h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

MIKROPROZESSORPRAKTIKUM

SS2023

Termin1

C-Programmierung für eingebettete Systeme

Name, Vorname	Matrikelnummer	Anmerkungen
Datum	Raster (z.B. Mi3x)	Testat/Datum

Legende: V:Vorbereitung, D: Durchführung, P: Protokoll/Dokumentation, T: Testat

Laborordnung

Sinn dieser Laborordnung ist die Festlegung von Regeln für die Benutzung der Labore in D10

Jeder ordentliche Student des Fachbereich wird in den Grundlagenveranstaltungen auf die gültige Laborordnung und die Brandschutzregeln hingewiesen.

1. Der Notausschalter des Labors D10/0.32 befindet sich über dem Lichtschalter. Der Notausschalter ist im Notfall zu betätigen, um sämtliche elektrische Geräte stromlos zu schalten. Achten Sie bei der Einführung auf die entsprechenden Hinweise.
2. Im Labor darf maximal Paarweise an den Geräten gearbeitet werden. Das Labor ist für 8 Gruppen ausgelegt.
3. Es ist nicht gestattet sich alleine im Labor aufzuhalten.
4. Die Ersthelfer sind Herr Rudi Scheitler (Raum 0.33 / Tel.: 68465), Herr Manfred Pester (Raum 0.33 / Tel.: 68428) und Herr Sergio Vergata (Raum 0.37 / Tel.: 68491). Wenden Sie sich bitte im Falle einer Verletzung direkt an eine dieser Personen. Erste-Hilfe-Materialien befindet sich im Eingangsbereich Südseite.
5. Es ist nicht gestattet Kabel zu entfernen, Gehäuse zu öffnen und Hardware (außer USB-Sticks) zu installieren oder Änderungen an der Laborinfrastruktur vorzunehmen. Sollte etwas nicht funktionieren, oder es wird etwas benötigt, welches die vorhandene Infrastruktur nicht abdeckt, so wenden Sie sich an den Betreuer des Labors oder direkt an den zuständigen Laboringenieur Manfred Pester (Raum 0.33 / Tel.: 68428).
6. Fahren Sie die von Ihnen benutzten Geräte am Ende Ihres Praktikum/Ihrer Übung herunter und schalten diese aus, es sei denn Sie bekommen vom zuständigen Betreuer andere Anweisungen.
7. Speisen und Getränke sind an den Arbeitsplätzen im Labor nicht erlaubt.
8. Bei der Benutzung des Labordrucker ist Sorgfalt und Sparsamkeit oberstes Gebot.
9. Evtl. ausgestellte Dokumentationen dienen der Laborarbeit und müssen im Raum verbleiben.
10. Die Benutzung von Mobiltelefonen ist untersagt. Schalten Sie vor dem Betreten des Raumes die Geräte ab (Flugzeugmodus ist ok). In dringenden Fällen können Sie sich über das Labortelefon mit der Nummer 06151 53368433 anrufen lassen.
11. Hängen Sie Ihre Kleidung (Mäntel, Jacken, ..) an die dafür vorgesehenen Kleiderständer und nicht über die Stühle.
12. Deponieren Sie Taschen, Laptops u.s.w. nicht in den Gängen, sondern möglichst an den Seiten des Labors oder unter den Tischen.
13. Verlassen Sie Ihren Arbeitsplatz aufgeräumt! Müll gehört in die mehrfach vorhandenen Mülleimer, Altpapier in die dafür vorgesehene blaue Altpapierwanne.
14. Die Fluchtwege sind frei zu halten.

Bei Verstößen gegen die Laborordnung kann die Benutzungsberechtigung versagt werden.

Lernziele:

Mit den folgenden Versuchen sollen Sie die Sprache "C" einmal aus einer anderen Sicht kennen lernen. An ganz einfachen Programmen sollen Sie ermitteln, welchen Code ein Compiler erzeugt, wo welche Variablen abgelegt werden, welchen Einfluss die Optimierungsstufen haben, wie ein c*all by value / reference* in ARM Assembler umgesetzt wird. Wie auf Peripherie (System on Chip) zugegriffen werden kann.

Arbeitsverzeichnis:

Kopieren Sie sich das Verzeichnis mpsSS2023. Dort stehen im Unterverzeichnis Termin1 die Dateien *Termin1AufgabeX.c als* Programmgerüstbeispiele und ein *makefile* zur Verfügung.

Die Aufgaben 1 bis 4 dieses Termins können wie in der Veranstaltung Rechnerarchitektur gelernt auch im Target Simulator (also ohne die im Labor zur Verfügung gestellte Hardware) gelöst werden.

Aufgabe 1:

Legen Sie im C Programm zwei lokale integer Variablen an. Eine Variable initialisieren Sie mit einem Wert. Weisen Sie der zweiten Variablen im Code einen Wert zu. Übersetzen Sie Ihr Programm und schauen Sie sich den erzeugten Maschinen-Code (Assembler) an. Führen Sie das Programm im Debugger aus. Dokumentieren Sie den erzeugten Code.

Hinterfragen Sie Ihre Beobachtungen z. B. mit:

- Wo liegen die Werte der Variablen im Speicher?
- Wie kommen die Werte in den Speicher?
- Welche Register werden verwendet?
- Wie wird auf die Variablen zugegriffen?
- ..

Ändern Sie im Makefile die Optimierungsstufe und beobachten und dokumentieren Sie die Veränderungen.

Aufgabe 2:

Verwenden Sie nun statt der lokalen Variablen globale (initialisierte und nicht initialisierte) Variablen.

- Was ändert sich?
- ..

Ändern Sie auch nun die Optimierungsstufe und beobachten und dokumentieren Sie die Veränderungen.

Aufgabe 3:

Legen Sie nun zwei globale und zwei lokale integer Variablen an. Weisen Sie den Variablen Werte zu. Machen Sie nun Zuweisungen der Form „global=lokal" und/oder „lokal=global".

- Was stellen Sie fest?
- Wo liegen die Werte der Variablen im Speicher?
- ..

Ändern Sie im Makefile die Optimierung und beobachten und dokumentieren Sie die Auswirkungen.

Aufgabe 4:

Vereinbaren Sie einen Funktionsprototypen für eine Funktion "int addition(int, int, int)", die 3 integer Parameter erwartet. Rufen Sie diese Funktion im Anschluss an die Zuweisung nach Aufgabe 2 und 3 mit den beiden lokalen und einer globalen Variablen auf. Dokumentieren Sie Ihre Beobachtungen. Ändern Sie im Makefile die Optimierung und beobachten und dokumentieren Sie die Auswirkungen.

Aufgabe 5:

Wir werden uns nun mit Registern der PIO des hier eingesetzten Mikrocontroller beschäftigen. Siehe in den zur Verfügung stehenden Unterlagen. Im Datenblatt Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf, im Due-Pinout.pdf und in den Unterlagen zur Vorlesung finden Sie Informationen.

Name	Adresse	Bedeutung
PIOB_PER	0x400E1000	PIOB Port Enable Register
PIOB_OER	0x400E1010	PIOB Output Enable Register
PIOB_SODR	0x400E1030	PIOB Set Output Data Register
PIOB_CODR	0x400E1034	PIOB Clear Output Data Register

Schreiben Sie zunächst den Wert 0x8000000 ins PIOB_PER und dann ins PIOB_OER. Danach schreiben Sie nacheinander den Wert 0x8000000 einige Male alternierend ins PIOB_SODR und

PIOB_CODR. Dadurch sollte die LED L auf dem Board Arduino DUE an und aus gehen. Testen Sie die Funktionalität im Debugger mit Einzelschritten aus. Im gegebenen Programmbeispiel Termin1Aufgabe5.c sind verschiedene Möglichkeiten des Beschreibens/Lesens der Register der PIO gezeigt. Beschäftigen Sie sich mit den verschiedenen Programmiermöglichkeiten.

- Welche Variante würden Sie bevorzugen und warum?
- Beobachten Sie wann die LED an und aus geht. Schauen Sie sich hierzu die Doku an. Beschreiben Sie Ihre Beobachtungen/Erkenntnisse.

Aufgabe 6 Zusatzaufgabe:

Versuchen Sie nun Ihr Programm auf minimale Programmgröße zu bringen.

- Wie viele Byte Speicher benötigen Sie für die Initialisierung und um die LED in einer Endlosschleife blinken zu lassen?
- Aus wie vielen Assemblerbefehlen besteht Ihr minimiertes Programm?
- Diskutieren Sie ob sich der Optimierungsaufwand lohnt.

Aufgabe 7:

Erstellen Sie zu diesem Termin ein Protokoll mit den Lösungen zu den Aufgaben und Ihren Erkenntnissen. Das Protokoll sollen Sie zu den nächsten Terminen vorlegen können. Denken Sie auch daran, dass Sie Programmteile in den nächsten Praktika nochmals benötigen. Beschäftigen Sie sich baldigst auch mit den Aufgaben der nächsten Termine. Im letzten Termin sollten Sie in der Lage sein, die gestellte Projektaufgabe zu realisieren.

Beispiel eines einfachen Makefile für die zu lösenden Aufgaben:

```
FILE = Termin1Aufgabe1
OPTI = 0

build:
	mkdir -p build
	arm-none-eabi-gcc -c -gdwarf-2 -mcpu=cortex-m3 -mlittle-endian -mthumb -O$(OPTI) -I../CMSIS_5/CMSIS/Core/Include/ -I../Atmel.SAM3X_DFP.1.0.50/include/ -I./include -D__SAM3X8E__ -obuild/$(FILE).o src/$(FILE).c
	arm-none-eabi-gcc -c -gdwarf-2 -mcpu=cortex-m3 -mlittle-endian -mthumb -O$(OPTI) -I../CMSIS_5/CMSIS/Core/Include/ -I../Atmel.SAM3X_DFP.1.0.50/include/ -I./include -D__SAM3X8E__ -obuild/system_sam3xa.o ../Atmel.SAM3X_DFP.1.0.50/gcc/system_sam3xa.c
	arm-none-eabi-gcc -c -gdwarf-2 -mcpu=cortex-m3 -mlittle-endian -mthumb -O$(OPTI) -I../CMSIS_5/CMSIS/Core/Include/ -I../Atmel.SAM3X_DFP.1.0.50/include/ -I./include -D__SAM3X8E__ -obuild/startup_sam3xa.o ../Atmel.SAM3X_DFP.1.0.50/gcc/gcc/startup_sam3xa.c

flash: build
	arm-none-eabi-ld --gc-sections -L ../Atmel.SAM3X_DFP.1.0.50/gcc/gcc/ -T sam3x8e_flash.ld -emain -o build/$(FILE).elf build/$(FILE).o build/system_sam3xa.o build/startup_sam3xa.o

sram: build
	arm-none-eabi-ld --gc-sections -L ../Atmel.SAM3X_DFP.1.0.50/gcc/gcc/ -T sam3x8e_sram.ld -emain -o build/$(FILE).elf build/$(FILE).o build/system_sam3xa.o build/startup_sam3xa.o

insight:
	arm-none-eabi-insight build/$(FILE).elf

debug:
	cp .gdbinit ~/.gdbinit
	arm-none-eabi-gdb build/$(FILE).elf -ex "target remote localhost:3333" -ex "load" -ex "break main"

clean:
	rm -r build

openocd:
	sudo openocd -s /usr/share/openocd/scripts -f interface/cmsis-dap.cfg -f board/atmel_sam3x_ek.cfg

gdb:
	cp .gdbinit ~/.gdbinit
	arm-none-eabi-gdb -ex "target remote :3333" -ex "load ./build/$(FILE).elf" -ex "symbol-FILE ./build/$(FILE).elf" ./build/$(FILE).efl
```