Chain-Gang-Bus

Protokolldefinition

**Präambel**

Dieses Dokument beschreibt ein serielles Bus-Protokoll für die Hausautomation zur Steuerung und Datensammlung von Sensornetzwerken. Es wird davon ausgegangen, dass die Bus-Teilnehmer BT niedriges Datenaufkommen erzeugen. Das Protokoll soll einfach und kostengünstig zu implementieren sein.

**Topologie**

Der Bus ist ein Master-Slave-Bus. Die BT kommunizieren über RS232-konforme serielle Leitungen. Dabei ist wie bei einer Eimerkette jeweils der TX-Ausgang eines BT mit dem RX-Eingang des nächsten BT verbunden. Jeder BT kann Daten versenden, wenn der Bus frei ist. Die Slaves sollen keine Datenpakete untereinander versenden. Der Master versendet Pakete mit der Zieladresse des Slaves. Der Slave versendet eigene Pakete mit seiner eigenen Adresse. Der Master leitet keine Pakete weiter.



**Paketstruktur**

Die BT versenden Datenpakete variabler Länge auf Basis von 8bit-Worten.

CTRL - Startbyte: Im Startbyte und nur dort ist das Bit7 gesetzt. So wird der Bus synchronisiert. Die übrigen 7Bit enthalten ein Control-Wort zur Bussteuerung. Die Befehlscodes sind so definiert, dass stets Bit 7 gesetzt ist.

---

CTRL

1

ADR1

LEN

DATA

DATA

DATA

CRC7

0

0

0

0

0

0

ADR – Adresse: Bit 0-7 enthalten die Adresse des Slaves. Der Master hat die Adresse 0. Damit sind 127 Slaves adressierbar.

LEN – Länge: Die Zahl der folgenden Anzahl Bytes kodierter Nutzdaten. LEN darf 0 sein, dann ist auch CRC7 0.

DATA – Daten: Hier folgen genau LEN Bytes. Verwendet werden jeweils nur die unteren 7 Bit. Der Slave muss diese nach 8 Bit umkodieren. Es können also pro Paket 105 Bytes (8 Bit) Nutzdaten transportiert werden.

CRC7 – Prüfsumme: Prüfsumme über die Datenbytes. Wenn die Prüfsumme nicht stimmt, wird das Paket verworfen und der Slave sendet 0xC2 (CRC failure) an den Master mit seiner Adresse und LEN 0.

**Allgemeine Paketverarbeitung**

Der Slave wartet im WAITFORSTART-Modus auf das Eintreffen eines CTRL-Bytes. Dieses wertet er aus. Die definierten CTRL-Worte stehen weiter unten. Im Falle von normalem Netztransport liest der Client die Adresse und vergleicht mit seiner eigenen, Trifft diese nicht zu, schaltet er in den PASSTHRU-Modus. Er versendet die seit CTRL eingegangenen und alle weiteren eingehenden Bytes bis zum Eintreffen eines neuen CTRL-Bytes.

Trifft die Adresse zu, liest der Slave LEN und alle folgenden Bytes bis zum CRC7. Stimmt die CRC, dann wandelt der Slave die 6-bit Daten in 8-bit um und reicht sie an die Anwendung weiter. Trifft die CRC nicht zu, verwirft er das Paket. Auch wenn zwischenzeitlich ein neues CTRL-Byte eingeht, wird das Paket verworfen. In beiden Fällen geht der Slave danach in den WAITFORSTART-Modus.

Der Slave versendet eigene Pakete nur, wenn er im WAITFORSTART-Modus ist. Dann schaltet er in den SEND-Modus und versendet das Paket gemäß den Kodierungsregeln. Eintreffende Bytes werden in der Zwischenzeit gepuffert. Ist das Senden beendet, liest der Slave die eingegangenen Bytes.

Versendet der Master Daten mit Data Ack (0x81), dann sendet der Slave bei erfolgreicher Übertragung ein Paket mit Data Ack, seiner Adresse und LEN 0. Der Slave darf keine Daten mit Data Ack anfordern.

**Adressaushandlung**

Die Slaves starten im WAITFORSTART-Modus mit unbekannter Adresse.

Der Slave sendet ein Adress-CTRL-Wort mit der ADR 0 und LEN 0 (=>CRC 0). Der Slave liest ADR, addiert 1 (nach den Regeln) und übernimmt diesen Wert als eigene Adresse. Der Slave sendet das Paket mit der aktualisierten Adresse weiter. So erhalten die Slaves eine Adresse entsprechend ihrer Position im Bus beginnend mit 1.

**Ping**

Beim Ping sendet der Master ein Ping Ctrl Wort (0x83) an eine Adresse und mit der Länge LEN 0. Der Slave antwortet mit einem identischen Paket.

**Fehlerbehandlung**

Es gibt drei mögliche Fehlerzustände:

1. Der Eingangspuffer ist voll
2. Ein Paket war unvollständig
3. CRC stimmt nicht.

In diesen Fällen sendet der Slave einen Fehler-CTRL-Code an den Master.

**CTRL-Worte**

0x80 = Data transport

0x81 = Data Ack

0x82 = Get Address

0x83 = Ping

0xC0 = Buffer full

0xC1 = Packet incomplete

0xC2 = CRC failure

**Mögliche Implementierung**

Die Slave-Funktion kann in einem dedizierten µC realisiert werden. Z. B. PIC16F1415 mit einem UART, einem SPI, und Hardware-CRC (0,80€). Als RS232-Bustreiber kommt der ST232B/C (0,50€) zum Einsatz. Beide Bauteile sind SMD (14-TSSOP, 16-TSSOP). Zusammen mit einer 5V-SV könnte das Modul als Huckepack-Platine ausgeführt werden.

**SPI-Protokoll**

Der µC kommuniziert mit dem Applikations-µC (AC) über SPI.

Die SPI-Leitungen sind MISO, MOSI, ´CS. Zusätlich gibt es die Steuerleitungen Data Available DA, und RD/´WR.

Hat der BC ein korrektes Datenpaket empfangen, setzt er DA auf Hi. Der AC kann nun Daten lesen.

Dazu setzt der AC die RD/´WR auf High, dann setzt er ´CS auf low. Dadurch legt der BC das 1. Byte auf den Bus. Solange Daten verfügbar sind, bleibt DA auf High. Hat der BC das letzte Byte auf den Bus gelegt, setzt er DA auf low. Vor dem Lesen jedes Byte sollte der AC prüfen, ob DA noch High ist.

Zieht der AC ´CS vorzeitig auf High, werden die restlichen Daten verworfen und DA auf Low gesetzt.

Will der AC senden, setzt er RD/´WR auf Low, dann setzt er ´CS auf Low und beginnt die Daten zu senden. Der BC schickt den Buffer Offset des Sendepuffers. Bei dem ersten Byte, sendet er 0. Bei dem 2. Byte sendet er 1, usw. Bei 105 ist der Puffer voll, weitere Bytes werden ignoriert. Nachdem alle Bytes übertragen wurden, setzt der AC `CS auf High. Durch dieses Signal beginnt der BC, die Daten zu codieren und zu versenden.

**Mögliche Fehlerquellen**

Die Puffer haben keine Reserven, trifft beim BC vom Bus ein neues komplettes Paket ein, bevor der AC die Daten abholt, gibt es eine Kollision und Daten im Puffer werden überschrieben. Es gibt keine Fehlerbehandlung.

Worst case beim Empfangen: Der Master sendet unmittelbar hintereinander 2 Pakete mit 2 Byte (Minimum) Nutzdaten. Dann ist die Paketlänge 6 Byte (CTRL, ADR, LEN, D1, D2, CRC). Der BC beginnt, den Paketpuffer zu überschreiben, wenn die CRC geprüft ist. D. h. Nach dem Eintreffen des ersten Pakets hat der AC bei 9600Baud 6ms Zeit, das erste Paket abzuholen, bis das zweite Paket das erste überschreibt. Ist das Paket länger bleibt entsprechend mehr Zeit.

Der AC sollte also inkl. Latenz den SPI-Bus mit mindestens 3 kHz Takt betreiben.

Worst case beim Senden: Wenn der AC ein Paket an den BC übertragen hat, muss dieser es codieren und in den Sende-FIFO schreiben. Schickt der AC die Maximallänge von 105 Byte, dann benötigt der BC bei 9600 Baud etwa 0,1s zum Versenden. In dieser Totzeit darf der AC kein weiteres Paket versenden, sonst werden korrupte Daten in den FIFO geschrieben.