Modul-Design Auditing System

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum	19.05.2010
Quelle	Dokumente\02_Design\02.02_Moduldesign
Autoren	Icken, Jan-Christopher
Version	0.7
Status	Zur Revision freigegeben

1Historie

Version	Datum	Autor	Bemerkung
0.1	09.12.2009	Felix Blüml	Erste Version.
0.2	22.12.2009	Felix Blüml	Kap. 5.2: Änderung der Groß-/Kleinschreibung von AS_msg_array und AS_next_msg. Dokumentenübergreifend: Änderung des Schnittstellennamens von as_init() in initAS(). Anpassung von Abb. 1, 2 und 3 an diese Änderungen. Dokumentenübergreifend: Änderung von „unsigned integer 8 Bit“ in „unsigned char“.
0.3	18.01.2010	Felix Blüml	Anpassung an die Änderung der Modulspezifikation: Die erhaltenen Meldungen der Module sollen nun nicht mehr nur in einem Ringpuffer abgelegt werden. Das Modul erhält jetzt eine Zeitscheibe, durch die Betriebsmittelverwaltung, in der die Meldungen an einen PC weitergeleitet werden sollen.
0.4	07.02.2010	Felix Blüml	Kap. 7.2: Korrektur des Wortlautes. Abb.1: Korrektur des Kommentars. Dokument-übergreifend: Änderung der Schnittstelle von send_msg in sendMsg. Anpassung an die Änderung der Modulspezifikation: Es soll nun nicht mehr versucht werden, alle Meldungen innerhalb eines Zeitfensters zu versenden. Pro Aufruf der Schnittstelle workAS() sollen nur maximal vier Meldungen an den Arduino versendet werden. Im Falle eines anstehenden Not-Aus soll das Modul Not-Aus-Treiber die neue Schnittstelle reportAllMsg() verwenden, um alle bisher gesammelten Meldungen noch versenden zu können.
0.5	09.02.2010	Felix Blüml	Kap. 5.4.4 und 6.2.4: Optimierung des Algorithmus. Entfernen der Wartepause aus reportAllMsg(). Kap. 5.4.3 und 6.2.4: Hinzufügen der Wartepause zu workAS().
0.6	14.02.2010	Felix Blüml	Kap 5.4.3: Korrektur des geschätzten Zeitaufwandes. Kap 6.2.3: Rechtschreibkorrektur. Kap. 7.2: Korrektur „Bit“ durch „Byte“. Korrektur des Wortlautes.
0.7	19.05.2010	Icken, Jan-Christopher	Abgleich mit dem vorhandenen Quellcode, Entfernung der äußeren Schnittstellen, Anpassung des Layouts, Korrektur

			von Rechtschreibfehlern

2Inhaltsverzeichnis

3Einleitung

Dieses Dokument beschreibt das Modul Auditing System (AS) des Software-Designs.

Das AS soll alle gelieferten Statusmeldungen, ausgewählter Module, chronologisch per serieller Schnittstelle an einen angeschlossenen PC übermitteln.

Da die Übertragung zeitintensiv ist, sollen die Statusmeldungen zunächst in einem Ringpuffer zwischengespeichert werden.
Bei zugeteilter Zeitscheibe durch die Betriebsmittelverwaltung sollen jeweils maximal vier gesammelte Meldungen, die bisher noch nicht versendet wurden, ausgelesen und über den I²C-Bus an den Mikrocontroller „Arduino Duemilanove“ (AD) versendet werden.
Von hieraus sollen die Meldungen per serieller Schnittstelle an den angeschlossenen PC weitergeleitet werden.

Bei erkanntem Versagen des Systems sendet das Modul Not-Aus-Treiber ein Not-Aus-Signal. Nach diesem Vorgang soll dieses Modul noch die Möglichkeit haben, alle bisher noch nicht gesendeten Meldungen aus dem Puffer zu versenden.

4Referenzierte Dokumente

Software-Design Version 1.0,
Aulis Gruppe WiSe0910: Design → Subsystemdesign → Software

Quellcode zur I²C Implementierung für den C515C (I2C_SW.C),
Aulis Gruppe WiSe0910: Implementierung

5Architektur

In den folgenden Unterkapiteln wird die Architektur des Moduls näher erläutert. Siehe dazu auch die Abbildung 1.

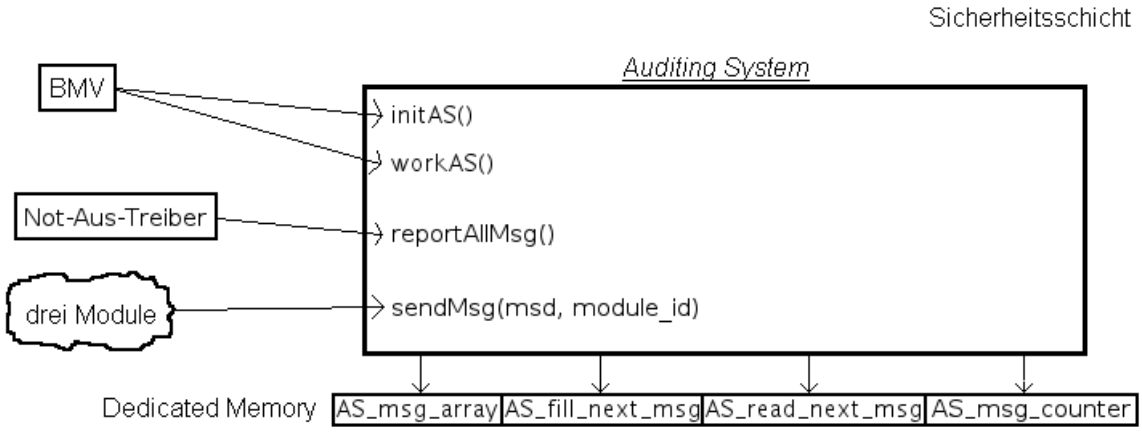

Abb. 1: Schnittstellen und globale Variablen des AS.

5.1Funktionshierarchie

Das Auditing-System ist im Software-Design in der Sicherheitsschicht angesiedelt. Es ist jedoch gestattet und gefordert, dass die Schnittstellen in einigen Fällen schichtenübergreifend aufgerufen werden. Aufrufe erfolgen durch die Module selbst. Siehe dazu auch die Architektur des Software-Designs.

5.2Daten

Zum speichern der Statusmeldungen der Module wird ein globales Array als Ringpuffer verwendet. In ihm werden die letzten 30 Vorkommnisse festgehalten um im späteren Programmverlauf zur Auswertung abrufbereit zu sein.
Daraus ergibt sich ein Platzbedarf von 30 * (|Statusmeldung| + |ModulID|) Byte = 30 * (6+1) Byte = 210 Byte:

unsigned char AS_msg_array [30][7].

Um festzuhalten, welches Feld des Puffers beim nächsten Aufruf durch das AS überschrieben werden soll, wird das Byte unsigned char $AS\_read\_next\_msg$ als IN-Index verwendet.

Das Byte unsigned char $AS\_fill\_next\_msg$ wird hingegen zum Daten auslesen als OUT-Index verwendet.

Zum zählen des Puffer-Füllstandes wird das Byte unsigned char $AS\_msg\_counter$ verwendet.

5.3Abhängigkeiten von anderen Modulen

Für die korrekte Verwendung der Schnittstellen SendMsg *(void sendMsg(byte msg[6], byte module_id))* und WorkAS *(workAS())* muss das Modul Betriebsmittelverwaltung vorher die Schnittstelle InitAS *(void initAS())* aufrufen.

Für die korrekte Verwendung der Schnittstelle ReportAllMsg *(void reportAllMsg())* muss das Modul Not-Aus-Treiber vorher den Watchdog des Moduls Software-Watchdog ausgeschaltet haben.

6Dynamisches Verhalten

Im Folgenden wird das dynamische Verhalten aller, im Modul vorhandenen, Schnittstellen beschrieben.

6.1Allgemeines

Es werden keine Rückmeldungen an den Software-Watchdog gegeben. Die im System-Design vorgeschlagene Schnittstelle *helloModul(module_id, status)* des SW wird nicht benutzt.
Durch Verwendung von *helloModul(module_id, status)* besteht die Gefahr, dass das gesamte System unnötig gestoppt wird. Mögliche Unregelmäßigkeiten beim sammeln oder versenden von Statusmeldungen sind vorgesehen.

6.2Funktionsbeschreibung

6.2.1 _I2CBitDly()

Führt beim Aufruf NOP-Instruktionen aus, die ein warten von ~4,7ns ermöglichen.

6.2.2 _I2CSCLHigh()

Setzt das SCL-Signal High und wartet so lange bis dies geschieht.

6.2.3 I2CSendAddr(byte addr, byte rd)

Generiert die Startbedingung für ein Senden an das Gerät mit der Adresse addr und sendet das byte rd.

6.2.4 I2CSendByte(byte bt)

Sendet das byte bt an ein Gerät welches vorher mit I2CSendAddr adressiert wurde.

6.2.5 I2CSendStop()

Generiert die Stopbedingung auf dem I2C-Bus.

6.2.6 warten()

Führt NOP-Instruktionen aus die für das Warten zwischen dem verschicken von Nachrichten über den I2C-Bus notwendig sind.

7Hardware-Umgebung

Um die Statusmeldungen der Module lesen zu können wird zusätzliche Hardware benötigt. Diese wird in den folgenden Unterkapiteln erklärt.

7.1Aufbau

Pin 4 des AD ist die SDA-Leitung des I²C-Busses. Dieser wird mit dem Pin 5.4 des C515C verbunden.

Pin 5 des AD wird Pin 5.5 verbunden und ist die SCL-Leitung.

Der AD wird seriell über USB mit einem PC verbunden.

7.2Daten auslesen und auswerten

Implementierung der Software auf dem Arduino siehe Dokument:

Google Code → Dokumente → 02_Design → 02.02_Moduldesign → Modul-Design_Auditing System_Arduino

Im Falle von Übertragungsstörungen versucht das Auditing-System nur halb gesendete 7er-Gruppen erneut zu senden. Es gibt keine Erfolgsgarantie für dieses Gelingen, da im Falle eines Pufferüberlaufs alte Daten überschrieben werden.