h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

PRAKTIKUM EINGEBETTETE SYSTEME

SS2024

Termin 0

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 wollen wir 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 ESEDASS2024 von den zur Verfügung gestellte Quellen.

Aufgabe 1 (Projektstart)

Starten Sie im Projektordner ESEDASS2024 VSCode durch Eingabe von “code .“ und machen Sie sich mit der IDE und den gegebenen Informationen vertraut.

Leitfragen:

- Welche Bedeutung haben die Unterordner: .vscode, cmsis und src?
- Welche Funktion hat/bietet das Makefile?
    - Welche PHONY-Targets sind definiert?
- Welche Toolchain wird verwendet?
- Wie ist die **main**-Funktion in main.c aufgebaut?

Aufgabe 2 (Debugging)

Bauen Sie das gegebene Projekt und stellen Sie eine Debug-Verbindung mit dem Mikrocontroller her. Gehen Sie hierfür in VSCode auf die Schaltfläche Run and Debug (CTRL+SHIFT+D) und führen Sie das Programm aus (F5). Überprüfen Sie, ob OpenOCD als Task im Hintergrund läuft. Dies sollte beim Öffnen des Workspace automatisch gestartet sein. Sollte dies nicht der Fall sein, kann dies über das Ausführen eines Tasks in VSCode oder über ein beliebiges Terminal im Projektordner durch das PHONY-Target “make openocd“ nachgeholt werden.

Leitfragen:

- Wofür wird OpenOCD verwendet?
- Wird das Programm ausgeführt?
- Funktionieren Breakpoints?
- Welchen Einfluss könnten verschiedene Optimierungsstufen haben?

Aufgabe 3 (Serielle Schnittstelle)

Stellen Sie mithilfe des Terminalprogramms **Minicom** eine serielle Verbindung zum Mikrocontroller her. Konfigurieren Sie Minicom (minicom -s) mit einer Baudrate von 115200, 8 Datenbits und keiner Parität (115200 8N1).

1. Inkludieren Sie serial.h in das Programm. (#include <serial.h>)
2. Initialisieren Sie die serielle Schnittstelle mit serial_init. (Rufen Sie serial_init innerhalb der init Funktion auf.)
3. Erzeugen Sie eine Ausgabe auf dem Terminal durch serial_transmit_string innerhalb der loop Funktion.

minicom -D /dev/ttyACM0

Aufgabe 4 (Menu)

Ihnen wird bereits ein Programmcode für eine Textbenutzeroberfläche (text user interface TUI) bereitgestellt. Diese werden wir im Verlauf der Praktika verwenden, um mit dem eingebetteten System zu interagieren und Daten z. B. von Sensoren auszulesen.

1. Inkludieren Sie menu.h in das Programm. (#include <menu.h>)
2. Initialisieren Sie die serielle Schnittstelle mit menu_init. (Rufen Sie menu_init innerhalb der init Funktion auf.)

3. Erzeugen Sie eine kontinuierliche Ausgabe des Menüs auf dem Terminal durch Aufruf der Funktion menu_loop innerhalb der loop Funktion.

Aufgabe 5 (Clock)

Unser Ziel ist es nun, unserer ersten Menükomponente Funktionalität zu verleihen. Hierfür werden wir die erste Peripheriekomponente einsetzen – konkret handelt es sich um einen Timer-Zählerblock. Der entsprechende Quellcode wird Ihnen bereitgestellt. Wir werden diesen verwenden, um verschiedene Metriken zu erfassen, beispielsweise die Frequenz, mit der die Haupt-Schleife (loop) ausgeführt wird.

1. Inkludieren Sie clock.h in das Programm. (#include <clock.h>)
2. Initialisieren Sie den Timer mit clock_init. (Rufen Sie clock_init innerhalb der init Funktion auf.)
3. Rufen Sie clock_loop zu Beginn der loop Funktion auf, um die Messdaten zu erfassen, die im Menü angezeigt werden sollen.

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:
	rm ~/.gdbinit
	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).elf
```