Universität der Bundeswehr München
Fakultät für Elektrotechnik und Technische Informatik
Wissenschaftliche Einrichtung 4 - Daten- und Schaltungstechnik

Prof. Dr.-Ing. Ferdinand Englberger
Prof. Dr.-Ing. Thomas Latzel

Dokumentation zur Schrittmotorsteuerung

Meilenstein 1a - Beschreibung der kompletten Steuersoftware

Autoren: Lt Marc Kossmann
Lt Michael Riedel
Gruppe: SoC04

Abgabedatum: 21.10.2014

Inhaltsverzeichnis

1. Planung des Projekts

Zur zeitlichen Planung wird das Gantt-Diagramm gemäß Abbildung 1.1 verwendet. Es stellt die zeitliche Abfolge der einzelnen Aufgaben, sortiert nach Meilenstein über die Dauer des Praktikums dar. Die aufgelisteten Aufgabe unterteilen sich in weitere Unteraufgaben, die zur Wahrung der Anschaulichkeit nicht aufgelistet werden.

Abbildung 1.2 zeigt den geplanten und benötigten Zeitaufwand für die Erstellung des Meilenstein 1a unterteilt in folgende Aufgabenbereiche:

- Einarbeitung
- Zeitplanung
- Design
- Implementierung
- Verifikation
- Dokumentation

Die Darstellung wird gesondert für die Studenten Marc Kossmann und Michael Riedel betrachtet. Diese Zeiten sind unabhängig von gemeinsam bearbeiteten Aufgaben. Die Abbildung 1.3 zeigt die komplette geplante und benötigte Zeit, die durch Aufsummierung der einzelnen Meilensteine entsteht.

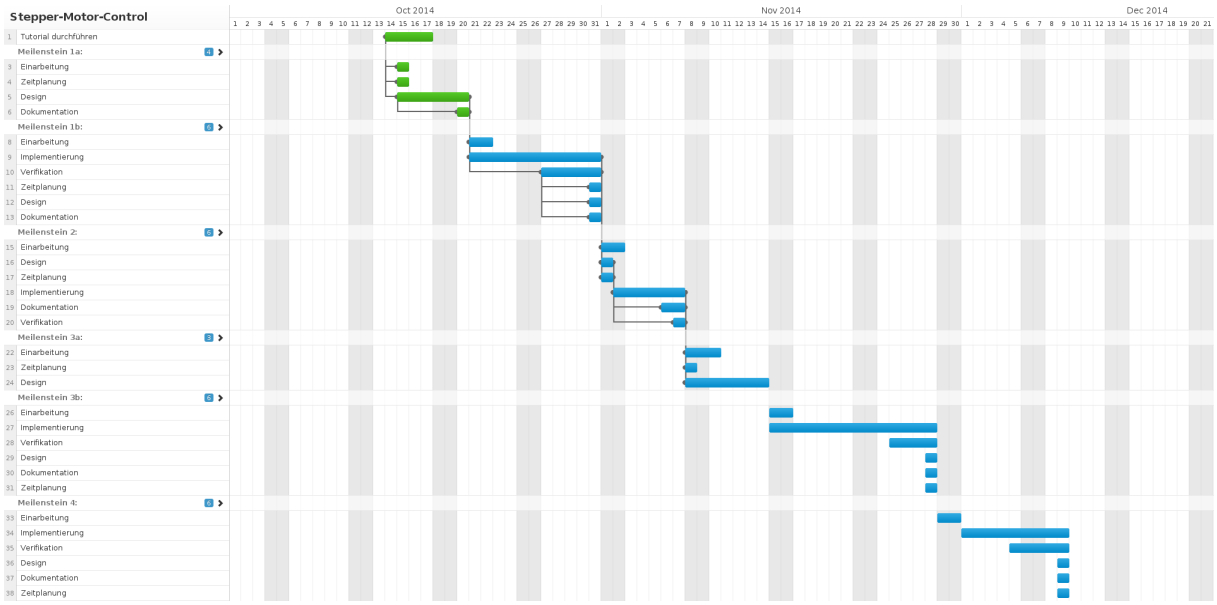

Abbildung 1.1: Gantt-Diagramm zur kompletten Zeitplanung

Meilenstein 1a

	Marc		Michael		gemeinsam		gesamt	
	geplant	benötigt	geplant	benötigt	geplant	benötigt	geplant	benötigt
Einarbeitung	-	3	-	2,15	7	5,7	7	10,85
Zeitplanung	-	1	-	5,25	3	3	3	9,25
Design	5	7	5	8,69	2	3,25	12	18,94
Implementierung	-	-	-	-	-	-	0	0
Verifikation	-	-	-	-	-	-	0	0
Dokumentaion	-	2	-	3	-	-	0	5
Summe	5	13	5	19,09	12	11,95	22	44,04

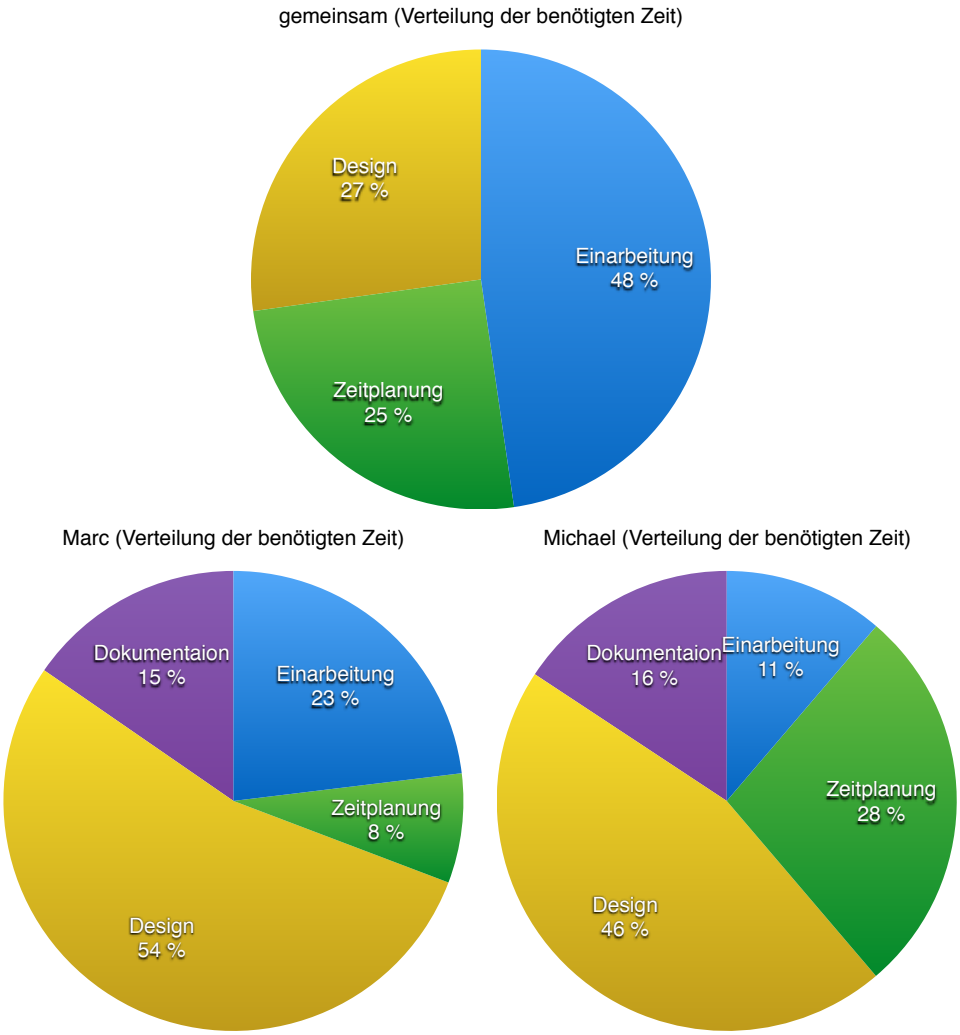

Abbildung 1.2: Projektplanung für Meilenstein 1a

Zeitbedarfsübersicht

	Marc	Michael	gemeinsam	gesamt
geplant	5	5	218	228
benötigt	13	19,09	11,95	44,04

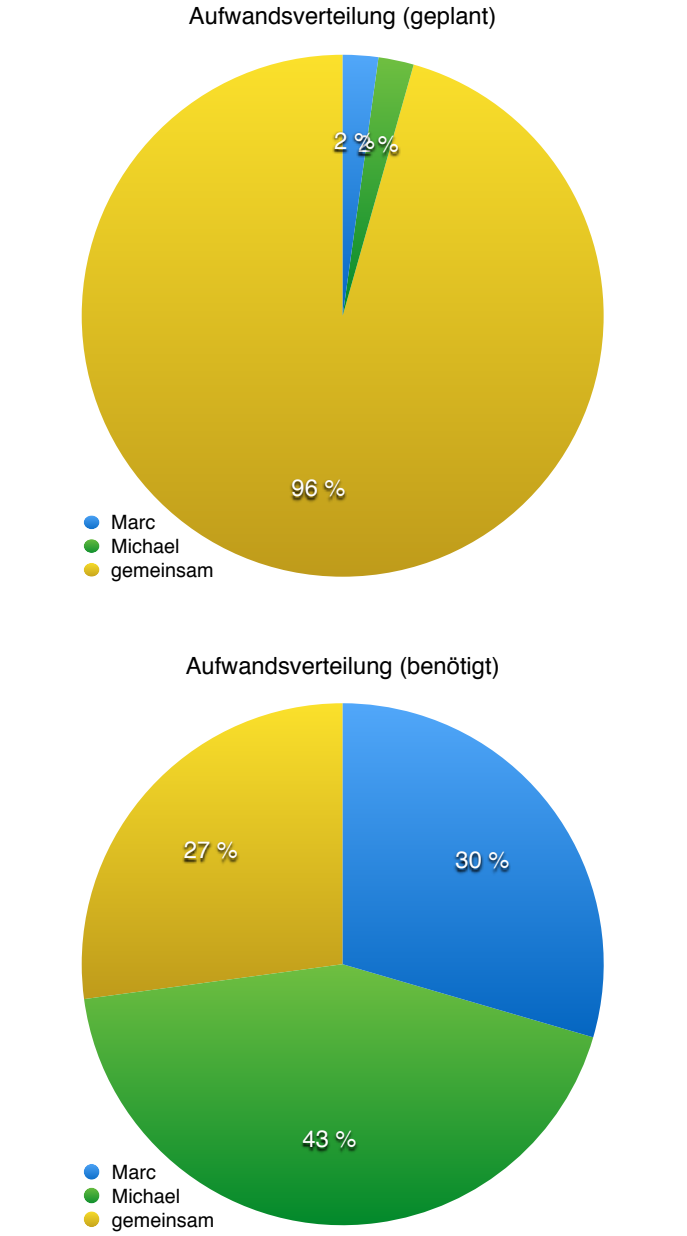

Abbildung 1.3: Zeitbedarfsübersicht für das gesamte Projekt

2. Beschreibung der Anwendungsfälle

Gemäß Abbildung 2.1 hat der Anwender (dargestellt als *User*) die Möglichkeit, den Modus der Steuerung (*mode*), die Drehrichtung (*direction*) und die Drehgeschwindigkeit (*speed*) einzustellen, sowie den Motor zu starten und zu stoppen (*start/stop motor*).

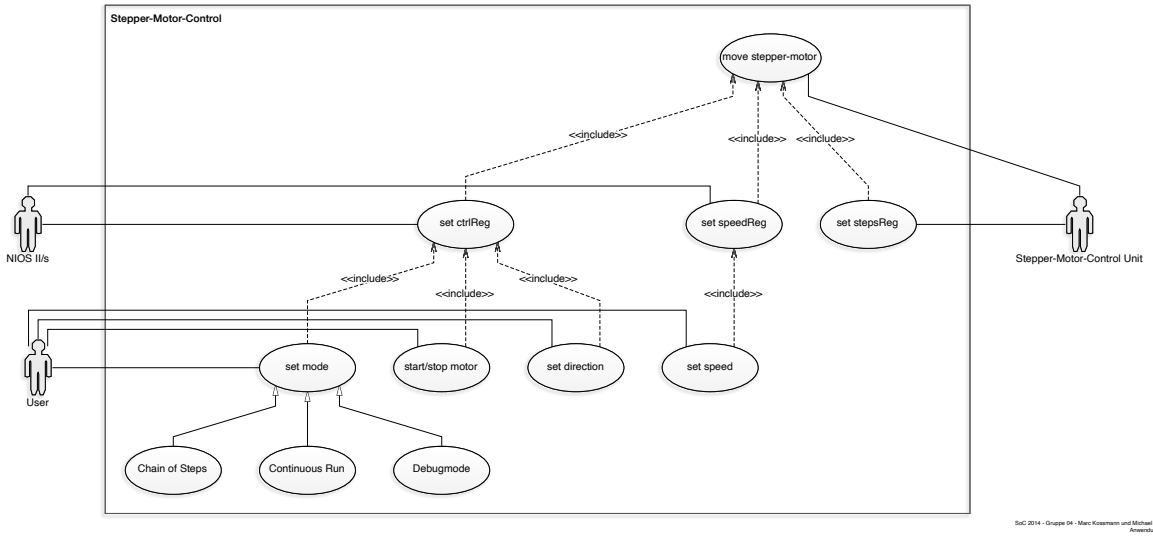

Abbildung 2.1: Anwendungsfälle

Der Anwender kann somit zwischen drei Modi wählen:

- *Chain of Steps*: Dabei wird der Motor um eine fest angegebene Schrittweite mit gewählter Geschwindigkeit verfahren.
- *Continuous Run*: Dabei verfährt der Motor mit vorher eingestellter Geschwindigkeit, bis er durch den Anwender gestoppt wird.
- *Debugmode*: Im Debug-Modus wird in regelmäßigen Abständen der Motor bewegt, ein Interrupt ausgelöst und die Register verändert.

Entsprechend den Eingaben durch den Anwender, werden durch den NIOS II/s-Prozessor die Register verändert und der Motor entsprechend durch die *Motor-Control-Unit (MCU)* verfahren.

3. Beschreibung der benötigten Tasks und Interrupt-Service-Routinen

Das Steuerprogramm zur Kontrolle des Schrittmotors reagiert auf Eingaben durch den Benutzer. Dies geschieht über verschiedene Schalter und Taster. Außerdem werden dem Benutzer während des Betriebs Informationen über ein LC-Display, eine *JTAG-UART Debug-Console* und LEDs angezeigt. Die Steuerung des Motors geschieht über eine eigene MCU, die anhand von in Registern abgelegten Informationen, bedient wird. Das Steuerprogramm und die MCU kommunizieren somit nur über die zur Verfügung stehenden Register und Interrupt-Leitungen.

3.1. benötigte Tasks

Damit Aufgaben parallel abgearbeitet werden können, werden drei Tasks verwendet:

3.1.1. Die Hauptsteuerung durch die User-Input-Task

Die „Main-Task“ oder auch `User-Input-Task` stellt, gemäß Abbildung 3.2, die höchste Kontrollinstanz dar. Nach der Initialisierung der Hardware, Interrupt-Service-Routinen (ISR) und der Ausgabe von Systeminformationen auf dem Display und in der *JTAG-UART Debug-Console* gemäß Abbildung 3.1, reagiert die Steuersoftware des NIOS II/s-Prozessors auf eingehende Benutzereingaben. Anhand derer werden, wenn nicht gerade der Motor läuft, die entsprechenden Register verändert.

Wird der Taster `Key 0` zum Starten des Motors gedrückt, so wird ein letztes Mal der Modus überprüft, sodass die MCU mit den Informationen der Register den Schrittmotor bewegen kann. Während der Motor läuft, ist vom Steuerprogramm nur die Drehrichtung veränderbar. Natürlich lässt sich der Schrittmotor jederzeit anhalten.

3.1.2. Die Ausgabe durch die User-Output-Task

Eine eigene `User-Output-Task` stellt, gemäß Abbildung 3.3 dem Benutzer Informationen zur Verfügung. Sie zeigt auf dem LC-Display neben der Programmversion, den Motormodus und Motorstatus an. Außerdem ist bei eingeschaltetem *Debug*-Modus dies kenntlich gemacht. Weil im Display nicht alle Informationen übersichtlich dargestellt werden können, werden

zusätzlich die HEX-Anzeigen 0 bis 2 zur Anzeige der Drehrichtung, der Geschwindigkeit und es Modus benutzt.

Währenddessen werden in einem über die JTAG-UART-Schnittstelle angeschlossenen Terminal Meldungen dargestellt. Diese enthalten alle Inhalte der Register zur Kommunikation mit der MCU.

3.1.3. Die Heartbeat und Debug-Task

Eine Forderung an die Steuersoftware ist die Anzeige eines *Heartbeats*. Damit lässt sich gemäß Abbildung 3.4 jederzeit erkennen, ob der Prozessor noch in einem funktionsfähigen Betriebszustand ist. Dieser Heartbeat wird über eine eigene Task namens `Heartbeat-Task` erzeugt und auf der HEX-Anzeige 3 und der LED 9 durch einen Blinkcode dargestellt.

Wenn der *Debug-Schalter* auf `1` gestellt wurde, wird in der *Heartbeat*-Task alle 3 Sekunden ein Interrupt erzeugt, das dem Steuerprogramm das Anhalten des Motors signalisiert. Wenn das `Run-Bit` im Control-Register (`ctrlReg`) auf `1` gesetzt ist, sich der Schrittmotor also dreht, wird entsprechend der Drehrichtung das Register `stepsReg` zum zählen der Schritte verändert. So lässt sich das Programm ohne MCU testen.

3.2. benötigte Interrupt-Service-Routinen

Um eine schnelle Interaktion mit der Schrittmotorsteuerung durch den Anwender zu ermöglichen, werden drei ISR verwendet:

3.2.1. Die Abfrage der Taster

Damit eine Reaktion auf Benutzereingaben in Echtzeit möglich ist, werden die Taster gemäß Abbildung 3.5 über eine `Key-ISR` ausgewertet. Sobald ein Taster gedrückt wurde, wird der Interrupt gesetzt und im *Interrupthandler* der Zustand der Taster abgefragt. Für den gedrückten Taster wird dann ein *Flag* an die Main-Task gesetzt.

3.2.2. Die Abfrage der Schalter

Die `Switch-ISR` reagiert gemäß Abbildung 3.6 auf Änderungen der Schalterstellungen. Zur Einsparung von Flags werden die Schalter etwas anders ausgewertet. Bei Betätigung wird auch ein Interrupt ausgelöst, allerdings wird im *Interrupthandler* die Stellung aller Schalter in einer *MessageQueue*-Variable abgespeichert und anschließend nur ein `Switch-Update -Flag` an die Main-Task gesetzt. Diese muss dann die Schalterstellungen aus der Queue abholen.

3.2.3. Die Abfrage der Motorsteuerung

Zu guter Letzt wird gemäß Abbildung über die `Motor-ISR` das Stoppen des Motors in der `Chain of Steps`-Betriebsart signalisiert. Im *Interrupthandler* wird nur ein Flag an die Main-Task gesendet. Diese ISR ist mit dem `IR`-Bit des Control-Registers (`ctrlReg`) verbunden.

3.3. Darstellung der Aktivitätsdiagramme

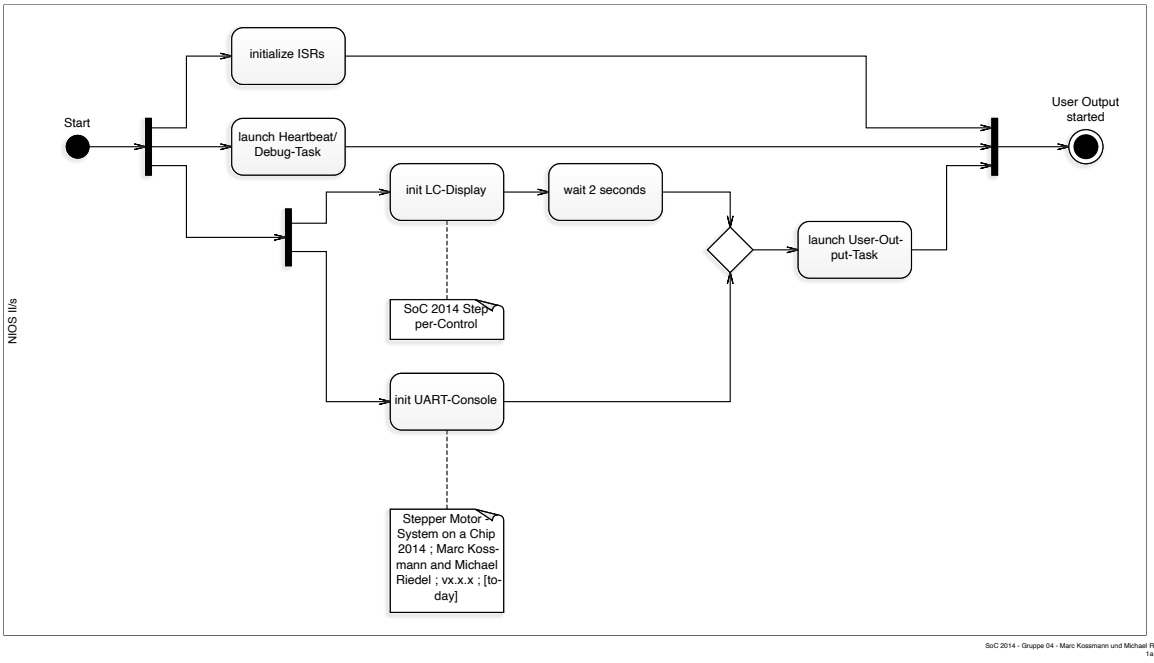

Abbildung 3.1: Initialisierung der Hardware, ISR und Tasks

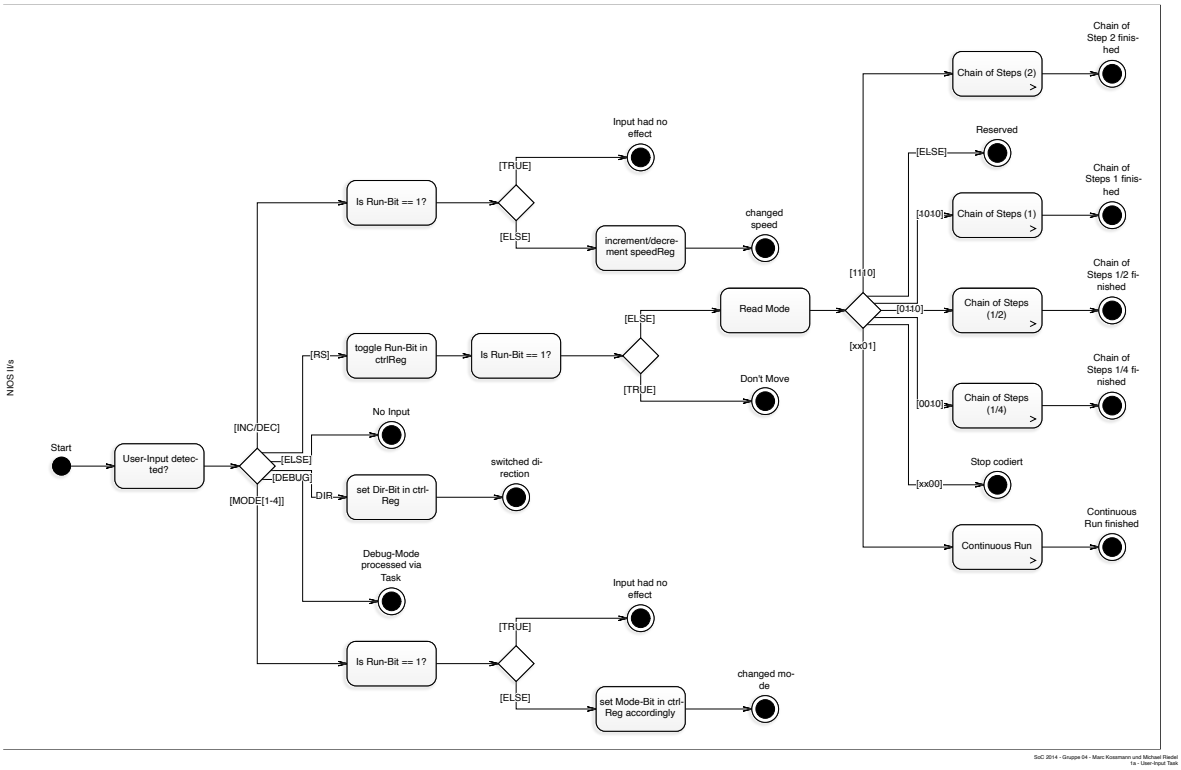

Abbildung 3.2: User-Input Task

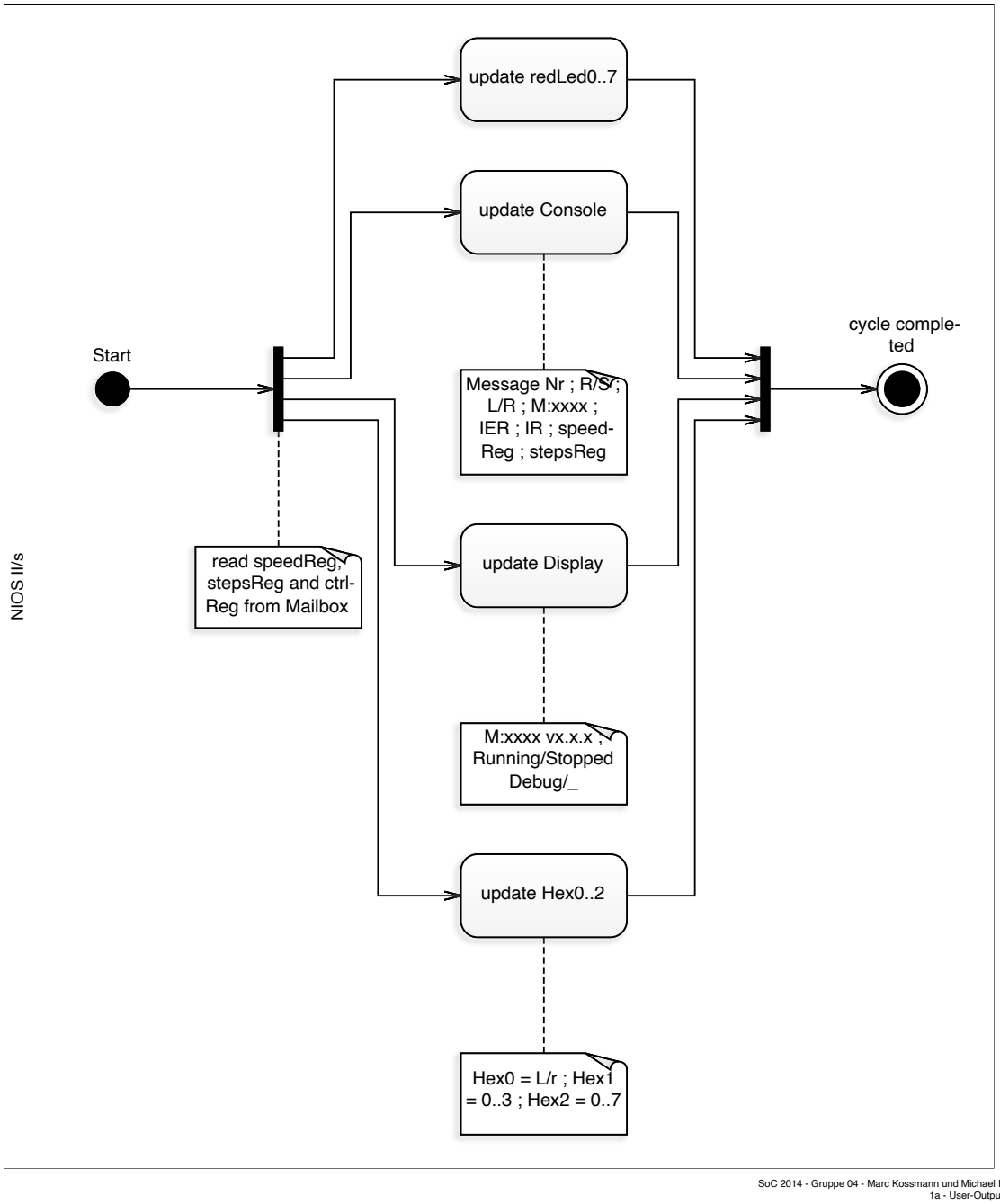

Abbildung 3.3: User-Output Task

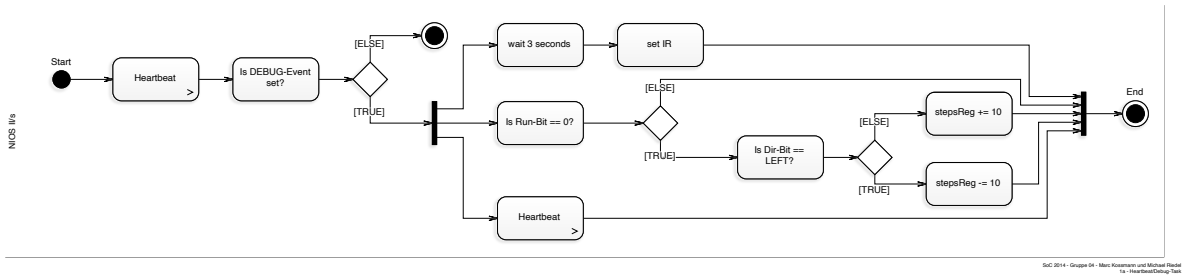

Abbildung 3.4: Heartbeat-Debug Task

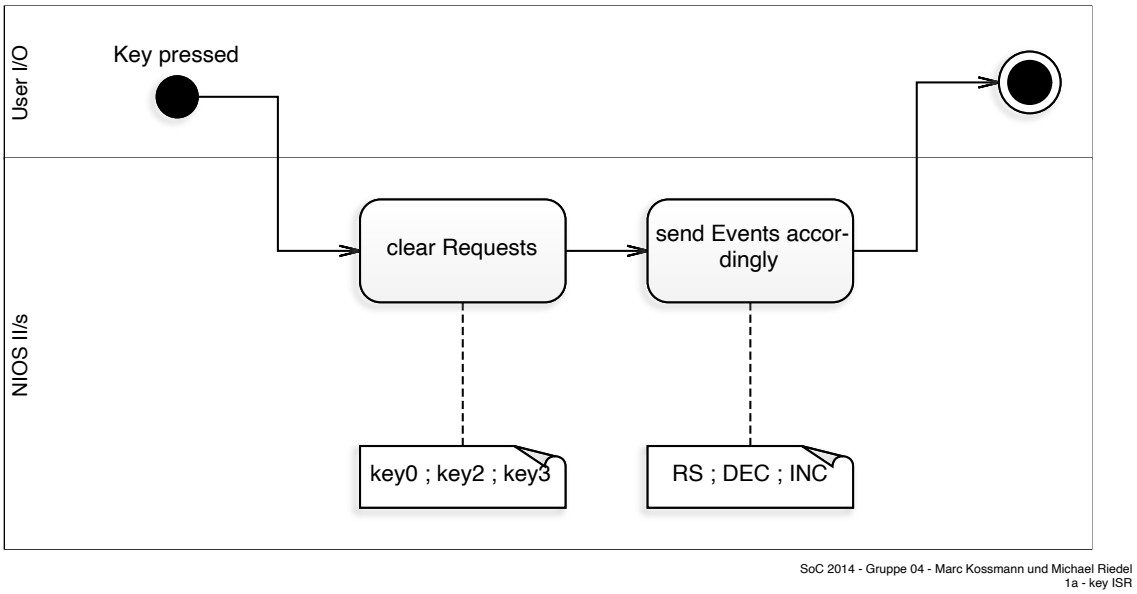

Abbildung 3.5: Key ISR

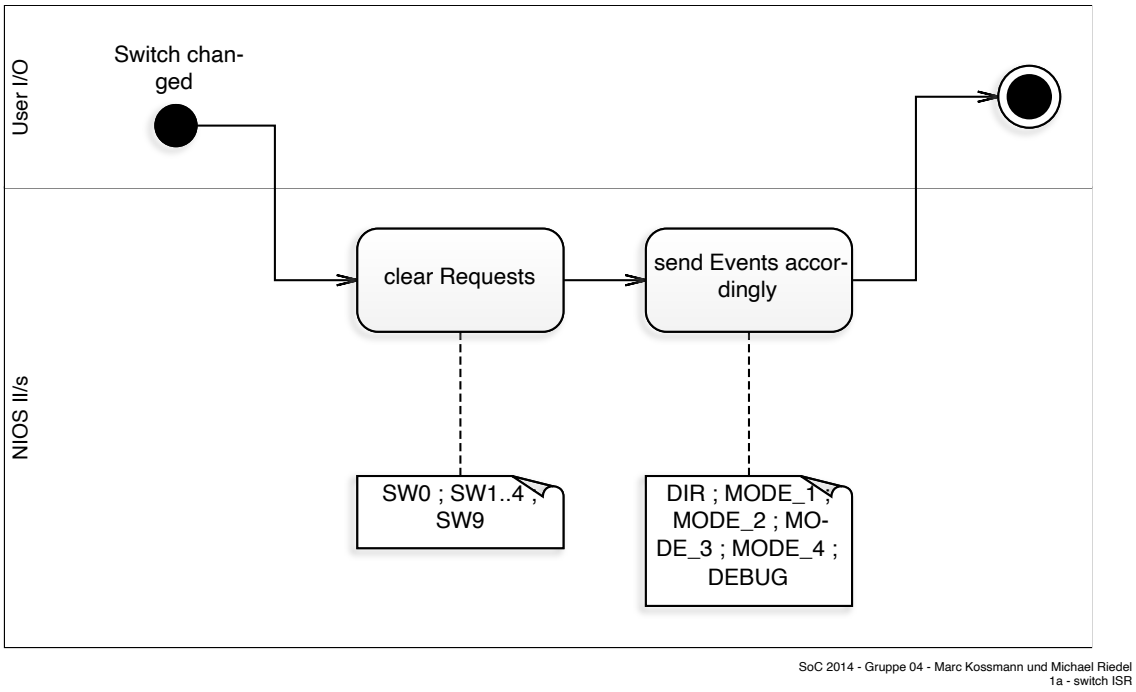

Abbildung 3.6: Switch ISR

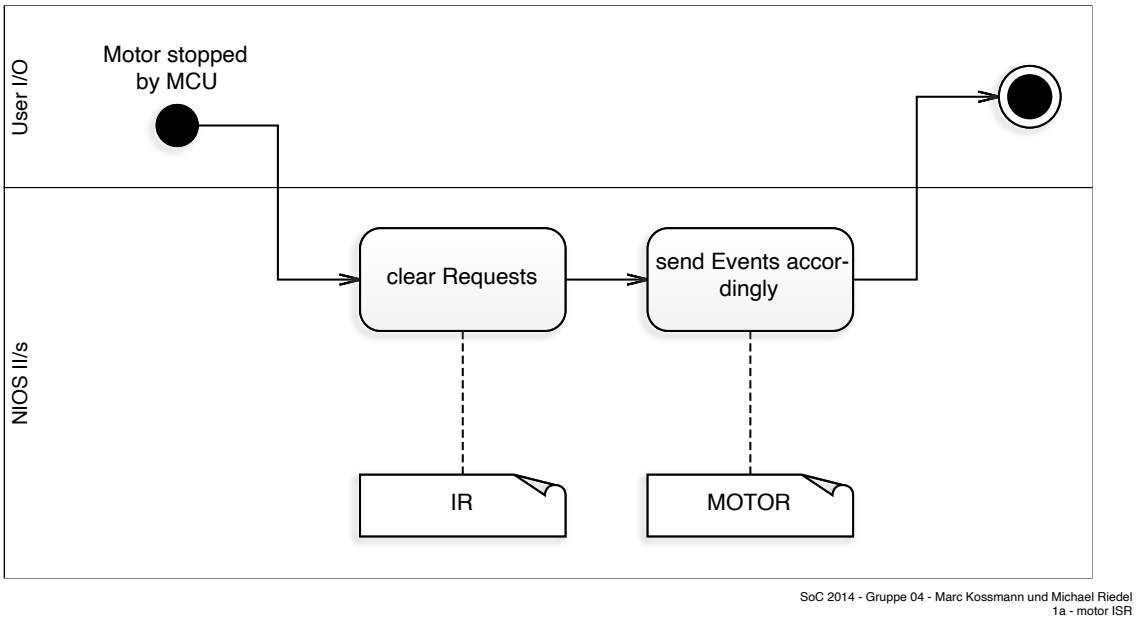

Abbildung 3.7: Motor ISR

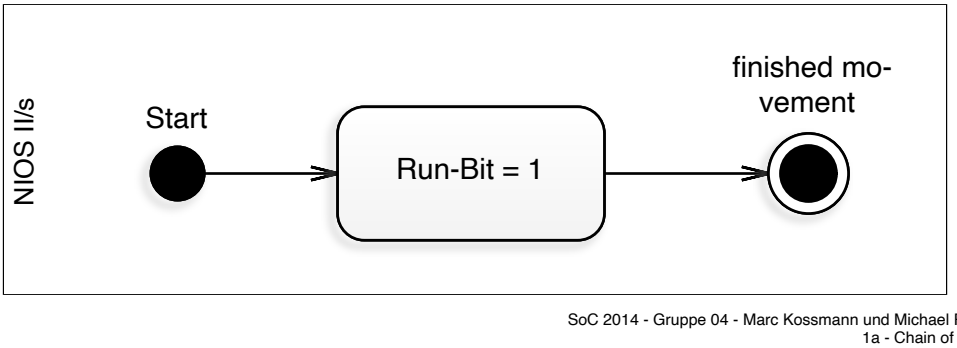

Abbildung 3.8: Chain of Steps Funktion

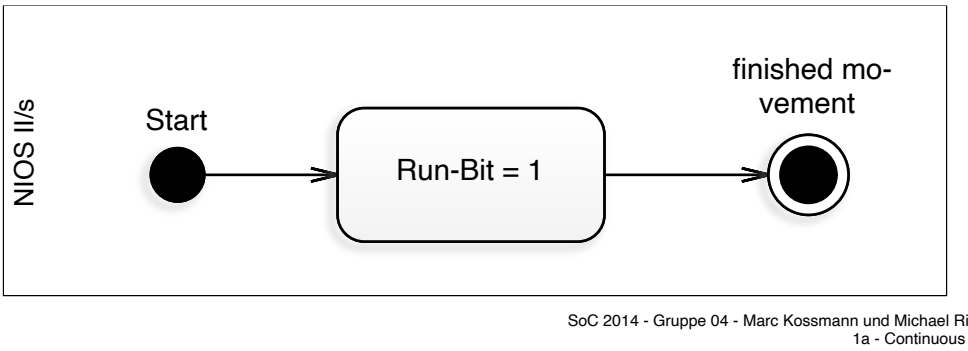

Abbildung 3.9: Continuous Run Funktion

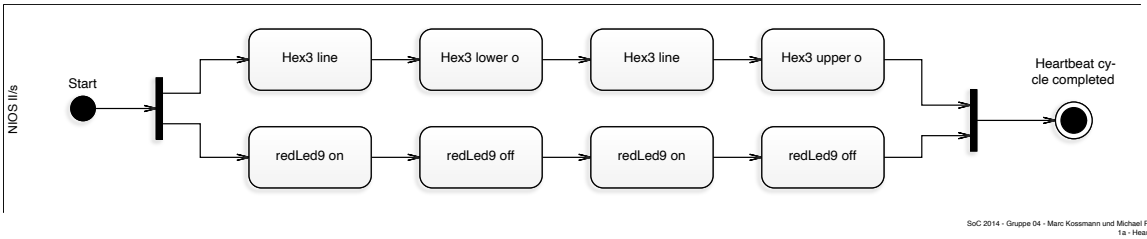

Abbildung 3.10: Heartbeat Funktion

4. Weitere Darstellungen zur Erläuterung der internen Kommunikation

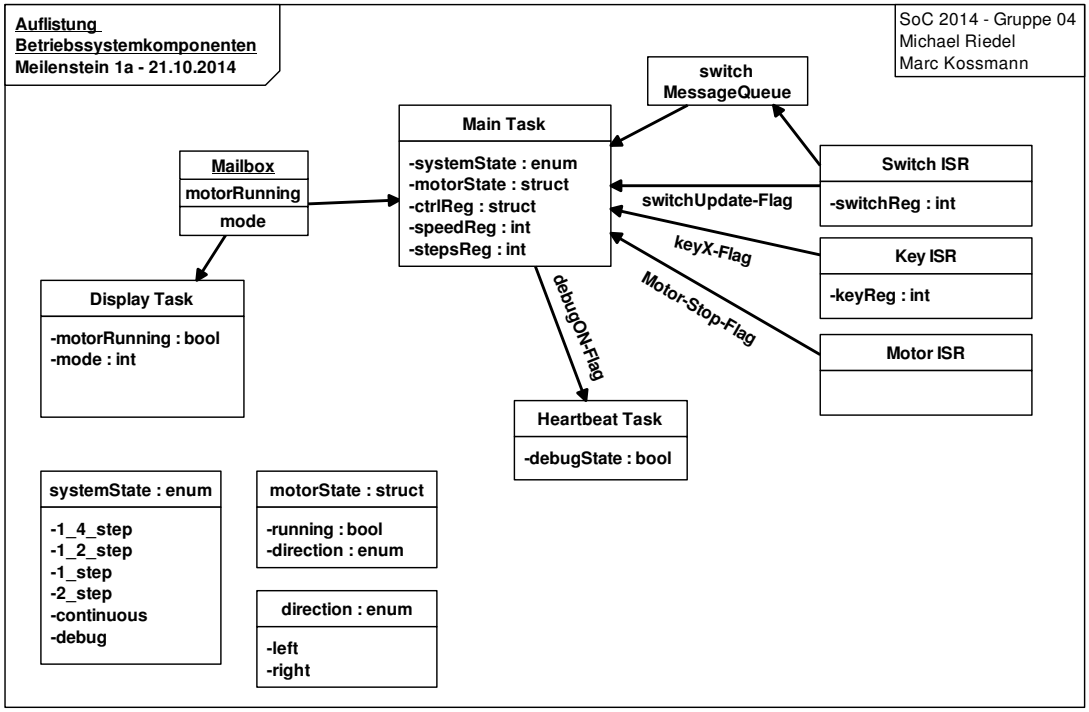

Abbildung 4.1: Auflistung Betriebssystemkomponenten

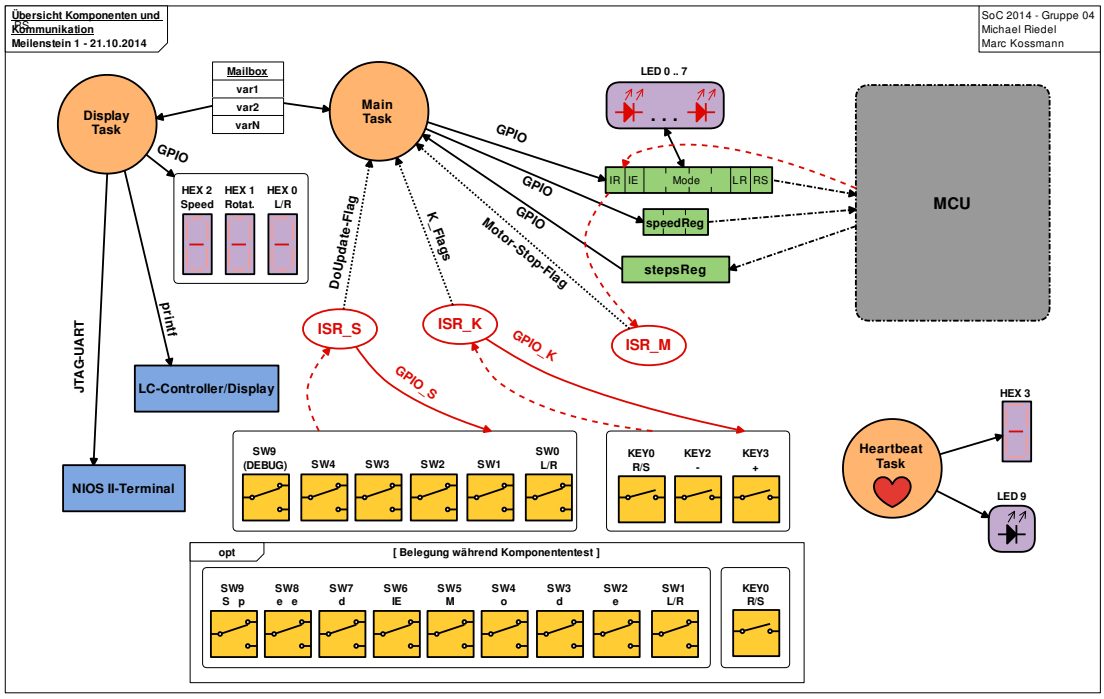

Abbildung 4.2: Übersicht der Komponenten und Kommunikation

Abbildungsverzeichnis