I2C Treiber

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum	10.01.2010
Quelle	Github → Dokumente → 02_Design → 02.02_Moduldesign
Autoren	Claas Schlüter
Version	1.0
Status	Freigegeben

1 Historie

Version	Datum	Autor	Bemerkung
0.1	01.12.10	Claas Schlüter	Initiale Beschreibung des Moduls
0.2	03.12.10	Claas Schlüter	Beschreibung/Funktionsweise des I2C-Busses
0.3	13.12.10	Claas Schlüter	Aktuell existente Probleme beschrieben
0.4	31.12.10	Robert Lucke	Review
0.5	09.01.10	Claas Schlüter	Anpassung entsprechend der Reviewvorschläge
1.0	10.01.10	Robert Lucke	Review und Freigabe

2 Inhaltsverzeichnis

3 Einleitung

Dieses Modul dient zur Anbindung des C515C Microcontrollers an ein I2C-Bus System. Über diese Schnittstelle soll es ermöglicht werden, Debug Informationen über den aktuellen Systemzustand abzusenden.

Der Treiber liest die vorhandenen Debug Informationen aus dem Shared-Memory und gibt diese anschließend an ein an den Bus angeschlossenes Gerät weiter.

4 Referenzierte Dokumente

Grundlage für dieses Moduldesign bilden folgende Dokumente:

- Aulis >
  Projektverzeichnis (WiSe0910>
  Schnittstellendokumentation >
  > **c515c Datasheet.pdf**
  > **c515c User Manual.pdf**

und

- Aulis >
  Projektverzeichnis (WiSe1011) >
  I2C-Dokumentation >
  > **i2c_spezifikation.pdf**

5 Architektur

5.1 Der I2C-Bus

5.1.1 Hardware-Konfiguration

Da der eingesetzte Microcontroller keine direkte I2C Funktionalität bereitstellt, muss diese eigens implementiert werden.

Hierfür werden 2 Pins des fünften Ports benutzt um die erforderlichen Leitungen bereitzustellen (SCL und SDA).

5.1.2 Aufgaben

Aufgabe der I2C-Treibers ist es, Daten aus einem im Shared Memory befindlichen Ringpuffer einzulesen und zu übertragen.

Hierfür ist es Aufgabe des Treibers, die Gegenstelle zu adressieren, die Daten zu übertragen und anschließend den Bus wieder frei zu geben.

Evtl. auftretende Probleme beim Senden auf den Bus, sollen in das Shared Memory geschrieben werden, um in höheren Schichten hieraus reagieren zu können.

Ferner ist es angedacht, dass der Treiber auch Daten auf dem Bus entgegen nimmt und im Shared-Memory in einem Ringpuffer ablegt.

6 Probleme

6.1 Zeiteinteilung

Zum aktuellen Zeitpunkt ist es nicht geklärt, ob innerhalb von einer Zeiteinheit (welche dem Treiber zur Verfügung gestellt wird) alle erforderlichen Debug Informationen übertragen werden können.

6.2 Abgebrochene bzw. fehlgeschlagene Übertragungen

Sollte eine Übertragung fehlschlagen, weil z.B. der *Slave* die Daten nicht schnell genug verarbeiten kann, oder die Zeiteinheit überschritten wurde, ist ein Verfahren zu überlegen, wie eine abgebrochene Übertragung fortgesetzt werden kann.

7 Anhang

7.1 I2C-Bus

Der I2C Bus ist ein Master-Slave Bussystem. Der Master sendet Daten auf den Bus und der Slave reagiert hierauf. In unserem Hardwareaufbau ist der C515C Mikrocontroller als Master zu betrachten und der Arduino Mikrocontroller als Slave.

7.1.1 Aufbau des Busses

Notwendig für die Realisierung eines I2C-Busses, sind 2 Signalleitungen:

SCL – Serial Clock Line

und

SDA – Serial Data Line

Beide dieser Leitungen werden per Pull-Up-Widerstand an die Versorgungsspannung angeschlossen.

7.1.2 Taktrate und Übertragungsgeschwindigkeit

Das Taktsignal, welches auf der SCL Leitung anliegt, wird vom jeweiligen Master erzeugt. Es gibt innerhalb der I2C Spezifikation vordefinierte Modus, welche eine maximale Taktrate in diesen vorgeben, allerdings ist es dem Master selbst überlassen, diese frei zu wählen.

Zusätzlich sieht es die I2C Spezifikation vor, dass ein langsameres Gerät z.B. ein Slave durch halten eines Low-Pegels auf der Taktleitung die Übertragung verzögern kann.

7.1.3 Dateneinheit und deren Gültigkeit

Eine Dateneinheit besteht aus 8 Daten- bzw. Adressbits und einem ACK (Acknowledge) Bit. Einzelne Datenbits sind nur dann gültig, wenn sich ihr Zustand während einer Clock-High Phase nicht ändert.

Das im neunten Takt übertragene ACK-Bit wird nur dann als solches gewertet, wenn auf der SDA Leitung Low-Pegel geführt wird. Andernfalls wird die vorangegangene Übertragung als nicht erfolgreich bewertet (siehe Taktrate und Übertragungsgeschwindigkeit – Verzögerung durch z.B. den Slave).

7.1.4 Start-Signal

Das Start-Signal wird durch eine fallende Flanke auf der SDA Leitung kenntlich gemacht, während die SCL Leitung High-Pegel führt.

7.1.5 Stopp-Signal

Das Stopp-Signal wird durch eine steigende Flanke auf der SDA Leitung kenntlich gemacht, während die SCL Leitung High-Pegel führt.

7.1.6 Adressierung

Für die Adressierung nutzt der der I2C Bus 7 Bits welche die Adresse des Gerätes enthalten, mit welchem kommuniziert werden soll, gefolgt von einem Bit (R/W), welches angibt ob von dem Gerät gelesen oder zu ihm geschrieben werden soll (Schreiben logisch: 0, lesen logisch: 1).

7.1.7 Ablauf einer Übertragung

Initiiert wird die Übertragung durch ein Start-Signal vom Master, gefolgt von der Adresse des Gerätes und des R/W Bits. Der Slave (**im Falle einer schreibenden Übertragung**) quittiert den Empfang mit einem ACK-Bit.

Nun können beliebig viele Bytes über den Bus übertragen werden, bis der Master die Übertragung mit dem Stopp-Signal abschließt.