flash not empty
43	Flash protected

5 Änderungsprotokoll

geänd. Version	neue Version	Kurzbeschrieb der Änderung / Bemerkungen	Datum	Visum / KoSt
	1.0	CAN-TALK Protokollbeschreibung	25.10.1996	eb / 0150
1.0	1.1	Block Read/Write added	13.5.2002	Ro