Embedded Systems

Calculating Pi

Eine Arbeit von Eric Lochner

Studiengang

HF Elektrotechnik TS TSE 2009 A

Inhalt

Aufgabenstellung

Aufgabe:

- Realisiere die Leibniz-Reihen-Berechnung in einem Task.
- Wähle einen weiteren Algorithmus aus dem Internet.
- Realisiere den Algorithmus in einem weiteren Task.
- Schreibe einen Steuertask, der die zwei erstellten Tasks kontrolliert.

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.
- Mit der vierten Taste kann der Algorithmus umgestellt werden. (zwischen Leibniz und dem zweiten Algorithmus)
- Die Kommunikation zwischen den Tasks kann entweder mit EventBits oder über TaskNotifications stattfinden.
- Folgende Event-Bits könnte man beispielsweise verwenden:
    - 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 drei Tasks müssen existieren.
    - Interface-Task für Buttonhandling und Display-Beschreiben
    - Kalkulations-Task für Berechnung von PI mit Leibniz Reihe
    - Kalkulations-Task für Berechnung von PI mit anderer Methode
- Erweitere das Programm mit einer Zeitmess-Funktion (verwende xTaskGetTickCount) und messe die Zeit, bis PI auf 5 Stellen hinter dem Komma stimmt. (Zeit auf dem Display mitlaufen lassen und beim Erreichen der Genauigkeit die Zeit berechnen. Die Berechnung von PI soll weitergehen.)

Dokumentationsarbeiten

Es soll ein kurzer Bericht zum Projekt verfasst werden. (ca. 8+ Seiten)
Darin enthalten sein soll folgendes:

- Erklärung des zusätzlich gewählten Algorithmus.
- Beschreibung der Tasks. Der Ablauf der Tasks und deren Funktionsweise muss ersichtlich sein. (z.B. Finite State Machine)
- Die EventBits/TaskNotification und deren Aufgabe/Verwendung soll erklärt werden.
- Die Resultate der Zeitmessung sollen aufgeführt und erklärt werden.
- Vergleiche die Geschwindigkeit von Leibniz gegenüber der anderen Methode
- Wage einen Rückschluss bezüglich der gemessenen Rechenleistung zur tatsächlichen Prozessorleistung zu machen.
- Die Softwareverwaltung GIT ist anzuwenden und online in einem Repository zu sichern.
  Die Aufgabe muss bis zum 25.10.2022 erledigt sein und abgegeben werden. (Link auf Repository auf Github reicht. Bericht ebenfalls im Repository)

Übung: Calculating Pi

Bei dieser Arbeit geht es darum zwei verschiedenen Algorithmen für die Berechnung der Zahl Pi im Microchip Studio - FreeRTOS zu berechnen, dies per Tastenkombination auf dem EduBoard (unser Mikrocontroller) zu steuern und an einer Anzeige auszugeben. Für das wurde uns einen Algorithmus vorgegeben diese wäre die Leibniz-Reihe, siehe Abbildung 1 Leibniz-Reihe unten. Je weiter man sie berechnet, desto näher zu Pi/4 geht die Zahl.

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

Abbildung 1 Leibniz-Reihe

Zusätzlich musste noch ein zweiter Algorithmus frei aus dem Internet gewählt werden, welcher in meinem Fall die Formel von dem indischen Mathematiker und Astronomen Kelallur Nilakantha Somayaji (Zeitraum ca. 1444-1544) wäre. Die Formel ist aufgebaut wie in der Abbildung 2 Pi-Formel von Kelallur Nilakantha Somayaji.

$$\pi = 3 + \frac{4}{3^3 - 3} - \frac{4}{5^3 - 5} + \frac{4}{7^3 - 7} - \frac{4}{9^3 - 9} + \cdots$$

Abbildung 2 Pi-Formel von Kelallur Nilakantha Somayaji

Definition von FreeRTOS

Was genau ist FreeRTOS?

FreeRTOS ist ein kostenloses Open-Source:Echtzeitbetriebssystem für embedded systems (eingebettete Systeme). Dieses System eignet sich für viele verschiedene Mikrocontroller-Architekturen und steht unter MIT-Lizenz. RTOS hat eine Bedeutung und steht für **R**eal **T**ime **O**perating **S**ystem.

Dieses Betriebssystem wurde ursprünglich von Richard Barry im Jahr 2003 entwickelt, ab 2007 übernahm Amazon die Verantwortung für das Projekt und deren Weiterentwicklung. Richard Barry ist nach wie vor noch an der mithilfe der Weiterentwicklung beteiligt.

Bei FreeRTOS handelt es sich um ein Echtzeitbetriebssystem. Dieses Betriebssystem wurde speziell für embedded System (eingebettete Systeme) optimiert, um besondere Anforderungen zu erfüllen. Verschiedene Ereignisse und Anfragen können durch externe Signale ausgelöst werden und über Hardwareschnittstellen empfangen werden.

Typische Merkmale für dieses System sind:

- Mess- oder Steueraufgaben
- Unterstützung des Interruptverhaltens der Hardwareplattform
- Unterstützung von Multitasking
- Vorhersagbare Zeitverhalten für Scheduling oder Speichermangement

Github

Für die saubere Ablage und Übersichtlichkeit der ausgeführten Arbeitsschritte im Code wurde unser Projekt auf Github abgespeichert. Dort kann man gut nachvollziehen was und wann man etwas geändert hat. Die richtige Anwendung dieser Onlineplattform kann einem sehr helfen da man von verschiedenen Standorten darauf zugreifen kann und im Falle eines Datenverlustes auf dem eigenen Laptop oder PC ein Backup auf Github hat. Git ist grundsätzlich nicht nur auf Codes beschränkt, es besteht die Möglichkeit auch andere Files/Dokumente dort hochzuladen.

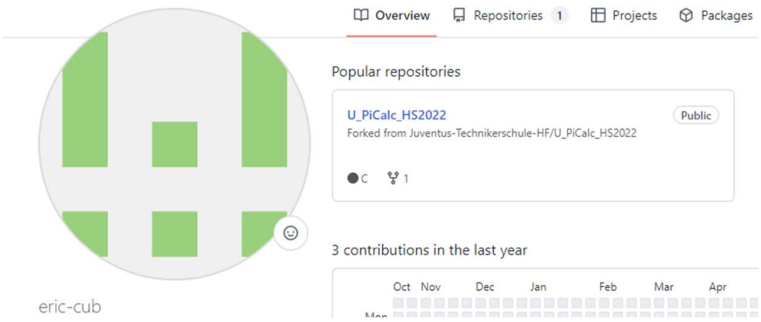

Abbildung 3 Github Profile

Die Tasks

Um das ganze Projekt umzusetzen und zum Laufen zu bringen, muss man mit verschiedenen Tasks arbeiten siehe Auflistung unten. Das Ganze habe ich versucht umzusetzen jedoch nicht ganz so wie ich mir das vorgestellt habe…

Button Handler – Controllertask

Der Button Handler ist dafür zuständig die verschiedenen Signale beim Betätigen eines Tasters zu verarbeiten und die richtigen Befehle anzusteuern. Bei diesem Projekt wurden vier Knöpfe benötigt mit den jeweiligen Aufgaben:

- 1 Button: Start der Berechnung
- 2 Buttton: Stopt die Berechnung
- 3 Button: Lässt Berechnung wieder weiterlaufen
- 4 Button: Reset der Berechnung

Der Sinn dahinter ist es für die verschiedenen Tasks einen oder mehrere Buttons zu verknüpfen welche nachher das ganze steuert.

Display – UI Task

Um die ganze Bedienung und die Zahl Pi auszugeben per Display wurde dies im UI Task abgehandelt, welcher das Display alle 500ms aktualisiert mit den aktuellen Daten. Im UI Task werden ebenfalls noch für die jeweiligen Algorithmen die Ausgabe am Display angepasst, sei es die Leibniz oder Kelullar Variante. Die Benützung der verschiedenen Knöpfe und ein Menü Bildschirm mit Titel und sonstigen Angaben sollte auch ersichtlich sein. Da bei verschiedenen Tasks die Ausführung so schnell passiert kann man diese meist gar nicht sehen am Display, um dieses Problem zu umgehen werden am Ende von dementsprechenden Tasks ein «Delay» hinzugefügt.

Calculation PI

Die beiden Algorithmen wurden jeweils in einem eigenen «Calculation» Task erstellt, wo sich eine *for* Schleife befindet, welche die ganze Berechnung immer und immer wieder durchlaufen lässt, so das Pi bei jeder Berechnung immer genauer wird. Bei den jeweiligen Algorithmen soll nachher der berechnete Wert per Variabel an den Display Task gesendet welchen diesen ausgeben sollte.

Suspend and Resume

Damit meine zwei Algorithmen nicht gegenseitig sich unterbrechen und reinfunken, wurde (versucht) ein suspend und resume Task verwendet, um dieses Problem zu verhindern. Standard mässig soll der Leibniz Algorithmus sobald Start gedrückt wird ablaufen, sobald man Button 4 betätigt soll Leibniz supended werden und Kelullar beginnen zu berechnen. Schlussendlich habe ich versucht mit den Buttonhandler dieses suspend and resume zu umgehen, welches nicht möglich ist wie sich rausgestellt hatte.

Counter – Zeitmanagement

Dieser Task würde benötigt werden um die Zeit, während dem Berechnen von Pi am Display ausgibt wie lange schon gerechnet wird. Jedoch kam ich zeitlich nicht mehr dazu dies zu

Persönliche Reflexion

Im Grundsatz macht mir das Programmieren im C eigentlich Spass und Freude, wenn mal was funktioniert, umso mehr. Da ich jedoch leider in meinem Leben bis jetzt dies nur in der Schule angetroffen und anwenden musste fällt es mir relativ schwer solche Aufgaben/Aufträge zu meistern. Sobald mal ein Error auftaucht, stellt sich meistens nur schon die Frage, wo denn genau der Fehler ist. Im Gegensatz zu anderen Mitschülern, welche dies täglich bei der Arbeit antreffen hatte ich (wie auch andere Mitschüler) schon etwas die schwierigere Ausgangslage, die Thematik selbst ist nicht das schwierige, sondern mehr das ganze Herleiten und Wissen was man alles benötigt, um den Code zum Laufen zu bringen wie auch deren Reihenfolge. Nichtsdestotrotz hatte ich ab und zu auch mal Erfolgs Momente erlebt welche mich ein paar Schritte weiter brachten.

Leider wurde ich mit der Arbeit nicht ganz fertig so dass sie auch lauft, aber ich hoffe man kann aus dem erstellten Code mit etwas Grundwissen das ganze entziffern und den Sinn dahinter entschlüsseln. Um selbst in der Lage zu sein solch einen Auftrag selbständig zu erarbeiten, benötige ich um einiges mehr an Zeit und so das ich nicht nebenbei noch andere Projekte und Aufgaben habe mit hoher Priorität.

Ich finde es etwas schade, dass man mit diesem Fach nicht schon etwas früher begonnen hat bzw. die Grundlagen schon seit den ersten paar Semester eingebaut hat anstatt Deutsch oder lern und Arbeitstechnik… Dies ist jedoch meine Meinung. Hoffentlich gibt diese Arbeit etwas Genügendes her, eine Meisterleistung ist es auf jeden Fall nicht.

Abbildungsverzeichnis