Hardwaresimulation

Zur Simulation der Hardware mit einem AT91M63200-Mikrocontroller setzen wir eine Softwaresimulation ein. Diese führt die Schritte in Software aus, die sonst die Hardware der Peripheriebausteine des Mikrocontrollers durchführt. Hierzu finden Sie eine Anzahl von Dateien in dem Verzeichnis ESSim.

Vorbereitung

Verschaffen Sie sich einen Überblick über die einzelnen Programmmodule in dem Videomodul **PS-UE** des Moodle-Kurses.

Öffnen Sie die die IDE Codeblocks und hiermit das Projekt `ESSim3.cbp`.

Vergleichen Sie die Module (*.h und *.c) des Projekts mit den Inhalten des Verzeichnisses ESSim. Welchen Inhalt hat das Verzeichnis AT91/h?

Die Datei `sim_config.h` enthält eine Reihe von Konfigurationsmöglichkeiten. Verschaffen Sie sich einen Überblick über die Konfiguration und Tracing-Möglichkeiten in dem Videomodul **PS-KO**.

Die Dateien `stb_*.[h|c]` enthalten Deklaration und Implementierung der Simulation des Verhaltens der Peripheriebausteine sowie einer an den Timerbaustein angeschlossenen Waage. Verschaffen Sie sich einen Überblick über den Programmcode in dem Videomodul **PS-WT**.

Einige Dateien enthalten die Möglichkeit des Tracings in eine Datei. Schauen Sie das Videomodul **PW-TR**, wie das Tracing realisiert ist.

1. Simulator für Peripheriebausteine

Wir verwenden im Weiteren als Konfiguration die Zielarchitektur **`TRGT_CLI`** im Programmcode des Simulators ESSim. Stellen Sie sicher, dass diese als Präprozessormakro (z.B. in Codeblocks unter Projekt -> Build Options -> #defines) definiert ist.

1A)

- Für welche Prozessorarchitektur wird der Maschinencode der ausführbaren Datei erzeugt? Welche C-Library wird durch das Programm verwendet, so dass Konsolenausgaben möglich werden?

Die Peripheriebausteine des AT91M63200-Mikrocontrollers sollen simuliert werden. Hierzu konstruiert die Simulationsumgebung mittels des Makros `STB_CONSTRUCT` (siehe `stb_at91.h`) einen Adressraum, in dem die simulierten Hardwareregister liegen.

1B)

i. Analysieren Sie die Datei `stb_mmio.h`. Wie werden die Basisadressen der Peripheriebausteine umdefiniert? Warum wird dies nicht im Verzeichnis AT91/h geändert?

ii. In welcher Funktion wird der Speicher für die Peripheriebausteine allokiert? An welcher Adresse liegt dieser?

iii. Was für eine Art an Arbeitsspeicher blendet ihr Betriebssystem für den Simulator ein? Nutzen Sie „`man` *`systemfunktion`*“ für Ihnen unbekannte Bibliotheksfunktionen.

1C)

- Was führt das Makro `STB_DRIVE_PERIPHERALS` durch? Analysieren Sie den Inhalt der Datei `stb_at91.c`.

2. Simulator für PIO-Bausteine

Der Simulator bietet die Möglichkeit, Vorgänge in der simulierten Hardware sichtbar zu machen d.h. zu tracen.

Überzeugen Sie sich, dass für das Modul `stb_pio.c` das Tracing eingeschaltet ist sowie die Ausgabe des Zustands der LEDs auf der Konsole stattfindet.

Erzeugen Sie eine ausführbare Datei für das Projekt und lassen Sie diese laufen.

2A)

i. Was bedeuten die Ausgaben in der Trace-Datei?

ii. In welcher Quelltextdatei werden an welchen Stellen die Ausgaben der Traces erzeugt?

iii. Welche Funktionsaufrufe (Call Stack) führen zum Trace mit dem Kürzel PIOSR?

iv. Welche Aktion der der simulierten Hardware erfolgt für diesen Trace?

Modifizieren die Beispielfunktion `example_pio1()` in der Quelltextdatei `usr_main3()`. Erzeugen Sie Schleifen oder Warteschleifen. Setzen Sie Breakpoints und stoppen das Programm zwischenzeitlich.

Untersuchen Sie das Zeitverhalten zur Laufzeit anhand der Zeitstempel in der Trace-Datei für einen Simulationslauf. Versuchen Sie nachzuvollziehen, mit welchem Quelltext die Zeit für die Zeitstempel in der Datei generiert werden. Welche Beobachtungen machen Sie?

2B)

i. Wie oft können Sie auf Ihrem Rechner eine Aktualisierung des Hardwarezustands pro Sekunde erreichen?

ii. Was passiert, wenn Sie bei einem Breakpoint im Debugger eine Zeitdauer verbringen?

iii. Welche Vorteile haben Traces in der Nachverfolgung des Programmzustands? Welche Nachteile gibt es?

2C)

- Suchen Sie den Quelltext, der die Modifikation des PIO-Statusregisters ODSR implementiert. Welcher C-Quelltext implementiert das Zusammenspiel von Enable-/Disable-/Statusregister? Vollziehen Sie die Operationen auf Bit-Ebene nach.

Ergänzen Sie nun die Beispielfunktion `pio_example2()` durch eine Warteschleife, so dass die LED-Ausgabe zyklisch ungefähr einmal pro Sekunde blinkt. Platzieren Sie `STB_DRIVE-PERIPHERALS` sowohl in der Warteschleife als auch außerhalb der Warteschleife. Justieren Sie die Zeitdauer des Blinkens jeweils auf eine Sekunde.

2D)

- Welchen Einfluss hat dies auf die Anzahl der Schleifendurchläufe? Geben Sie beide Anzahlen an. Welcher Unterschied würde sich ergeben, wenn Sie den Programmcode auf der Zielhardware des AT91M63200-Mikrocontrollers laufen lassen würden?

3. Timer-/Counter

Schalten Sie das Tracing und Ausgabe für die PIO-Bausteine aus. Machen Sie sich mit den Tracing-Möglichkeiten für das Quelltextmodul `stb_tc.c` vertraut. Verwenden Sie das Beispielprogramm `example_tc()`. Dieses zeigt die Verwendung eines Timerbausteins.
Hinweis: Das erreichen Sie durch Änderung des Quelltextes. Machen Sie zur Sicherheit einen kompletten Rebuild (für alle *.c - Übersetzungseinheiten).

3A)

i. In welchem Modus wird der Timerbaustein verwendet?

ii. Was tracen die gegebenen Tracing-Möglichkeiten?

Konfigurieren Sie alle Tracing-Möglichkeiten für die Timerbausteine in Ihr Programm. Analysieren Sie den Programmcode in der Funktion nach, die in denen der Trace `TCCNT` ausgegeben wird. Lassen Sie das Beispielprogramm laufen.

3B)

i. Wie wird der Wert des Zählregisters in der Simulation verändert?

ii. Wie wird der Wert des Zählregisters in der Hardware verändert?

iii. Warum ist die Implementierung in der Simulation anders gewählt worden? Bedenken Sie dabei die Geschwindigkeit des von Ihnen verwendeten Rechners.

iv. Beschreiben Sie die Berechnung des Zählerstands.

Das Zeitverhalten des Simulators ist in der Datei `sim_config.h` auf `ENV_REALTIME` eingestellt. Die Simulationsumgebung berechnet für jeden Aufruf des Simulatorprogrammcodes die aktuelle Zeit in Mikrosekunden (usec). Sie finden in der Datei `stb_at91.c` die Realisierung.

3C)

i. Mit Hilfe welcher „Uhr“ wird die aktuelle Zeit berechnet?

ii. Wie oft werden Tracing-Ausgaben gemacht?

iii. Wieviel Daten entstehen, wenn Sie mit den jetzigen Einstellungen über längere Zeit tracen würden?

Hinweis: Sollte Ihre Codeblocks-IDE einmal „hängen“ bleiben, dann können Sie mit „ps –ef“ die Prozess-ID (PID) heraussuchen und mit „kill -9 <pid>“ den Prozess stoppen. Beachten Sie, dass dann keine Daten mehr gespeichert werden. Wenn Sie andere Prozesse stoppen, können die merkwürdigsten Dinge passieren.