

CAN-BusTreiber

Dok-Rev. 1.3 vom 18.03.2004
Software-Rev. 2.0 vom 11.10.1996

Inhaltsverzeichnis

1	Urheberrecht und Haftung	3
1.1	Handhabung	3
1.2	Erklärung	3
2	Allgemeine Beschreibung	4
3	Initialisierung	5
4	Empfangsfenster setzen	6
5	Lesen von CAN-Messages	7
6	Setzen eines Timeout beim Senden	8
7	Schreiben von CAN-Messages	9
8	Abfrage der Anzahl Empfangsmessages	10

Revisionsliste:

Rev.	Datum	Na.	Änderung
1.0	11.07.1996	Ko	Erstellung
1.1	06.07.2000	Ko	Umstellung auf software.dot
1.2	19.03.2002	Ko	PEARL-Prozeduren in Großbuchstaben
1.3	18.03.2004	Ko	"ENTRY" fehlte bei den PEARL-SPC's

1 Urheberrecht und Haftung

Alle Rechte an diesen Unterlagen liegen bei der IEP GmbH, Langenhagen.

Die Vervielfältigung, auch auszugsweise, ist nur mit unserer ausdrücklichen schriftlichen Genehmigung zulässig.

In Verbindung mit dem Kauf von Software erwirbt der Käufer einfaches, nicht übertragbares Nutzungsrecht. Dieses Recht zur Nutzung bezieht sich ausschließlich darauf, daß dieses Produkt auf oder in Zusammenhang mit jeweils **einem** Computer zu benutzen ist. Das Erstellen einer Kopie ist ausschließlich zu Archivierungszwecken unter Aufsicht des Käufers oder seines Beauftragten zulässig. Der Käufer haftet für Schäden, die sich aus der Verletzung seiner Sorgfaltspflicht ergeben, z.B. bei unautorisiertem Kopieren, unberechtigter Weitergabe der Software usw.. Der Käufer gibt mit dem Erwerb der Software seine Zustimmung zu den genannten Bedingungen. Bei unlizensiertem Kopieren muß vorbehaltlich einer endgültigen juristischen Klärung von Diebstahl ausgegangen werden. Dies gilt ebenso für Dokumentation und Software, die durch Modifikation aus Unterlagen und Programmen von IEP hervorgegangen ist, gleichgültig, ob die Änderungen als geringfügig oder erheblich anzusehen sind.

Eine Haftung seitens IEP für Schäden, die auf den Gebrauch von Software, Hardware oder Benutzung dieses Manuskriptes zurückzuführen sind, wird ausdrücklich ausgeschlossen, auch für den Fall fehlerhafter Software oder irrtümlicher Angaben.

Das Einverständnis des Käufers oder Nutzers für den Haftungsausschluß gilt mit dem Kauf und der Nutzung der Software und dieser Unterlagen als erteilt.

1.1 Handhabung

Lesen Sie bitte zuerst sorgfältig diese Dokumentation bevor Sie anfangen zu programmieren. Sie sparen Zeit und vermeiden Probleme.

1.2 Erklärung

Wir behalten uns das Recht vor, Änderungen, die einer Verbesserung der Schaltung oder des Produktes dienen, ohne besondere Hinweise vorzunehmen. Trotz sorgfältiger Kontrolle kann für die Richtigkeit der hier gegebenen Daten, Schaltpläne, Programme und Beschreibungen keine Haftung übernommen werden. Die Eignung des Produktes für einen bestimmten Einsatzzweck wird nicht zugesichert.

2 Allgemeine Beschreibung

Die Ansteuerung der CAN-Bausteine erfolgt über eine Prozedurschnittstelle. Es stehen 6 Funktionen zur Verfügung. Im folgenden wird zuerst die C, dann die PEARL-Syntax dargestellt. Zum Senden und Empfangen wird die Struktur CAN_MESSAGE mit folgendem Aufbau genutzt:

```
typedef struct CAN_MESSAGE          TYPE CAN_MESSAGE STRUCT
{
    WORD    identifier    ;          [
    WORD    rtr           ;          identifier    FIXED(15) ,
    WORD    data_length   ;          rtr         FIXED(15) ,
    UBYTE   data[8]      ;          data_length  FIXED(15) ,
                                          data         CHAR(8)
}                                       ] ;
    CAN_MESSAGE ;
```

identifier darf im Bereich 0...2032 liegen. Mit rtr = 1 wird eine Remote Datenübertragung angefordert. Sonst ist rtr auf 0 zu setzen. In der data_length ist die Anzahl der Datenbytes angegeben, zulässig sind Werte zwischen 0 und 8. In data sind die Nutzdaten abgelegt. data muß **nicht** als CHAR(8) abgelegt werden, nur die Länge **muß** 8 Byte sein! So könnten z.B. für data auch 2 FIXED(31) Variablen angelegt werden.

Diese Beschreibung ersetzt nicht das Studium des Datenblattes des 82C200/SJA1000. Genausowenig werden Kenntnisse über die Funktionsweise des CAN-Busses vermittelt.

3 Initialisierung

Vor dem Aufruf anderer Funktionen muß die CAN-Schnittstelle initialisiert werden. Das geschieht mit der Funktion `can_init`. Dabei wird ein Reset auf dem CAN-Baustein ausgelöst, es wird festgelegt, welches Interface genutzt werden soll, wie groß die Baudrate ist und wieviele Plätze der Empfangspuffer hat. Der Empfangspuffer wird gelöscht.

```
WORD can_init( WORD can_nr, CAN_INIT *can_init_ptr ) ;
```

```
SPC CAN_INIT ENTRY ( FIXED(15), CAN_INIT IDENT )  
                    RETURNS( FIXED(15) ) GLOBAL;
```

Die Struktur `CAN_INIT` hat folgenden Aufbau:

```
typedef struct CAN_INIT          TYPE CAN_INIT STRUCT  
{                                [  
    UBYTE      btr0 ;           btr0      CHAR(1)  ,  
    UBYTE      btr1 ;           btr1      CHAR(1)  ,  
    WORD       iso  ;           iso       FIXED(15) ,  
    WORD       anz  ;           anz       FIXED(15)  
}                                ] ;  
CAN_INIT ;
```

Die `can_nr` gibt die Nummer der CAN-Schnittstelle an, zulässig sind z.Z. die Werte 1 oder 2. Die Baudrate muß über die Register `BTR0` und `BTR1` gesetzt werden. Weitere Informationen entnehmen Sie bitte dem Datenblatt des 82C200. Die Variable `iso` ist auf 1 zu setzen, wenn das ISO-Interface genutzt werden soll, auf 0 für die RS485-Schnittstelle. Bei Einsatz der MOCAN ist nur `iso = 1` zulässig. `anz` gibt die Anzahl Plätze des Empfangspuffers an, max. werden 64 Plätze unterstützt. Die Akzeptanzmaske wird so gesetzt, daß alle Messages empfangen werden können.

Der Rückgabewert gibt eine evt. Fehlernummer zurück:

Name	Wert	Bedeutung
<code>E_OK</code>	0	kein Fehler
<code>E_NO_CAN</code>	-1	CAN-Baustein nicht verfügbar
<code>E_WRONG_PHYS</code>	-3	Interface unzulässig
<code>E_WRONG_READ_BUFFER</code>	-4	Empfangspuffergröße unzulässig
<code>E_NO_MEMORY</code>	-14	kein Speicher für Empfangspuffer

Wird ein Fehler zurückgegeben, so ist der CAN-Baustein nicht sende-/empfangsbereit, d.h. die Initialisierung muß wiederholt werdend.

`E_WRONG_READ_BUFFER` zeigt an, daß versucht wurde mehr als 64 Empfangspuffer zu allozieren. Wird `E_NO_MEMORY` zurückgegeben, so stand RTOS-UH nicht mehr genügend freier Speicher zur Verfügung.

4 Empfangsfenster setzen

Mit dem `acr` (Acceptance Code Register) und `amr`-Register (Acceptance Mask Register) kann eine Akzeptanzmaske eingegeben werden. Nur Messages, deren Identifier akzeptiert werden, werden in den Empfangspuffer eingetragen. Beim Aufruf dieser Funktion wird der Empfangspuffer gelöscht.

```
WORD can_set_accept( WORD can_nr, UBYTE acr, UBYTE amr ) ;
```

```
SPC CAN_SET_ACCEPT ENTRY ( FIXED(15), FIXED(15), FIXED(15) )  
    RETURNS( FIXED(15) ) GLOBAL ;
```

Die Bits 7 bis 0 aus `acr` werden mit den Bits 10 bis 3 des Identifiers der einlaufenden Message verglichen. Sind die Bits gleich, wird im Ergebnis eine 1 gesetzt. Das Ergebnis wird mit den negierten `amr` Bits 'verodert'. Nun muß `$FF` herauskommen, damit die Message akzeptiert wird. Anders formuliert: Nur die Bits, die im `amr` auf LOW gesetzt sind, werden beim Vergleich beachtet. Als Rückgabewerte können auftreten:

Name	Wert	Bedeutung
<code>E_OK</code>	0	kein Fehler
<code>E_NO_CAN</code>	-1	CAN-Baustein nicht verfügbar
<code>E_NO_INIT</code>	-5	keine Initialisierung durchgeführt

5 Lesen von CAN-Messages

Die Verwaltung des Empfangspuffers wird von der CAN-Betreuungstask durchgeführt. Beim Lesen mit `can_read` wird die übergebene Struktur gefüllt. Der Aufrufer wird bis zum Eintreffen einer Message blockiert. Beim Aufruf der `can_init`-Funktion werden evt. noch wartende Leser freigegeben.

```
WORD can_read( WORD can_nr, CAN_MESSAGE *read_ptr ) ;
```

```
SPC CAN_READ ENTRY ( FIXED(15), CAN_MESSAGE IDENT )  
                    RETURNS( FIXED(15) ) GLOBAL ;
```

Der Rückgabewert gibt eine evt. Fehlernummer zurück:

Name	Wert	Bedeutung
	>0	Anzahl noch wartender Messages
E_OK	0	kein Fehler
E_NO_CAN	-1	CAN-Baustein nicht verfügbar
E_ERROR_LEVEL	-2	Warning Level des 82C200 erreicht
E_NO_INIT	-5	keine Initialisierung durchgeführt
E_OFF_BUS	-6	Off-Bus
E_OVERRUN	-7	Overrun aufgetreten
E_READ_BUF_OVERRUN	-8	Empfangspuffer übergelaufen
E_RESET	-9	Reset ausgelöst (mit <code>can_init</code>)

Wird ein Wert größer Null zurückgegeben, so liegen noch weitere Nachrichten im Empfangspuffer vor, die sofort abgeholt werden können.

Bei den Fehlern `E_OVERRUN` und `E_READ_BUF_OVERFLOW` sind Empfangsdaten verlorengangenen! Es sind aber gültige Daten von der Schnittstelle gelesen und in der übergebenen Struktur eingetragen worden. Diese beiden Fehlermeldungen werden nur einmal übergeben, außer sie treten erneut auf. `E_OVERRUN` heißt, daß die Daten nicht schnell genug vom CAN-Baustein abgeholt worden sind. Das kann nur passieren, wenn das System zulange im "OFF" ist oder eine Interrupt-routine auf höherem Level zu lange läuft. `E_READ_BUF_OVERRUN` zeigt an, daß der Empfangspuffer zu klein ist bzw. zu selten Daten von der CAN-Schnittstelle abgeholt wurden. Bei dem Fehler `E_ERROR_LEVEL` ist der Warning-Level des CAN-Bausteins erreicht, die Daten wurden aber trotzdem korrekt übertragen.

6 Setzen eines Timeout beim Senden

Mit der `can_set_timeout`-Funktion kann ein Timeout beim Senden angegeben werden.

```
WORD can_set_timeout( WORD can_nr, LONG timeout ) ;  
SPC CAN_SET_TIMEOUT ENTRY ( FIXED(15), DURATION )  
                            RETURNS( FIXED(15) ) GLOBAL ;
```

`timeout` ist in Millisekunden anzugeben.

Der Rückgabewert gibt eine evt. Fehlernummer zurück:

Name	Wert	Bedeutung
E_OK	0	kein Fehler
E_NO_CAN	-1	CAN-Baustein nicht verfügbar
E_NO_INIT	-5	keine Initialisierung durchgeführt

Ein Timeout kann z.B. auftreten, wenn kein weiterer Teilnehmer am Bus ist, da dann keine Bestätigung für die erfolgreiche Versendung erfolgt und der CAN-Baustein ununterbrochen sendet. Im Fehlerfall wird die Sendung abgebrochen und der Write-Auftrag mit einer entsprechenden Fehlermeldung zurückgegeben.

7 Schreiben von CAN-Messages

Mit der `can_write`-Funktion wird eine Message verschickt.

```
WORD can_write( WORD can_nr, CAN_MESSAGE *write_ptr ) ;
```

```
SPC CAN_WRITE ENTRY ( FIXED(15), CAN_MESSAGE IDENT )  
      RETURNS( FIXED(15) ) GLOBAL ;
```

Der Rückgabewert gibt eine evt. Fehlernummer zurück:

Name	Wert	Bedeutung
E_OK	0	kein Fehler
E_NO_CAN	-1	CAN-Baustein nicht verfügbar
E_ERROR_LEVEL	-2	Warning Level des 82C200 erreicht
E_NO_INIT	-5	keine Initialisierung durchgeführt
E_OFF_BUS	-6	Off-Bus
E_RESET	-9	Reset ausgelöst (mit <code>can_init</code>)
E_IDENTIFIER	-10	Identifizier unzulässig
E_LENGTH	-11	Länge unzulässig
E_RTR	-13	rtr-Wert unzulässig
E_TIMEOUT	-15	Timeout beim Senden

Bei Rückgabe von `E_ERROR_LEVEL` wurde die Nachricht erfolgreich verschickt. Bei allen anderen Fehlern wurde die Nachricht nicht verschickt.

8 Abfrage der Anzahl Empfangsmessages

Es wird die Anzahl noch im Empfangspuffer stehender Messages zurückgegeben.

```
WORD can_nr_of_messages( WORD can_nr, WORD *nr_of_message_ptr ) ;
```

```
SPC CAN_NR_OF_MESSAGES ENTRY ( FIXED(15), FIXED(15) IDENT )  
                                RETURNS( FIXED(15) ) GLOBAL ;
```

Der Rückgabewert gibt eine evt. Fehlernummer zurück:

Name	Wert	Bedeutung
E_OK	0	kein Fehler
E_NO_CAN	-1	CAN-Baustein nicht verfügbar
E_NO_INIT	-5	keine Initialisierung durchgeführt
E_OFF_BUS	-6	Off-Bus
E_OVERRUN	-7	Overrun aufgetreten
E_READ_BUF_OVERRUN	-8	Empfangspuffer übergelaufen

Bei den . . .OVERRUN-Fehlern wird die richtige Anzahl der im Empfangspuffer stehenden Messages zurückgegeben. Die Fehlermeldungen werden nicht gelöscht!