School of Engineering
InES Institute of Embedded Systems

MC1 Praktikum

Cooperating FSMs

1 Einleitung

In diesem Praktikum realisieren Sie eine grössere Anwendung mit mehreren kooperierenden Finite State Machines (FSM). Sie erhalten dazu einen Programmrahmen mit diversen vorgegebenen Modulen. Im Rahmen der Vorlesung erhalten Sie eine Einführung in die Funktionalität des realisierten Programmes sowie in die verwendeten Strukturen. Benutzen Sie für die Bearbeitung des Praktikums die Vorlesungsfolien.

2 Lernziele

- Sie können sich in einen grösseren, vorgegebenen Programmrahmen eindenken und diesen gezielt erweitern.
- Sie können kooperierende FSMs mit entsprechenden Event Queues implementieren.
- Sie können Callback Funktionen anwenden.

3 Aufgaben

Verwenden und ergänzen Sie für die folgenden Aufgaben den bereitgestellten Programmrahmen. Führen Sie das Programm aus. Dieses implementiert den Egg Timer. Im ersten Schritt werden wir das Programm so ergänzen, dass die Nachladezeit des Egg Timers durch den Benutzer gesetzt werden kann. Im zweiten Schritt folgt die Implementierung der Stoppuhr. Abschliessend wird das Modul zur Zählung der Tastendrücke realisiert.

3.1 Reload

Einbinden in mode_control

Erweitern Sie zuerst die FSM im Modul *mode_control* um den für das Einstellen des Reload-Wertes vorgegebenen State. Wir wollen erreichen, dass man über die Taste T0 zwischen dem Egg Timer und dem Reload-Mode hin und her wechseln kann. Im Reload-Mode soll die erste Zeile auf dem LCD Bildschirm angezeigt werden. Die eigentliche Reload-Funktionalität wird erst später implementiert.

Orientieren Sie sich am Code des vorgegebenen Beispielmoduls *egg_timer_ctrl*. Die Schnittstelle des Moduls ist im gegebenen File **`reload_ctrl.h`** vorgegeben

Implementieren Sie für die LCD-Ausgabe die Funktion **`rl_ctrl_update_display()`**, welche über einen Callback vom Modul *lcd* aus aufgerufen wird. Die Text Strings für die Anzeige sind in entsprechenden Macros im **`.c`** File bereits vordefiniert.

Implementieren Sie die Funktion **`rl_ctrl_put_queue()`.**

Implementieren Sie eine ganz einfache Version der Funktion **`rl_ctrl_handle_event()`** noch ohne FSM. Sie soll nur den Event für den Update des LCDs generieren. Registrieren Sie die Funktion beim Scheduler.

Bei jedem Wechsel des States im Modul *mode_control* muss die zugehörige Callback Funktionen des neuen States im Modul *lcd* registriert werden.

Testen Sie die implementierte Funktionalität.

FSM reload_ctrl

Implementieren Sie die FSM im Modul *reload_ctrl* um die Nachladezeit des Egg Timers durch den Benutzer zu setzen. Das Modul *reload_ctrl* benutzt das bereits implementierte Modul *reload*.

Die zu implementierende FSM soll über zwei Zustände verfügen: Einen für das Einstellen der Sekunden und einen für das Einstellen der Minuten. In beiden Fällen soll der Benutzer den entsprechenden Wert über die entsprechenden Events inkrementieren bzw. dekrementieren können.

Die Events und ihre Funktionen sind in **`reload_ctrl.h`** beschrieben.

Die Reload Funktion soll jetzt so aussehen wie in der Vorlesung beschrieben. Die Anzeige der zweiten Zeile auf dem LCD ist dabei vom State in *reload_ctrl* abhängig.

3.2 Stop Watch

Implementieren Sie nun die Stoppuhr. Diese zeigt Sekunden und Zehntelsekunden an. Dazu müssen Sie die Module *stop_watch_ctrl* und *stop_watch* schreiben. Die Schnittstellen sind in den zugehörigen Headerfiles beschrieben.

Beginnen Sie wiederum mit der Erweiterung der *mode_control* FSM.

Wenn das Wechseln in den Stop-Watch-Mode und die zugehörige Anzeige der ersten LCD Zeile funktionieren, folgen die Implementation der eigentlichen Stop Watch und der zugehörigen Kontroll-FSM.

3.3 Button Count

Implementieren Sie die Zähler für die Tasten T0 – T3 sowie eine zugehörige FSM um die Zähler einzeln abfragen zu können. Beides ist im Modul *button_count* vorgesehen.

Die FSM für die Abfrage hat vier States; je einen pro Taste.

Da wir die Tastendrücke auch zählen wollen, wenn die Abfrage-FSM nicht selektiert ist, wird die Zählfunktion direkt als Callback beim Modul *buttons* registriert. Rufen Sie dazu in der Funktion **`button_count_init()`** die Funktion **`buttons_register_observer()`** auf.

4 Bewertung

Das Praktikum wird mit maximal 6 Punkten bewertet:

- Aufgabe 3.1 Reload 2 Punkte
- Aufgabe 3.2 Stop watch 2 Punkte
- Aufgabe 3.3 Button Count 2 Punkte

Punkte werden nur gutgeschrieben, wenn die folgenden Bedingungen erfüllt sind:

- Der Code muss sauber, strukturiert und kommentiert sein.
- Das Programm ist softwaretechnisch sauber aufgebaut.
- Die Funktion des Programmes wird erfolgreich vorgeführt.
- Der/die Studierende muss den Code erklären und zugehörige Fragen beantworten können.
- Der Zusatzpunkt wird nur vergeben, wenn alle anderen Aufgaben gelöst sind.