CAN - Anwendung mit dem C167CR

Christian Perschl DSK-536\\CAN

1 Der CAN-Bus

Der CAN-Bus (CAN steht für Controller Area Network) wurde ursprünglich für die Automobiltechnik entwickelt und dient zum Datenaustausch zwischen verschiedenen Mikrocontroller-Einheiten. Aufgrund seiner Universalität ist er aber auch in anderen Aufgabenbereichen einsetzbar.

Der Bus ist als Zweidrahtleitung mit symmetrischen Signalen aufgebaut, wobei zwei Spannungszustände auftreten können (rezessiv = 1 und dominant = 0). Wie bei einem Wired-AND setzt sich bei mehreren gleichzeitigen Signalen der dominante Zustand 0 durch. Der CAN-Bus ist an den Busenden terminiert und schafft bis zu 1000 kBit/s Übertragungsrate. Bei geringeren Baudraten kann auf die Terminierung verzichtet werden.

CAN funktioniert, ähnlich Ethernet, mittels CSMA/CD mit NDA (carrier sense multiple access/collision detection with non destructive arbitration). Dies bedeutet, eine Einheit, welche eine Nachricht versenden will, „horcht" auf dem Bus, ob gerade Daten über den Bus geschickt werden oder nicht. Ist der Bus frei, so beginnt die Einheit die Übertragung. Kommt es trotzdem zu einer Kollision (dies wird durch permanentes Vergleichen der eigenen Nachricht mit jener auf dem Bus kontrolliert), so wird - im Gegensatz zum Ethernet, wo die Übertragung aller Datenpakete abgebrochen wird - die momentan höchstpriore Nachricht (message) fertig übertragen. Alle anderen Übertragungen von Nachrichten werden abgebrochen, und erfolgen selbsttätig später.

Der Austausch von Daten (sog. CAN messages) ist nachrichtenorientiert, das bedeutet, es wird nicht eine bestimmte Einheit (ein bestimmter Mikrocontroller) angesprochen, sondern die CAN message ist mit einem Identifier versehen, der Auskunft darüber gibt, um welche Nachricht es sich handelt (z.B. Motordrehzahl, Motortemperatur..). Daher kann eine Message nicht nur von einem Sender zu einem Empfänger gesendet werden, sondern jede Einheit, für die eine Nachricht interessant scheint, kann diese empfangen (Multicast- bzw. Broadcast-Verfahren).

Eine Nachricht muß aber nicht unbedingt auf Initiative des zuständigen Senders versendet werden. Es ist möglich, eine Nachricht mittels „remote frame" anzufordern.

Der entsprechende Sender antwortet, indem er die gewünschte Message sendet. In diesem Fall ist es notwendig, daß es nur einen Sender für diese Nachricht gibt.

Prinzipiell gibt es zwei CAN-Frames: Standard-CAN und Extended-CAN. Der Unterschied liegt in der Länge des Identifiers: Bei Standard-CAN ist der Identifier 11 Bits lang (2032 Möglichkeiten, 16 Messages sind reserviert), bei Extended-CAN 29 Bits (ca. 5.10^8 Möglichkeiten). Der Identifier enthält aber auch die Prioritätsinformation: Je niedriger der Identifier (), desto höher die Priorität der Message. Die eigentliche Nachricht (Daten) kann 0 bis 8 Bytes lang sein.

2 Der C167CR

Der Mikrocontroller SAB C167CR aus der C166-Familie von Siemens hat neben den für Mikrocontroller üblichen Peripherie-Funktionalitäten (I/O-Ports, Timer, asynchrone und synchrone serielle Schnittstelle, CAPCOM und PWM-Einheit, Watchdog, PLL, AD-Wandler) einen CAN-Controller integriert.

Über 15 sog. Message Objects ist es möglich, CAN-Nachrichten zu senden, zu empfangen oder anzufordern. Dabei ist für jedes Message Object einstellbar, welche CAN-Nachricht (welcher Identifier, Standard oder Extended) empfangen oder gesendet werden soll. Auch die Länge der Nachricht ist für jedes Objekt einstellbar.

Der Betrieb mittels dieser 15 Message Objects wird Full CAN genannt, welcher die CPU kaum belastet, da nur bei der Initialisierung jedes Message Object einer CAN-Message (einem Identifier) zugeordnet wird. Das Auslesen der Nachricht kann zu einem beliebigen Zeitpunkt erfolgen.

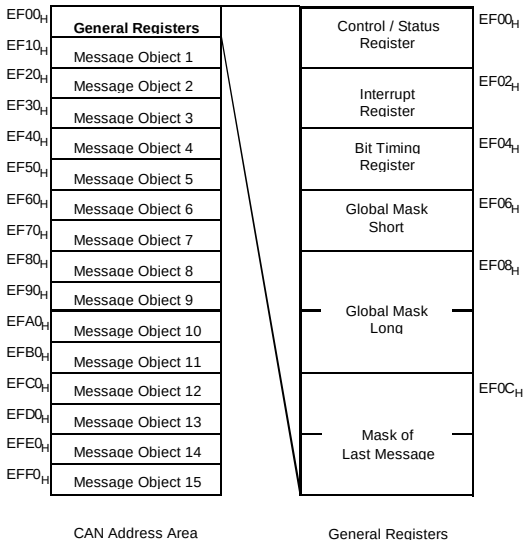
Da es für große Applikationen unter Umständen notwendig ist, mehr als 15 verschiedene Nachrichten zu empfangen oder zu senden, gibt es noch die Möglichkeit des BASIC CAN - Betriebes. Hier können über ein Message Object unterschiedliche CAN Messages empfangen werden.

Empfangene CAN Messages sollten allerdings sofort ausgelesen werden, da nachfolgende CAN Messages (mit anderem Identifier) dieses wiederum überschreiben. Dies erhöht natürlich die CPU-Belastung.

Das Senden oder Anfordern einer Nachricht muß von CPU-Seite nur einmal angeordnet werden, der CAN-Controller kümmert sich selbsttätig darum, ob es zu einer Kollision/einem Fehler gekommen ist und eventuell die Übertragung wiederholt werden muß.

Ein CRC (cyclic redundancy check) wird auch automatisch vom CAN-Modul im Mikrocontroller durchgeführt.

Die folgende Übersicht zeigt die Speicherbelegung des CAN-Moduls im C167CR:



3 Das I+ME - Entwicklungsboard

Zur einfacheren Entwicklung von Anwendungen mit dem C167CR gibt es von verschiedenen Herstellern Prototypen-Boards mit entsprechender Software. Ein sehr preiswertes Prototypenboard gibt es von I+ME: Um ca. ATS 4.200,- inkl. MwSt., Versand und Verpackung ist das Board plus Download-Kabel, Netzteil und Demo-Softwarepaket bei Firma Willert erhältlich.

Das Board ist mit C167CR, 256 kB SRAM mit 70ns, RS-232 - Schnittstelle zum PC, LEDs, DIP-Schalter und CAN-Bus-Verbindungsstecker versehen. Alle Anschlüsse des C167CR sind über einen optionalen Stecker herausgeführt. Eine Erweiterung des Speichers ist ebenfalls möglich.

Die im Paket enthaltene Software enthält den Tasking-Demo-Compiler für die C166-Familie, Windows-Demo-Debugger, MIP-Mikrocontroller Initialisierungs-Programm und ein Download-Programm hexload (von I+ME).

Die im folgenden beschriebene Applikation wurde mit 3 dieser Prototypen-Boards realisiert. Die Programme wurden mit dem BSO-Tasking-Compiler (Vollversion) erstellt, der Download zum Board erfolgt mittels hexload.

Als CAN-Bus-Verbindungsstecker wurde eine 9-polige SUB-D-Buchse (männlich) gewählt. Belegt sind 3 Pins, nämlich die zwei Signalleitungen CAN-L (Pin 2) und CAN-H (Pin 7) sowie Masse (Pin 3).

Ein passendes Kabel zur Verbindung mehrerer Boards mittels CAN ist einfach herzustellen, es empfiehlt sich bei Verwendung von mehr als 2 Boards bzw. für die Terminierungen an den Enden des Busses eine Art

T-Stück zu basteln (es müssen immer nur Pin 2 mit 2, 3 mit 3 und 7 mit 7 verbunden werden).

Für die Terminierung reicht ein 120 Ohm - Widerstand zwischen CAN-L (2) und CAN-H (7).

4 Das Projekt

4.1 Aufgabenstellung

Mithilfe des CAN-Busses sollen 3 Stationen (Europa, Japan, Amerika) Daten austauschen:

a) Jede Station fungiert lokal als Uhr, die Uhrzeit soll einstellbar und abfragbar sein.

b) Die LEDs jeder Station sollen angesteuert werden können.

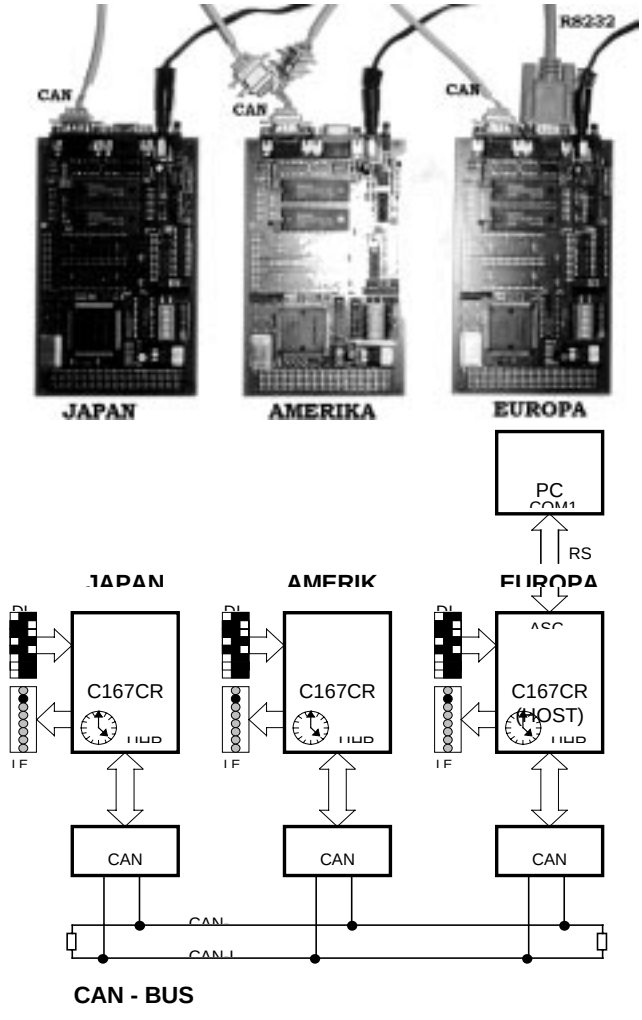
c) Bei einer Änderung der DIP-Schalter soll der neue Wert automatisch gemeldet werden.

Die Station Europa fungiert dabei als Host zum PC, da alle Einstellungen und Abfragen und die Visualisierung (Human Interface) mittels PC erfolgen sollen.

Erhält Europa vom PC (die Kommunikation erfolgt über die serielle Schnittstelle) einen Befehl, so wird dieser entsprechend aufbereitet über CAN zu den anderen Stationen gesendet oder, falls er nur die Station Europa betrifft, lokal bearbeitet.

Jede Station wurde mit dem I+ME Prototypenboard mit C167CR realisiert, die physikalische Verbindung der CAN-Schnittstellen wurde wie in Kapitel 3 beschrieben hergestellt.

Photo



4.2 Messages

Folgende Nachrichten wurden festgelegt:

ID	Beschreibung	Datenlänge/Daten	Extended?
10	Uhrzeit Japan einstellen	3 (Stunden, Minuten, Sekunden)	Nein
20	Uhrzeit Amerika einstellen	3 (Stunden, Minuten, Sekunden)	Ja
30	Uhrzeit Japan abfragen	3 (Stunden, Minuten, Sekunden)	Nein
40	Uhrzeit Amerika abfragen	3 (Stunden, Minuten, Sekunden)	Ja
50	LEDs einstellen (Für alle Stationen)	1 (Neuer Wert Lauflichter)	Nein
60	DIP-Stellung Japan	1 (Neuer Wert DIP Japan)	Ja
70	DIP-Stellung Amerika	1 (Neuer Wert DIP Amerika)	Ja

Uhrzeit einstellen

Beim PC wird die einzustellende Uhrzeit eingegeben und über die serielle Schnittstelle zum Host (Europa) gesendet.

Soll die Zeit von Europa eingestellt werden, so ist keine CAN-Message notwendig, der Host (Europa) liest über die serielle Schnittstelle alle Werte (Stunden, Minuten, Sekunden) ein und setzt seine Zeit. Bei einer Zeiteinstellung von Japan oder Amerika generiert der Host dann die entsprechende CAN-Message (10 oder 20).

Uhrzeit abfragen

Der PC sendet (aufgrund Benutzereingabe) einen Befehl (Zeitanforderung) über die serielle Schnittstelle zum Host.

Soll die Zeit von Europa abgefragt werden, ist keine CAN-Message notwendig, der Host sendet dem PC seine Zeit. Bei Anforderung der Zeit von Japan oder Amerika generiert der Host einen remote frame über die entsprechende CAN-Message (30 oder 40) und wartet auf den Empfang der angeforderten Daten. Sobald der Host die Zeit von Japan oder Amerika über den CAN-Bus empfangen hat, sendet er sie weiter zum PC.

LEDs einstellen

Beim PC kann das Lauflicht ein- und ausgeschaltet werden. Ist es eingeschaltet, so sendet der PC regelmäßig einen fortlaufenden Wert des Lauflichtes zum Host. Dieser stellt nun einerseits seine LEDs ein und generiert andererseits eine CAN-Message (50), um die anderen Stationen vom Lauflichtstatus zu informieren. Beim Empfang der CAN - Message (50) stellen auch Japan und Amerika ihre LEDs auf den neuen Wert.

DIP - Schalter

Die Stellung der DIP-Schalter soll zum PC gesendet werden, wenn sie sich geändert haben. Die Versendung der CAN-Messages 60 und 70 geht dabei von der jeweiligen Station aus, welche überprüft, ob sich die DIP-Stellung geändert hat. Bei einer Änderung wird der neue Wert des DIP-Schalters mit der entsprechenden CAN-Message (60 oder 70) versendet. Der Host (Europa) empfängt diese Messages über das Basic CAN Message Object (15) und sendet den neuen Wert zum PC. Natürlich sendet der Host auch bei einer Änderung seiner DIP-Schalterstellung den neuen Wert zum PC.

4.3 MC - Programme

Für jede Station ist ein eigenes Programm erforderlich, da unterschiedliche Aufgaben zu bewältigen sind. Zwar wären die Programme für Japan und Amerika sehr ähnlich, aber aufgrund der Zusammenfassung würde die Übersichtlichkeit des resultierenden Programmes leiden. Zur Unterscheidung würden außerdem die DIP-Schalter benötigt werden, welche aber bereits für eine andere Funktion vorgesehen sind.

Nichtsdestoweniger gibt es Konstanten und (Initialisierungs-)Routinen, welche von allen Programmen benötigt werden: die CAN-Register des C167CR, die Message-Object-Initialisierung, die Baudrateneinstellung und die Timer-Initialisierung für die Uhr.

In einer gemeinsamen Header-Datei `c167can.h` befinden sich Bezeichner für die Special Function Register des C167CR-CAN-Moduls, über welche dann auf die Register und die Message Objects zugegriffen wird. Weiters befinden sich im Header DEFINEs für CPU-Clock und die gewünschte CAN-Baudrate.

In der Funktion `init_message_object`, welche ebenfalls im Header abgelegt ist, wird für ein bestimmtes Message Object festgelegt, ob es überhaupt benutzt werden soll, ob empfangen oder gesendet werden

soll, welche CAN-Message empfangen/gesendet werden soll, wieviele Datenbytes die Nachricht lang ist und ob der CAN Identifier Standard (11 Bit) oder Extended (29 Bit) ist.

```
init_message_object (int oid , int use , int dir , int dlc , int xtd, long mid)
```

`oid` Object ID, für welches Message Object die Parameter eingestellt werden sollen

`use` Gibt an, ob dieses Message Object verwendet (1) oder nicht verwendet (0) werden soll.

`Dir` Gibt an, ob das Object zum Senden (1) oder Empfangen (0) einer Message verwendet werden soll.

`Dlc` Gibt die Länge der Nachricht in Bytes an

`xtd` Gibt an, ob der Identifier Standard (0) oder Extended (1) ist

`mid` Gibt den CAN-Identifier für dieses Object an

So ergeben sich für die einzelnen Stationen folgende Aufrufe in der Initialisierungsfunktion `init_can`:

Europa:	
`init_message_object (1,1,1,3,0,10)`	Sende Uhrzeit nach Japan
`init_message_object (2,1,1,3,1,20)`	Sende Uhrzeit nach Amerika
`init_message_object (3,1,0,3,0,30)`	Empfange Uhrzeit von Japan (nach entspr. remote request)
`init_message_object (4,1,0,3,1,40)`	Empfange Uhrzeit von Amerika (nach entspr. remote request)
`init_message_object (5,1,1,1,0,50)`	Sende Lauflicht - Wert (broadcast)
`init_message_object (15,1,0,1,1,0)`	Empfange DIP von Japan (60) oder von Amerika (70) mittels basic CAN

Japan:	
`init_message_object (1,1,0,3,0,10)`	Empfange Uhrzeit von Europa
`init_message_object (2,1,1,3,0,30)`	Sende Uhrzeit nach Europa
`init_message_object (3,1,0,1,0,50)`	Empfange Lauflicht-Wert
`init_message_object (4,1,1,1,1,60)`	Sende DIP-Wert

Amerika:	
`init_message_object (1,1,0,3,1,20)`	Empfange Uhrzeit von Europa
`init_message_object (2,1,1,3,1,40)`	Sende Uhrzeit nach Europa
`init_message_object (3,1,0,1,0,50)`	Empfange Lauflicht-Wert
`init_message_object (4,1,1,1,1,70)`	Sende DIP-Wert

Die richtige Berechnung der Registerwerte für das Bit Timing Register, welches für die resultierende Baudrate des CAN-Busses zuständig ist, ist recht komplex, vor allem, weil es entweder gar keine Lösung gibt (es lassen sich nicht x-beliebige Baudraten einstellen), oder aber gleich mehrere. Die Funktion `init_can_baudrate` berechnet die richtigen Werte und setzt gleich die entsprechenden Register. Die Berechnung der Baudrate ist an [1] angelehnt.

`init_can_baudrate()` berechnet für folgende Baudraten entsprechende Werte für das Bit Timing Register:

BAUDRATE	BRP	TSEG1	TSEG2	SJW
10000	49	10	7	2
20000	24	10	7	2
25000	19	10	7	2
40000	24	4	3	2
50000	9	10	7	2
100000	4	10	7	2
125000	3	10	7	2
200000	4	4	3	2
250000	1	10	7	2
500000	0	10	7	2
625000	0	8	5	2
1000000	0	4	3	2

In jedem der einzelnen Programme für die Stationen gibt es eine Initialisierungsroutine `init_can`. Hier wird das Control-Register richtig gesetzt, die Baudrate initialisiert, die Message Objects konfiguriert und die Maskenregister gesetzt. Dabei ist wichtig, daß für das Basic-CAN-Feature das Last-Message-Maskenregister (für das Message Object 15 gibt es ein eigenes Maskenregister, welches mit dem globalen Maskenregister verUNDet wird) auf 0 gesetzt wird und somit alle auftretenden Messages (außer jene, welchen bereits ein anderes Message Object zugeordnet ist) mittels Basic Can (Message Object 15) empfangen werden.

Zur allgemeinen Initialisierung wird die Funktion `Init` aufgerufen. In dieser Funktion werden zusätzliche Parameter für den externen Bus gesetzt, die bereits beschriebene Funktion `init_can` aufgerufen, Uhr und Timer initialisiert.

[WR1]

Im Hauptprogramm der Stationen Japan und Amerika werden lediglich Änderungen der DIP-Schalter festgestellt. Kommt es zu einer Änderung, wird eine entsprechende CAN Message generiert (Europa sendet den neuen DIP-Wert direkt zum PC).

4.4 Kommunikation zwischen Host (Europa) und PC

Der Host Europa kommuniziert mit dem PC über die asynchrone serielle Schnittstelle ASC. Prinzipiell muß unterschieden werden, ob Terminal-Ausgaben oder „echte" Daten übertragen werden. Terminal-Daten sind Daten, die zu Debug-Zwecken eingebaut wurden und im CAN-Monitor Window der PC-Oberfläche erscheinen (z.B. die Meldung „Uhrzeit von Japan nach Europa empfangen: 12:23:11"). Das PC-Programm kann Terminal-Ausgaben des Host nicht auswerten. Dazu gibt es den Transfer-Modus:

Zwecks Identifikation des Modus (Terminal-Modus=Textübertragung=0, Transfer-Modus=Datenübertragung=1) beginnt ein Datentransfer vom Host zum PC immer mit dem Bytewert 1. Der PC schaltet dann in den Transfer-Modus (Datentransfer Host-PC) um und erkennt anhand des nächsten Bytes (wir werden dieses Byte Communication ID nennen) , welche und wieviele Daten der Host übermittelt. Nachdem alle Daten empfangen wurden, schaltet der PC automatisch wieder in den Terminal-Modus.

Folgende Communication IDs treten dabei auf :

Comm. ID	Beschreibung	Datenbytes	Datenformat
2	Uhrzeit von Europa	3	Stunden/Minuten/Sekunden
4	Uhrzeit von Japan	3	Stunden/Minuten/Sekunden
6	Uhrzeit von Amerika	3	Stunden/Minuten/Sekunden
7	DIP von Europa	1	DIP-Stellung
8	DIP von Japan	1	DIP-Stellung
9	DIP von Amerika	1	DIP-Stellung

Beispiel für einen Frame im Transfer-Modus

1 Schaltet in Transfer-Modus um
4 Uhrzeit von Japan
12 Stunden
34 Minuten
20 Sekunden

1	4	12	34	20

Demnach ist es in Japan 12:34:20 Uhr.

Beim Datentransfer vom PC zum Host ist es nicht notwendig, zwischen Transfer-Modus und Terminal-Modus zu unterscheiden. Der Host befindet sich praktisch immer im Transfer-Modus. Daher ist das Senden des Bytes 1 nicht erforderlich.

Folgende Transfers treten auf:

Comm. ID	Beschreibung	Datenbytes	Datenformat
1	Uhrzeit nach Europa	3	Stunden/Minuten/Sekunden
2	Uhrzeit Europa anfordern	0	
3	Uhrzeit nach Japan	3	Stunden/Minuten/Sekunden
4	Uhrzeit Japan anfordern	0	
5	Uhrzeit nach Amerika	3	Stunden/Minuten/Sekunden
6	Uhrzeit Amerika anfordern	0	
7	LED (an alle)	1	LED-Wert für Lauflicht

4.5 Das PC - Steuerungsprogramm (Human Interface und Terminal)

Um die entsprechenden Befehle zu erteilen, Werte zu empfangen und darzustellen, dient das PC-Programm `can.exe`. Es zeigt für jede Station Uhrzeit und DIP-Schalterstellung sowie das Lauflicht an. Mithilfe der Tasten 1-7 können Befehle an den Host (Europa) gesendet werden.

Außerdem besitzt `can.exe` eine Terminalfunktion. Im unteren Abschnitt des Bildschirms werden Ausgaben des Host, die nicht als Datenaustausch dienen, angezeigt.

Hier ein Screenshot des PC-Steuerungsprogrammes:

5 Fazit

Der CAN-Bus erweist sich als ideale Kommunikationsschnittstelle zwischen verschiedenen Mikrocontrollern:

Hohe Übertragungsraten (bis 1 MBit/s), hohe Fehlersicherheit (CRC, Symmetrische Signale, „Handshake"), Echtzeitfähigkeit, einfache Programmierung (durch Message Objects).

Der physikalische Aufwand ist minimal, der Bus kommt mit 2 (mit Masse 3, beim Auto ist Masse am Chassis) Leitungen aus, im Grenzfall sogar mit 1 Leitung (!), die Terminierung kann für niedrige Baudraten weggelassen werden. Als Übertragungsmedium ist praktisch alles verwendbar: verdrillte Zweidrahtleitung, BNC-Kabel, sogar Lichtwellenleiter (es ist nur ein entsprechender Transceiver erforderlich).

Die hohe Betriebssicherheit wird einerseits durch NDA (non-destructive arbitration) und andererseits durch CRC (cyclic redundancy check) gewährleistet.

Das größte Plus des CAN-Busses ist aber die Anwender- bzw. Programmierseite. Man braucht sich kaum um etwas selbst kümmern: Die Zuordnung von einlangenden Nachrichten zum entsprechenden Message Object, der CRC, ev. wiederholtes Senden im Fehler- oder Kollisionsfall und der Informationsaustausch bei einem remote frame erfolgen automatisch.

Die Message Objects ermöglichen es, daß Nachrichten jederzeit ausgelesen werden können.

Außerdem ist der CAN-Bus genormt (ISO 11898) und somit der Nachrichtenaustausch zwischen Bausteinen verschiedener Hersteller möglich.

Neben dem C167CR hat auch ein Derivat der 8051/C501 Familie, nämlich der C515C den CAN-Controller integriert. Es gibt auch CAN-Controller als eigenständige Bausteine (z.B. SAE 81C90/91), welche ein Mikrocontrollersystem ohne CAN-feature um dieses erweitern können.

Für die beschriebene Applikation war ein Zeitaufwand (Einarbeitung in die Materie, Installation der Software und Hardware, Aufbau der CAN-Verbindung und Terminierung, Programmierung der MC-Programme, Programmierung des PC-Programmes, Dokumentation) von insgesamt 10 Tagen erforderlich, wobei die Programmierung des CAN-Teiles völlig unproblematisch war. Das Einarbeiten in die Thematik wurde durch die angeführte Literatur wesentlich erleichtert.

Resümee des Autors

Der CAN-Bus ist sehr vielseitig und doch einfach zu programmieren, und sollte ich einmal den ganzen Haushalt vernetzen, werde ich den CAN-Bus verwenden.

6 Dateienverzeichnis

Dateien mit allgemeinen Funktionen für MC	
`hwtrap.c`	Interruptvektoradressen für Laufzeitfehler sinnvoll beschreiben
`serio.c`	Vom Compilerhersteller mitgelieferte Datei (für `printf` erforderlich)
`serio.h`	Vom Compilerhersteller mitgelieferte Datei (für `printf` erforderlich)
`cstart.asm`	Vom Compilerhersteller mitgelieferte Datei. Programm, welches vor `main()` aufgerufen wird und den MC initialisiert

MC-Programme	
`c167can.h`	Initialisierungsfunktionen für das CAN-Modul (wird von allen MC- Programmen benötigt)
`europe.c`	Host-Applikations-Programm für das Board Europa
`japan.c`	Applikations-Programm für das Board Japan
`america.c`	Applikations-Programm für das Board Amerika

Steuerdateien	
`makefile`	Steuerdatei für mk166 (Generierung der ausführbaren MC-Programme)
`cmd_link`	Kommandodatei für den Linker
`cmd_loc`	Kommandodatei für den Locater

Batch-Dateien	
`load_eu.bat`	Download Europa
`load_ja.bat`	Download Japan
`load_am.bat`	Download Amerika

PC-Programmdateien	
`can.c`	Source-Code für PC-Applikation
`can.exe`	Ausführbare PC-Applikation

Dokumentation	
`canproj.doc`	Diese Datei (im Winword 6.0 - Format)
`speicher.wmf`	Speicherbelegung des CAN-Moduls im C167CR
`bsb.wmf`	Blockschaltbild der gesamten Applikation
`pcprog.bmp`	Screenshot der PC-Applikation
`foto.bmp`	Photographie des Versuchsaufbaues

7 Literaturverzeichnis

[1] CAN-Anschluß für die Mikrocontrollerfamilien C166 und C500 (PCN-LIT-117 bzw. als Datei PCN-DSK-515)

[2] Siemens C167 Derivatives User´s Manual, 03.96 Version 2.0