Ingenieurbüro für Echtzeitprogrammierung

Arcnet-Treiber

Dok-Rev. 1.1 vom 22.09.2006

Software-Rev. 1.0 vom 18.02.2003

Inhaltsverzeichnis

Revisionsliste:

Rev.	Datum	Na.	Änderung
1.0	06.03.2003	Ko	Erstellung
1.1	22.09.2006	Ko	`pending_rx` richtig erklärt

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

Der Arcnet-Treiber stellt eine einfache Schnittstelle zur Kommunikation mit anderen Acrnet-Geräten zur Verfügung. Die Verwaltung der Kommunikation (z.B. Tokenweitergabe, Tokeninitialisierung usw.) wird komplett vom Arcnet-Controller durchgeführt. Sie als Anwender müssen den Arcnet-Controller einmalig initialisieren, danach können Sie sofort Pakete senden und empfangen. Arcnet führt normalerweise eine gerichtete Kommunikation durch, d.h. die Pakete werden von einem bestimmten Absender an einen bestimmten Empfänger geschickt. Die Pakete werden vom Empfänger bestätigt, d.h. Sie als Absender können sicher sein, dass das Paket beim Empfänger angekommen ist.

Es ist aber auch möglich, Broadcast-Nachrichten an alle Busteilnehmer zu verschicken. Dann wird als Empfängeradresse die 0 eingetragen. Es darf also keine Station mit der Stationsadresse 0 geben. Ein Broadcast-Paket wird natürlich nicht bestätigt, die Empfänger können sogar den Empfang von Broadcast-Paketen ablehnen. Sie können also nie sicher sein, ob das Broadcast-Paket alle gewünschten Empfänger tatsächlich erreicht hat.

Der Arcnet-Bus läuft grundsätzlich mit 2,5 Mbps.

3 Initialisierung

Vor der Initialisierung können keine Daten verschickt bzw. empfangen werden. Nur die Statusabfrage kann durchgeführt werden. Mit dem Aufruf der folgenden Funktion wird die Initialisierung durchgeführt:

```
UWORD arcnet_init( USR_Init *init_ptr ) ;
```

Der `init_ptr` muß auf die folgende Struktur verweisen:

```
typedef struct USR_Init
{
   UBYTE  source_id ;
   UBYTE  flags     ;
   UBYTE  max_nodes ;
   UBYTE  dummy     ;
}
   USR_Init ;
```

`source_id` enthält die eigene Knotennummer, sie muss in einem Arcnet-Netz eindeutig sein. In `flags` werden einige Parameter des Arcnet eingestellt, beachten Sie bitte, dass einige Werte im gesamten Netz gleich eingestellt sein müssen, damit eine Kommunikation möglich ist:

```
#define L_FLAG   0x80   /* long packets allowed               */
#define R_ALL    0x40   /* Receive all , monitor mode        */
#define S_4NACK  0x20   /* set to 4 Nack if 1 , else 128 Nack */
#define U_ET3    3      /* set extended Timeout to   82 us   */
#define U_ET2    2      /* set extended Timeout to  328 us   */
#define U_ET1    1      /* set extended Timeout to  656 us   */
#define U_ET0    0      /* set extended Timeout to 1312 us   */
```

In `max_nodes` wird die Anzahl der Knoten im Arcnet-Netz angegeben. Es sind die Werte 16, 32, 64 und 255 zulässig. Bei anderen Werten wird 255 eingestellt. Auch dieser Wert muss auf allen Knoten identisch eingestellt sein. Die Anzahl der Knoten beeinflußt die Länge des Timeouts bei Änderungen im Netzwerk. Es treten Werte von 26,25 ms bis 420 ms auf.

4 Statusabfrage

Der Status des Arcnet-Controllers kann jederzeit – auch vor der Initialisierung – abgefragt werden:

```
UWORD arcnet_status( USR_Stat *status_ptr ) ;
```

Die Struktur `USR_Stat` wird mit den folgenden Werten gefüllt:

```
typedef struct USR_Stat
{
   UBYTE  stat           ; /* Status of Driver             */
   UBYTE  nodeid         ; /* own Node-ID                  */
   UBYTE  pending_rx     ; /* Pakets not fetch yet by user */
   UBYTE  pending_tx     ; /* Pakets not yet sended        */
   UWORD  rx_packets     ; /* all received pakets          */
   UWORD  tx_packets     ; /* all transmitted pakets       */
   ULONG  nodes[ 256/32] ; /* aktiv nodes in network       */
}
    USR_Stat ;
```

In `stat` sind folgende Werte definiert:

```
#define IS_INIT 0x01 /* ARCNET Driver is initialized */
#define IS_BUS  0x80 /* Online on BUS                */
```

Bekommen Sie also `0x00`, so ist der Treiber nicht initialisiert und nicht im logischen Ring vertreten. Bei `0x81` ist der Treiber initialisiert und online, d.h. Sie können Daten versenden und empfangen.

In der `nodeid` ist die eigene Knotennummer eingetragen. `pending_tx` enthält die Anzahl der noch zu sendenden Pakete. Der Wert in `pending_rx` zeigt an, wieviel unbearbeitete `arcnet_read` Aufträge noch beim Treiber in Bearbeitung sind. In `rx/tx_packets` wird die Anzahl der empfangenen bzw. verschickten Pakete gezählt. Diese Zähler zählen rund, d.h. nach dem maximal Wert beginnt der Zählerstand wieder bei 0. In dem Feld `nodes` erhält man eine Liste der am Bus verfügbaren Teilnehmer.

Nach der Initialisierung ist der Arcnet-Knoten immer am Bus verfügbar. Im Fehlerfall (z.B. Kabeltrennung) ändert sich der Wert in `stat`. Ist der Fehler behoben, nimmt der Knoten automatisch wieder am Busverkehr teil.

Im Feld nodes erhalten Sie eine Liste aller aktiv am Bus vertretenen Teilnehmer. Für jeden aktiven Teilnehmer steht eine `1` in der Liste, sonst eine `0`.

5 Daten empfangen

Nach der Initialisierung können mit der folgenden Funktion Daten empfangen werden:

```
UWORD arcnet_read( USR_Data *data, int timeout ) ;
```

Die Struktur USR_Data hat folgenden Aufbau:

```
typedef struct USR_Data
{
   UBYTE  source_id      ;
   UBYTE  dest_id        ;
   UWORD  len            ;
   UBYTE  data[MAX_DATA] ;
}
   USR_Data ;
```

Die Struktur wird beim Empfang eines Paketes vom Treiber gefüllt. Sie müssen MAX_DATA auf 508 stellen, damit Sie jedes beliebige Paket empfangen können. Ist Ihr Empfangspuffer zu kurz, werden andere Daten überschrieben, das kann ggf. den Absturz des Systems zur Folge haben!

Für die `arcnet_read`-Funktion kann ein Timeout eingestellt werden, d.h. wird in der eingestellten Zeit kein Paket empfangen, so kehrt die Funktion mit einer Fehlermeldung zurück. Das Timeout wird auf ein 512 ms Raster abgebildet, der maximale Wert ist 127 * 512 ms = 65 sec. Timeoutwerte von 0-511 geben also kein Timeout, 512-1023 geben 512 ms Timeout usw.

6 Daten senden

Nach der Initialisierung können mit der folgenden Funktion Daten gesendet werden:

```
UWORD arcnet_write( USR_Data *data ) ;
```

Die Struktur USR_Data hat folgenden Aufbau:

```
typedef struct USR_Data
{
   UBYTE  source_id      ;
   UBYTE  dest_id        ;
   UWORD  len            ;
   UBYTE  data[MAX_DATA] ;
}
   USR_Data ;
```

In `source_id` tragen Sie die eigene Knotennummer ein, in `dest_id` die des Empfängers. In `len` muß die Länge der Nutzdaten stehen, maximal sind 508 Byte zulässig, aber nur wenn Sie bei der Initialisierung die langen Pakete zugelassen haben. Kurze Pakete haben eine maximal zulässige Länge von 253 Bytes. Mit `dest_id = 0` können Sie einen Broadcast versenden.

Sie können max. 2 Pakete gleichzeitig verschicken. Weitere Pakete werden erst bearbeitet, wenn wieder ein Sendepuffer frei geworden ist.

7 Fehlermeldungen

Die folgenden Fehlermeldungen können auftreten:

```
#define ERR_TOO_LONG  0x8021  /* Packet too long        */
#define ERR_EXNACK    0x8022  /* Excessive NACK         */
#define ERR_TIMEOUT   0x8023  /* Timeout for Reading    */
#define ERR_NO_INIT   0x8024  /* Driver not Initialized */
#define ERR_DUP_ID    0x8025  /* Duplicate ID           */
#define ERR_NO_NET    0x8026  /* No Network aktivity    */
#define ERR_NO_STATIO 0x8027  /* No Station response    */
#define ERR_HW_FAIL   0x8028  /* Hardware Failed        */
#define ERR_INIT_DONE 0x8029  /* Init already done      */
```