2019

Calculating Pi

LET’S LOAD OUR CPU

PHILIPP FRASE

Inhaltsverzeichnis

Abbildungsverzeichnis

Revisionshistorie

Rev. nr.	Datum	Änderung durch	Kommentar
1.0	14.04.2019	Philipp Frase	Erstausgabe

Quellenverzeichnis

- Power Point Präsentationen, Martin Burger
- www.freertos.org
- AVR-XMEGA-Microcontroller, Günter Spanner, ISBN: 978-3-89576-288-8

Aufgabenstellung

Aufgabe:

Wähle einen Algorithmus (Es darf auch ein anderer sein. Es finden sich diverse im Netz.)
Realisiere den Algorithmus in einem auf FreeRTOS basierenden Programm.
Dabei soll folgendes stets gegeben sein:

- Der aktuelle Wert soll stets gezeigt werden. Update alle 500ms
- Der Algorithmus wird mit einem Tastendruck gestartet und mit einem anderen Tastendruck gestoppt.
- Mit einer dritten Taste kann der Algorithmus zurückgesetzt werden.
- Die Kommunikation zwischen den Tasks soll mittels EventBits stattfinden.
    - EventBit zum Starten des Algorithmus
    - EventBit zum Stoppen des Algorithmus
    - EventBit zum Zurücksetzen des Algorithmus
    - EventBit für den Zustand des Kalkulationstask als Mitteilung für den Anzeige-Task
- Mindestens zwei Tasks müssen existieren.
    - Interface-Task für Buttonhandling und Display-Beschreiben
    - Kalkulations-Task für Berechnung von PI
- Erweitere das Programm mit einem Zeitmess-Hardware-Timer (wie in der Alarm-Clock Übung) und messe die Zeit, bis PI auf 5 Stellen hinter dem Komma stimmt. (Zeit auf dem Display mitlaufen lassen und bei Erreichen der Genauigkeit den Timer anhalten. Die Berechnung von PI soll weitergehen.)

Dokumentationsarbeiten:

Es soll ein kurzer Bericht zum Projekt verfasst werden. (ca. 2-3 Seiten)
Darin enthalten sein soll folgendes:

- Erklärung des gewählten Algorithmus.
- Beschreibung der Tasks. Der Ablauf der Tasks und deren Funktionsweise muss ersichtlich sein.
- Die EventBits und deren Aufgabe soll erklärt werden.
- Die Resultate der Zeitmessung sollen aufgeführt und erklärt werden.
- Wage einen Rückschluss von der gemessenen Rechenleistung auf die tatsächliche Prozessorleistung zu machen.
- Die Softwareverwaltung GIT ist anzuwenden und online in einem Repository zu sichern.
  Die Aufgabe muss bis zum 9.4. erledigt sein und abgegeben werden.
  (Link auf Repo reicht. Bericht ebenfalls im Repo)

Ausführung

Algorithmus

Für die Berechnung von Pi wird dir Leibniz-Reihe (Abb. 1) gewählt.

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}.$$

Abbildung 1 Leibniz-Reihe

Umso öfter dieser Algorithmus berechnet wird, umso näher erreicht man Pi/4.
Um anschliessend Pi auf dem Display zu sehen, muss man Pi/4 noch mit 4 multiplizieren, dies geschieht in der „printf“ Funktion.

vInterface

In diesem Task wird die ganze Display Anzeige ausgeführt. Das Display zeigt die aktuelle Version der Firmware, welche gerade auf dem Controller läuft, die Anzahl Iterationen und der momentan berechnete Wert für Pi. Das Display wird alle 500ms aktualisiert.

vButtonHandler

Der Button Handler Task sorgt dafür, dass jeder Button weiss, was er zu tun hat.
In dieser Anwendung sind die Buttons wie folgt programmiert:

- Button 1: Startet die Berechnung
- Button 2: Hält die Berechnung an
- Button 3: Setzt die Berechnung fort
- Button 4: Setzt die Berechnung zurück

vCalculation

Im Calculation Task wird der Algorithmus berechnet. Die Berechnungen finden in einer for-Schleife statt. Gesteurt wird diese Schleife mit der Laufvariable «i». Durch festlegen des Endwertes in der Durchlaufbedingung, kann man festlegen, wie oft die Berechnung durchgeführt werden soll. In diesem Fall sind die Durchläufe auf 10 Mio. mal limitiert.

Codeverwaltung mit Github

Abbildung 2 Github Logo

Um eine saubere Versionskontrolle zu haben, wurde zur Codeverwaltung die Plattform Github verwendet (Abb. 2).

Sobald ein neues Feature erfolgreich programmiert und getestet wird, „committed“ und „pusht“ man sein Projekt und hat somit seine Daten sowohl lokal wie auch auf den Github Servern.

Dies macht es sehr übersichtlich und man kann zu jeder Zeit nachvollziehen, was, wann und mit welchem Update etwas geändert wurde.

Ein weiterer Vorteil ist, wie vorhin angesprochen, dass die Daten auf den Github Servern gespeichert sind. Somit hat man im Falle eines Datenverlustes ein Backup.

Folgender Link führt zum Github Repositorie des Projekts:

- https://github.com/pfra17/Calculating_PI

Technisches Fazit

Leider wurde ich nicht ganz fertig mit dem Programm, sodass sich doch noch einige Bugs finden. Zum Beispiel das zurücksetzen auf null und erneute Starten funktioniert nicht ganz wie ich es mir vorgestellt hatte.

Um mehr Übersicht zu erhalten, entschied ich mich während des Projekts dazu einen dritten, den Button Handler Task, einzuführen. Dies führte, wie von unserem Dozenten vorhergesagt, zu sichtbaren Performance Verlusten.
Die Software schafft nun circa 324'000 Berechnungen pro Minute. Wobei es vor Einführung des dritten Tasks ungefähr 400'000 Berechnungen pro Minute waren. Das sind rund 76'000 Berechnungen weniger, was meines Erachtens schon erheblich langsamer ist.

Leider kam ich auch nicht mehr dazu den Timer zu programmieren, da ich zu beschäftigt war, die anderen Probleme in den Griff zu kriegen.

Persönliche Reflexion

Das Projekt hat mir persönlich sehr viel spass gemacht. Ich konnte meine noch nicht so starken Programmierkenntnisse wieder etwas verbessern und habe einiges über das Thema FreeRTOS gelernt.

Ich kann mir gut vorstellen, dass man solch eine Arbeit für kommende Semester vielleicht sogar als Semesterarbeit auslegen kann.