T. Dauner
M. Krenmayr

Laborbericht Realtimesystems

Projekt: Eisenbahn

Aufgabenstellung

Es soll eine Eisenbahn mittels eines RTAI Linux gesteuert werden. Zur Steuerung ist die Übertragung von Datenpaketen notwendig. Die Eisenbahn kommuniziert mit dem NMRA-DCC Protokoll.

NMRA-DCC Protokoll

Das NMRA-DCC Protokoll ist ein Protokoll zur Steuerung von Modelleisenbahnanlagen. Es arbeitet Paketorientiert. Ein Paket besteht aus einer Präambel zur Synchronisation, einem Trennbit, einem Adressbyte, einem Trennbit, einem Befehlsbyte, einem Trennbit und einem Prüfbyte mit anschließendem Endbit. Das Adressbyte beinhaltet die Adresse des zu steuernden Geräts, das Befehlsbyte enthält den auszuführenden Befehl. Das Prüfbyte wird aus einer XOR Verknüpfung von Adress- und Befehlsbyte berechnet.

Pakete

Die Pakete müssen dauerhaft gesendet werden. Ein Paket muss mindestens alle 30 ms erneut gesendet werden.

1-Bit

Als 1-Bit wird ein Signal erkannt, welches 58 us an sowie 58 us aus ist.

0-Bit

Als 0-Bit wird ein Signal erkannt, welches 116 us an sowie 58 us aus ist.

Präambel

Zur Synchronisation werden 14 1-Bits übertragen

Trennbit

Ein Trennbit ist immer ein 0-Bit.

Adressbyte

Das Adressbyte beinhaltet die Adresse der Lok. In diesem Fall die Zahl 3, dual codiert.

Befehlsbyte

Im Befehlsbyte wird der entsprechende Befehl übertragen

Prüfbyte

Das Prüfbyte dient zur Überprüfung der Datenübertragung. Es wird aus der XOR Verknüpfung von Adressbyte und Befehlsbyte gewonnen.

Endbit

Das Endbit ist ein 1-Bit.

Ansatz

Timer

Als Timer verwenden wir einen periodischen Timer, da die Intervalle sich während der Laufzeit nicht verändern. Die Timer-Periode ist auf 58 us eingestellt. Die Task-Periode ist in unserem Fall unwichtig, da wir den Task in Endlosschleife ausführen, da wir keine Task-Periode abwarten müssen.

Kommunikation mit Eisenbahn

Die Signale werden über die Serielle Schnittstelle am Port 0x378 gesendet.

Kommunikation mit UI

Für die Kommunikation mit dem User-Interface verwenden wir eine FIFO-Warteschlange.

Synchronisation

Um den Zugriff auf die FIFO-Warteschlange zu Synchronisieren wird eine Semaphore verwendet.

Programmaufbau

Initialisierungsroutine

In der Initialisierungsroutine werden Timer und Semaphore initialisiert, der Task gestartet und der FIFO-Handler erzeugt.

Task

Im Task wird in Endlosschleife die Funktion „run“ aufgerufen. „Run“ kümmert sich um die Datenübertragung. Zunächst wird nach erfolgter Abfrage der Semaphore aus dem von der UI eingelesenen Befehl das Prüfbyte berechnet. Anschließend wird das Paket gesendet. Hierfür werden diverse Hilfsfunktionen verwendet.

FIFO-Handler

Bei eingehender Nachricht in der FIFO-Warteschlange wird diese durch den Handler verarbeitet. Der eingelesene Befehl der UI wird dann ausgewertet. Es wird das Befehlsbyte festgelegt. Während diesem Vorgang wird der Zugriff auf das Befehlsbyte mittels der Semaphore blockiert um Fehler zu vermeiden. Anschließend wird die Nachricht aus der FIFO-Warteschlange entfernt.

UI

Die UI besteht aus einer Skript-Datei. Mittels eines Menüs können durch die Zahleneingabe 1 bis 7 verschiedene Funktionen ausgeführt werden. Das Skript liest durchgehend die Nutzereingabe ein und schreibt eine entsprechende Nachricht an das „/dev/rtf3“ Device, welches eine FIFO-Warteschlange ist.