Modul-Design Betriebsmittelverwaltung

Für das studentische Projekt *Sichere Eisenbahnsteuerung*

Datum 17.06.2010
Quelle Dokumente\02_Design\02.02_Moduldesign
Autoren Ole Bohn
Version 3.0
Status freigegeben

1 Historie

Version	Datum	Autor	Bemerkung
0.1	26.12.09	A.Gottwald	Ver. 0.1
0.2	08.01.10	A.Gottwald	unsigned char anstatt Byte #define
0.3	10.01.10	A.Gottwald	Byte typedef
1.0	18.01.10	A.Gottwald	Zeitscheibe geändert alle unsigned char gegen byte getauscht Synthax in den structs korrigiert Variable AS_msg_counter hinzugefügt typedefs eingerückt Fehlerbytes mit 0 initialisiert HW Reset durch SW Reset entfernt Betriebsmittelverwaltung.h statt BMV.h
1.1	24.01.2010	A.Gottwald	Sprachliche Verbesserungen, Modul IDs anders benannt
1.2	02.02.2010	A.Gottwald	S88 und AS Ports neu zugewiesen
2.0	29.04.2010	O. Bohn	Anpassung des Layouts an den Standard des Projekts „Sichere Eisenbahnsteuerung“ im SoSe 10. Anpassen der Beschreibung an die Funktionalität des Quellcodes in den Dateien Betriebsmittelverwaltung.c und Betriebsmittelverwaltung.h Überarbeiten der Abschnitt 5 und 6.
2.1	06.05.2010	F. Geber, O. Bohn	Dateipfad anpassen, Layout anpassen
3.0	17.06.2010	K. Dziembala	Anpassung des Kapitels 6.1 und 6.2 an den aktuellen Quell-Code; Freigabe des Dokuments

2 Inhaltsverzeichnis

3 Einleitung

Das Modul „Betriebsmittelverwaltung“ ist für die Verwaltung der Systemressourcen zuständig. Hierzu zählen sowohl die Software- als auch die Hardwareressourcen.

Die Module werden von der Betriebsmittelverwaltung, im Kooperativen Multitasking aufgerufen. Die Überwachung der einzelnen Module hingegen wird nicht von der Betriebsmittelverwaltung vorgenommen, dies geschieht im Modul Software Watchdog der Sicherheitsschicht. Ein Debug Modus ist in dieser Modul Design Version nicht vorgesehen.

In diesem Dokument werden folgende Abkürzungen verwendet:

- AS (Auditing System)
- BMV (Betriebsmittelverwaltung)
- BV (Befehlsvalidierung)
- EV (Ergebnisvalidierung)
- FP (Fahrprogramm)
- LZ (Leitzentrale)
- RS232 (RS232-Treiber)
- S88 (S88-Treiber)
- SM (Shared Memory)
- SSC (SSC-Treiber)
- SW (Software Watchdog)
- HW (Hardware Watchdog)

4 Referenzierte Dokumente

- Software Design 1.0 vom 17.11.2009, Quelle: Aulis – Projektverzeichnis - 02: Design - 02: Subsystemdesign - Software
- Auditing Modul Design 0.1 vom 09.12.2009, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- Befehlsvalidierung Modul Design 1.1 vom 18.01.2010, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- Ergebnisvalidierung Modul Design 0.9 vom 13.01.2010, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- Fahrprogramm Modul Design 1.1 vom 10.01.2010, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- Not-Aus-Treiber 0.3 Modul Design vom 14.12.2009, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- RS232 Treiber 0.4 vom 12.12.2009, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- S88 Modul-Design 0.5 vom 30.11.2009, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- SCC Modul Design 0.8 vom 13.01.2010, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- Software Watchdog 1.2 vom 12.01.2010, Quelle: Aulis – Projektverzeichnis - 02: Design - 03: Moduldesign
- C für den Mikrokontroller 80C515C, Quelle: Aulis – Projektverzeichnis - 03: Implementierung
- Einführung in die maschinennahe Programmierung, Unterlagen zur MPR Vorlesung, HS Bremen 2003, Quelle: Prof.Dipl.Ing.S.Myrzik, MPR Vorlesung

5 Architektur

5.1 Funktionshierarchie

Entsprechend dem Software-Design Dokuments ist das Modul Betriebsmittelverwaltung in der Treiberschicht der Software angeordnet.

In diesem Modul befindet sich die Hauptmethode des gesamten Programms, die sogenannte main () Funktion. Aus dieser Methode heraus werden zyklisch die anderen Module initialisiert bzw. aufgerufen.

Aufgebaut ist das Modul Betriebsmittelverwaltung aus der C-Datei Betriebsmittelverwaltung.c sowie der Header-Datei Betriebsmittelverwaltung.h. In der Header-Datei wird die Zuordnung bzw. Festlegung der Systemressourcen vorgenommen.

5.2 Ressourcenbelegung

5.2.1 Shared Memory Variablen

Die Kommunikation zwischen den einzelnen Modulen sowie der Austausch von Daten wird über den Shared-Memory zwischen der Treiber- bzw. Sicherheitsschicht und zwischen der Sicherheits- bzw. Anwenderschicht vorgenommen. Hierfür werden in dem Modul Betriebsmittelverwaltung strukturierte globale Variablen deklariert.

In der Header-Datei Betriebsmittelverwaltung.h werden die nachfolgend aufgelisteten strukturierten Variablen angelegt:

Strukturierte Variable „Streckenbefehl“:

- Lok: byte
- Weiche: byte
- Entkoppler: byte
- Fehler: byte

Strukturierte Variable „Sensordaten“:

- Byte0: byte
- Byte1: byte
- Fehler:byte

Der Datentyp 'byte' wird aus dem Datentyp 'unsigned char' definiert. Dadurch wird ein durchgängig positiver Zahlenbereich von 0 bis +255 definiert.

Neben diesen strukturierten Variablen werden weitere globale Variablen angelegt, die von den anderen Modulen genutzt werden:

- LZ_BV_streckenbefehl: Streckenbefehl
- BV_LZ_sensordaten: Sensordaten
- BV_LZ_bestaetigung: byte
- S88_BV_sensordaten: Sensordaten
- BV_EV_streckenbefehl: Streckenbefehl
- SCC_EV_streckenbefehl: Streckenbefehl
- EV_SSC_streckenbefehl: Streckenbefehl
- SW_status_array: byte[6]
- EV_RS232_streckenbefehl: Streckenbefehl
- AS_msg_counter: byte

Um einen definierten Startzustand zu erhalten werden alle Variablen zu Beginn initialisiert.

Die Variablen vom Typ Streckenbefehl werden mit {LEER,LEER,LEER,0} initialisiert, Variablen vom Typ Sensordaten werden mit {LEER,LEER,0} initialisiert und die Variablen vom Typ byte werden mit LEER initialisiert.

5.2.2 Datentyp „Boolean“

Es wird ein Datentyp „Boolean“ aus dem Datentyp „bit“ definiert.

5.2.3 Interrupts

In die Implementierung wird die Belegung von folgenden Interrupts für Module nur als Kommentar angegeben.

- Interrupt 1 für SW
- Interrupt 4 für RS232
- Interrupt 18 für S88

Für die Nutzung der Interrupts müssen diese aktiviert werden (EAL).

5.2.4 Konstanten

Es werden folgende Konstanten im definiert:

- TRUE = 1
- FALSE = 0
- LEER = 255

5.2.5 Ports

Für die Benutzung der Ports werde diese folgenden Variablen zugewiesen:

- P3.4 = NOTAUS_PIN
- P3.5 = RS232TREIBER_CTSPIN
- P4.1 = (für SSC reserviert)
- P4.2 = (für SSC reserviert)
- P4.3 = (für SSC reserviert)
- P4.4 = (für SSC reserviert)
- P4.6 = (für CAN reserviert)
- P4.7 = (für CAN reserviert)
- P5.0 = S88_PS
- P5.1 = S88_RESET
- P5.2 = S88_CLK
- P5.3 = S88_DATA
- P5.4 = AS_PORT_I2C_SDA
- P5.5 = AS_PORT_I2C_SCL

5.3 Weitere Belegungen

5.3.1 Modul IDs

Der SW benötigt für die Zuordnung der Statusmeldungen zu den einzelnen Modulen eindeutige Modul-ID für jedes Modul. Diese werden als Konstanten in der Header-Datei Betriebsmittelverwaltung.h festgelegt. Vergeben werden die nachfolgend aufgelisteten IDs:

- LZ – 0
- BV – 1
- EV – 2
- S88 – 3
- SCC - 4
- RS232 – 5
- AS – 6

Vereinbart ist, dass die IDs folgendermaßen benannt werden:

- MODUL_<Modulkürzel>

5.3.2 Timer

Die Betriebsmittelverwaltung hat ein Timer-Modul zu enthalten, welches den anderen Modulen die Möglichkeit gibt ein Timerevent anzulegen. Ist die vom Modul angegebene Zeit verstrichen, so hat sich das Timer Modul zurück zu melden.

6 Dynamisches Verhalten

6.1 Initialsierungs-Aufrufe

Das dynamische Verhalten des Moduls „Betriebsmittelverwaltung“ kann in zwei unterschiedliche Abläufe unterteilt werden. Zu Beginn der Ausführung der main () Methode wird die Funktion initAll() aufgerufen. Damit werden sämtliche Module durch einen Aufruf der void init <Modulkürzel> () Methode initialisiert. Diese Methode ist Parameter und rückgabelos. Wichtig ist, dass dieser Aufruf zu Beginn ausgeführt wird. Andernfalls kann es zu Fehlern aufgrund ungültiger Variablen Zustände kommen. Die Bearbeitung der Fehler wird von den einzelnen Modulen vorgenommen und kann detailliert in dem jeweiligen Modul-Design nachgelesen werden (siehe Abschnitt 4: Referenzierte Dokumente).

Die zwingend einzuhaltende Inititialisierungsreihenfolge sieht folgendermassen aus:

- NOTAUS
- SW
- AS
- RS232
- S88
- SSC
- EV
- BV
- LZ
- FP

Nach jedem einzelnen Aufruf der void init <Modulkürzel> Methode wird über die Schnittstelle hello() des Moduls Software Watchdog der Software Watchdog zurückgesetzt.

6.2 Haupt Funktion/ Modul Aufrufe

Nach den Aufrufen der Initialisierungsmethoden wird in der main () die Methode work() aufgerufen. In dieser Funktion werden in einer endlosen For-Schleife die die einzelnen Module zum Arbeiten aufgerufen. Hierzu wird die Schnittstelle void work<Modulkürzel>() der einzelnen Module entsprechend einer vordefinierten Zeitscheibe aufgerufen.

Die Zeitscheibe ist in Abb.1 dargestellt.

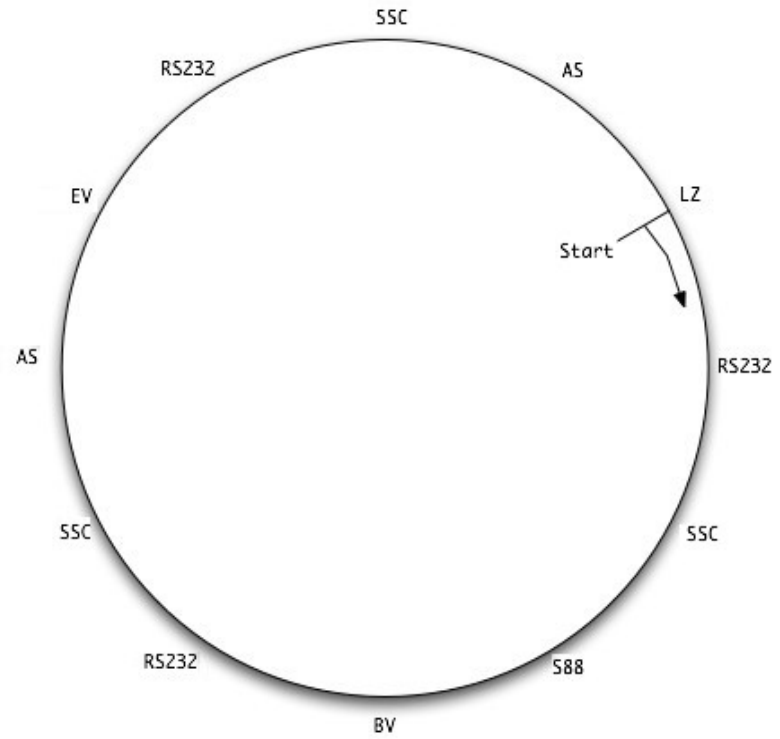

Abbildung 1: Zeitscheibe für die Modul Aufrufe

Begonnen wird mit dem Aufruf des Moduls Leitzentrale, siehe Startpunkt in der Abb.1. Ist die Zeitscheibe einmal vollständig durchlaufen, beginnt ein neuer Zyklus und die Aufrufe werden erneut vorgenommen.

Falls ein Modul sich erfolgreich zurückmeldet, so wird der Software Watchdog über die Schnittstelle hello() zurückgesetzt. Falls ein Modul sich nicht zurückmelden sollte, so wird das Rücksetzen des Software Watchdogs nicht vorgenommen. Der Software Watchdog generiert daraufhin eine Not-Aus.

Im Aktivitätsdiagramm in Abbildung 2 sind die Vorgänge dargestellt.

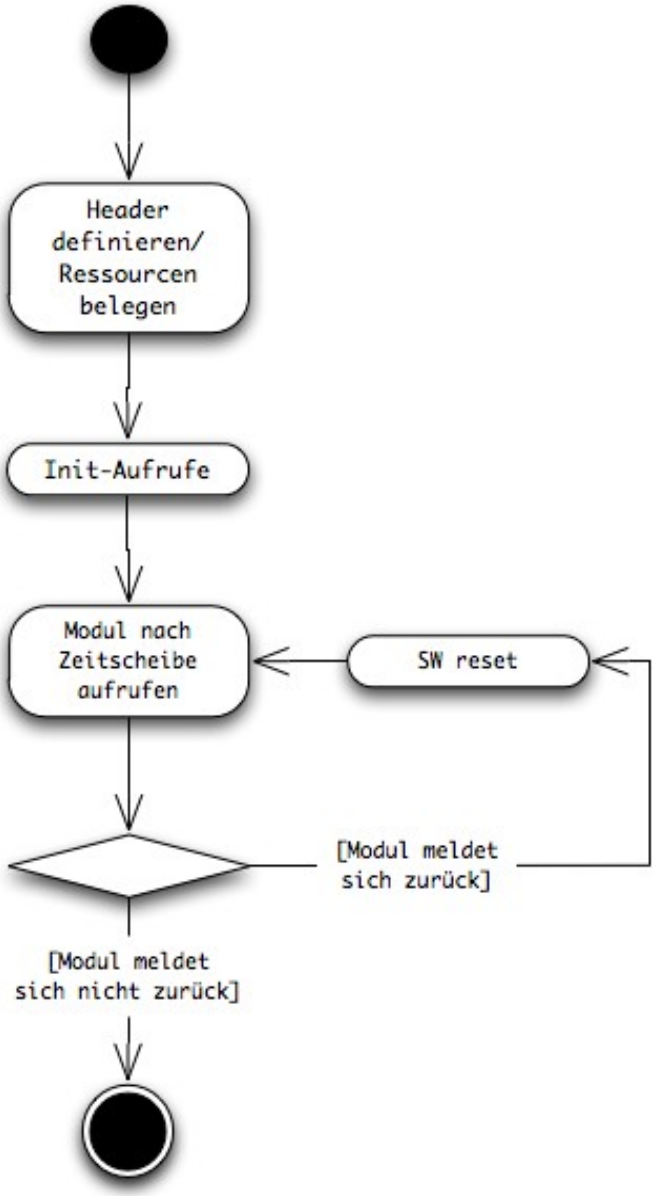

Abbildung 2: Aktivitätsdiagramm der Betriebsmittelverwaltung in main()