School of Engineering
InES Institute of Embedded Systems

MC1
Praktikum 1

Revision Control / Debouncing

1 Einleitung

Die MC1 Praktika verwenden das bekannte CT-Board mit dem Cortex-M4 und die Keil uVision 5 Entwicklungsumgebung. Sie erhalten, wie aus CT gewohnt, für jedes Praktikum jeweils einen Programmrahmen (Keil uVision 5 Projekt). Diesen erweitern Sie mit Ihrer Lösung.

Beachten Sie beim Schreiben Ihres Codes die Coding Guidelines:
OLAT MC1: Unterlagen/UnterlagenAllgemein/CodingGuidelines.pdf

- Falls Sie auf Ihrem eigenen Computer entwickeln:
  Installieren Sie das neueste CT-Board Pack. Siehe https://ennis.zhaw.ch

2 Lernziele

- Sie können für einfache Laborprojekte SVN als Revision Control System einsetzen.
- Sie können für einen Schalter eine Flankendetektion mit Debouncing in Firmware implementieren.
- Sie können das Prellen eines Schalters mit dem Oszilloskop messen.
- Sie verstehen, dass der Einsatz von `'static'` Variablen die mehrfache Verwendung einer Funktion einschränken kann und können dies umgehen.

3 Revision Control mit Subversion (SVN)

3.1 Allgemein

Die Verwaltung Ihres Source Codes für alle MC1 Praktika soll mit Subversion (SVN) erfolgen. Für die Integration in Windows eignet sich TortoiseSVN (http://tortoisesvn.net/index.de.html). Beides sind Open Source Tools.

- Falls Sie auf Ihrem eigenen Computer entwickeln:
  Installieren Sie TortoiseSVN. Sie finden die Installationsdateien auf der TortoiseSVN homepage.
- Falls Sie die Arbeit mit SVN kennen, können Sie selbstverständlich auch einen anderen SVN Client verwenden.

3.2 Getting started

Lesen Sie als Vorbereitung für das Praktikum die Kapitel "Getting Started" und "Basic Version-Control Concepts" im Benutzerhandbuch von TortoiseSVN (online oder pdf auf OLAT).

Welche Vorteile bietet der Einsatz eines Revision Control Systems wie SVN? Zählen Sie 3 Vorteile auf.

- Versionskontrolle
-
-

Wie geht SVN damit um, dass mehrere Benutzer gleichzeitig das gleiche File bearbeiten möchten?

SVN benutzt den Lock-Modify-Unlock-Mechanismus

Was machen die folgenden SVN Befehle?

SVN Checkout

Arbeitskopie erstellen

SVN Update

Änderungen herunterladen

SVN Add

Dateien zum Repository hinzufügen

SVN Commit

Änderungen an Dateien hochladen

SVN Revert

Frühere Versionen wiederherstellen

3.3 Repository einrichten

Richten Sie mit dem Onlinetool auf dem SVN Server der ZHAW (https://rom.zhaw.ch/) ein eigenes Repository ein. Siehe Abbildung 1. Verwenden Sie den Namen **`MC1_<Ihr_ZHAW_Kurzzeichen>`**. Verleihen Sie Ihrem betreuenden Dozenten Schreib- und Leserechte.

Abbildung 1: Online Tool unter rom.zhaw.ch

Legen Sie mit dem Repo-browser (*TortoiseSVN → Repo-browser*) die für jedes Repository üblichen 3 Verzeichnisse an: *'branches'*, *'tags'* und *'trunk'*. Siehe Abbildung 2.

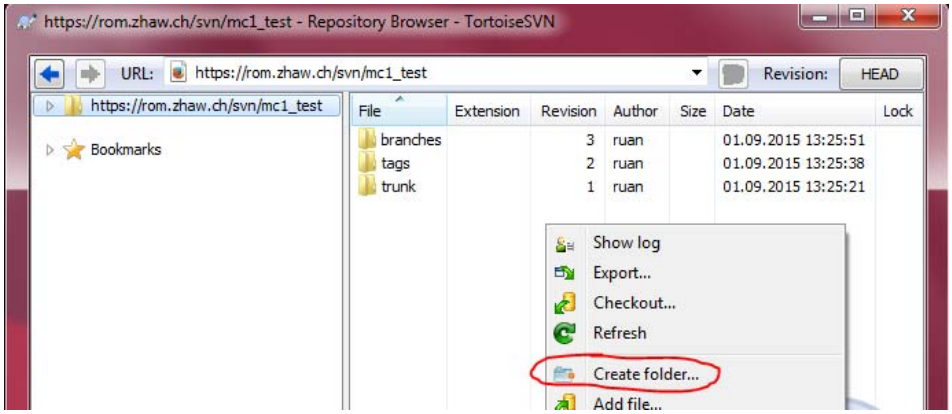


Abbildung 2: Anlegen von Verzeichnissen im TortoiseSVN Repo-Browser

trunk enthält den Hauptentwicklungspfad. Hier ist jeweils die aktuelle Version zu finden. *branch* enthält Nebenpfade. Dies kann beispielsweise ein Bugfix für eine frühere stabile Version sein. Oder es können hier grössere Umbauten am Code stattfinden, ohne die Benutzer auf dem *trunk* zu stören. *tags* enthält den Stand von *trunk* oder *branches* zu einem ganz bestimmten Zeitpunkt, zum Beispiel den Stand des ersten Releases. Unter *tags* finden in der Regel keine commits statt. Wir werden in MC1 vor allem den *trunk* verwenden.

3.4 Praktikum importieren

Erstellen Sie mit *SVN Checkout* eine Working Copy in Ihrem lokalen Verzeichnis. Siehe Abbildung 3.

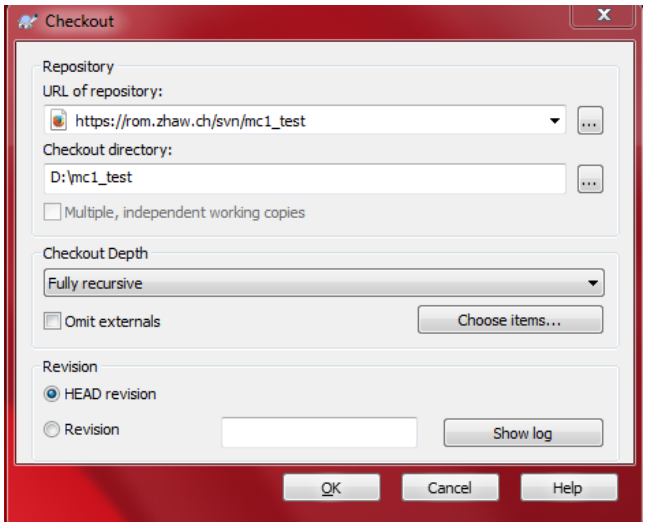


Abbildung 3: Erstellen einer Working Copy

Legen Sie in Ihrer Working Copy das Verzeichnis **`\trunk\labs\lab1`** an. Entpacken Sie das Zip File mit dem Coderahmen in dieses Verzeichnis. Gehen Sie im Windows Explorer auf das Verzeichnis labs und fügen Sie dieses inklusive Unterverzeichnissen mit *TortoiseSVN → Add* hinzu. Anschliessend mit SVN Commit ins Repository hochladen.

Keil uVision produziert beim Compilieren viele Binär- und Hilfsdateien. Diese können jederzeit aus den Sourcen wieder erzeugt werden und werden daher nicht unter Revision Control verwaltet. Mit *"Unversion and add to ignore list"* kann man SVN mitteilen, dass eine Datei ignoriert werden soll. Selektieren Sie alle Dateien im Verzeichnis "*build*" und führen Sie diesen Befehl aus. Im Projektverzeichnis können alle Dateien ausser die Unterverzeichnisse sowie **`*.uvoptx`** und **`*.uvprojx`** SVN mässig ignoriert werden.

Somit ist jetzt alles bereit, um mit dem Codieren zu beginnen. Nutzen Sie Subversion und führen Sie häufig einen *TortoiseSVN → commit* durch um einzelne Versionen im Repository zu sichern.

4 Entprellen von Schaltern

4.1 Zusatzboard mit Schiebeschaltern

Verwenden Sie für dieses Praktikum das Zusatzboard mit den 4 Schiebeschaltern. Siehe Abbildung 4. Dieses wird an Port P6 (**`GPIOB[11:0]`**) angeschlossen. Die Schalter liegen an den Bits [3:0]. Sie finden das Schema im Anhang.

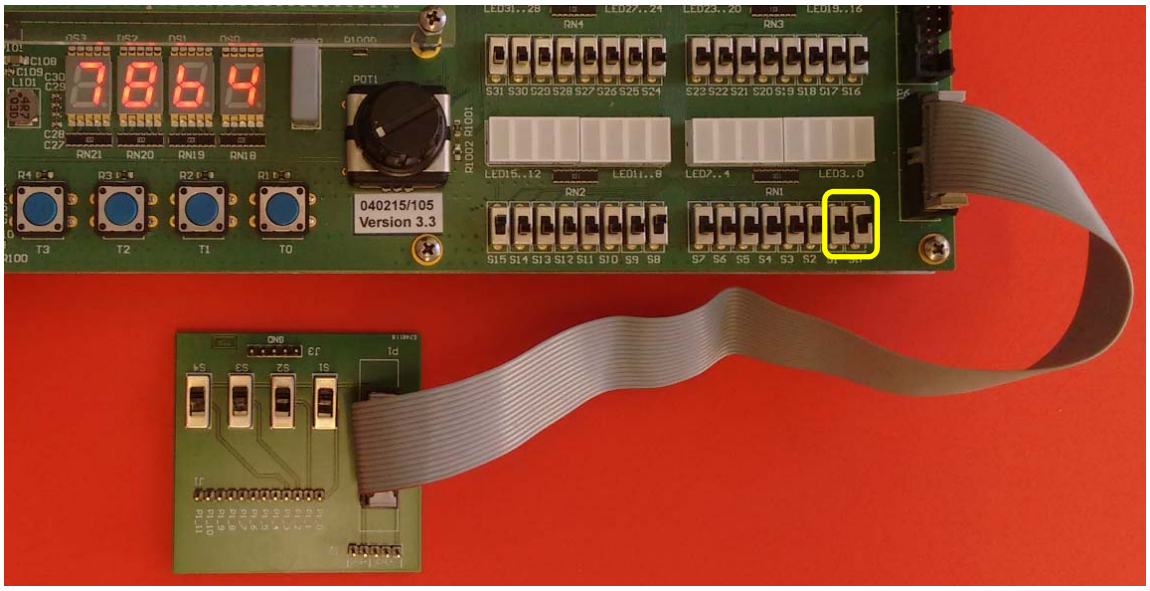


Abbildung 4: Zusatzboard mit Schiebeschaltern

4.2 Flankendetektion ohne Entprellen

Diese Funktionalität wird verwendet, wenn sich die CT-Board Schalter S1/S0 beide in der Stellung **unten** befinden.

Beispielprogramm

Im vorgegebenen Programmrahmen ist eine Flankendetektion auf den vier Schiebeschaltern implementiert. Bei jeder steigenden oder fallenden Flanke eines Schalters wird der zugehörige Zähler hochgezählt und an die zugehörige 7-segment Anzeige ausgegeben.

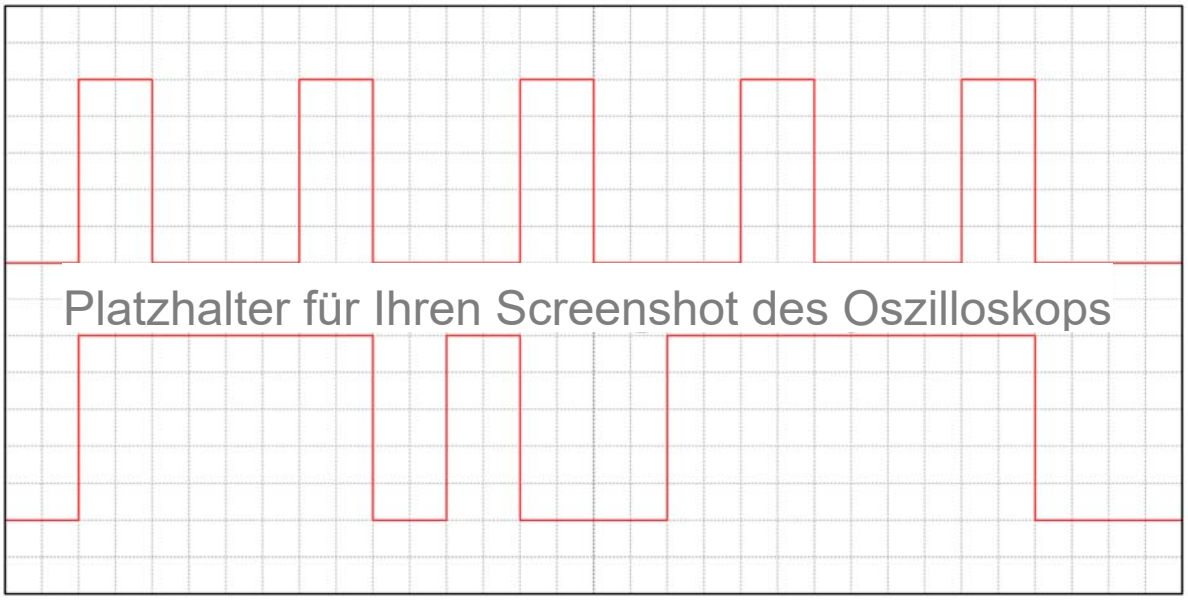
Die Beschreibung der Funktion **`detect_switch_change()`** und ihres Rückgabewertes finden Sie im Code. In dieser Funktion wird eine lokale, statische Variable verwendet um die Werte der vier Schalter von einem Funktionsaufruf zum nächsten zu speichern.

Welche Variable ist das?

Messen

Testen Sie die Funktion auf dem CT-Board. Sie werden feststellen, dass die Schalter prellen. Messen Sie das Prellen eines Schalters mit dem Oszilloskop. Sie können für den Anschluss der Sonde den entsprechenden Pin auf dem Stecker J1 des Zusatzboardes verwenden.

Zusätzlich möchten wir auf dem Oszilloskop sehen, in welchen Zeitabständen die Funktion **`detect_switch_change()`** aufgerufen wird, d.h. wie häufig wir abtasten. PB6 wird gleich zu Beginn der Funktion **`detect_switch_change()`**getoggelt. Erstellen Sie unten ein Foto oder einen Screenshot, der sowohl das Prellen des Schalters als auch das Signal PB6 (P1_6) zeigt.



4.3 Flankendetektion mit Entprellen

Diese Funktionalität wird verwendet, wenn sich die CT-Board Schalter S1/S0 in den Stellungen **unten/oben** befinden.

Implementieren Sie eine Flankendetektion mit Entprellen in der Funktion **`detect_switch_change_debounce()`**. Dies erreichen Sie, indem Sie ein Sliding Window Verfahren (wie in der Vorlesung gezeigt) implementieren. Definieren Sie dazu in der Funktion einen statischen Array, z.B.

```
static uint8_t switch_samples[NR_SAMPLES];
```

Dieser Array dient der Speicherung der zuletzt von PB[3:0] gelesenen Werte. Die Anzahl gespeicherter Werte wird durch das Präprozessormakro **`NR_SAMPLES`** definiert. Bei jedem Aufruf der Funktion werden die gespeicherten Werte um eine Stelle im Array geschoben. Am einen Ende des Arrays wird der älteste Wert weggeworfen und auf der anderen Seite des Arrays durch den von GPIOB gelesenen Wert ersetzt.

Die Funktion sucht für alle vier Pins nach

- Steigenden Flanken
  Lauter Nullen gefolgt von einer Eins (neuestes Bit)
- Fallenden Flanken
  Lauter Einsen gefolgt von einer Null (neuestes Bit)

Für die parallele Suche nach diesen Flanken eignen sich die bitweise **`AND`** Operation (**`&`**) und die bitweise Negation (**`~`**).

Wählen Sie die Grösse des Arrays so, dass kein Prellen mehr auftritt.

4.4 Flankendetektion und Entprellen für mehrere Boards

Diese Funktionalität wird verwendet, wenn sich der CT-Board Schalter S1 in der Stellung **oben** befindet.

Die Funktionen in den vorherigen Abschnitten weisen Nachteile auf:

1. Die Funktionen sind spezifisch für den Port B geschrieben und können nicht auf mehreren Ports gleichzeitig eingesetzt werden.
2. Für das Zufügen eines Samples, müssen alle anderen Samples umkopiert werden (Shift des Buffers). Dies erfordert Rechenzeit.

Nachteil 1 – Eingeschränkter Reuse

Die Wiederverwendung (Reuse) wird durch die **`static`** Variablen verhindert. Will man ein zweites Zusatzboard verwenden, z.B. an Port P5 (**`GPIOA[11:0]`**) muss eine zweite Funktion erstellt werden. Würde man zweimal die gleiche Funktion verwenden, dann würden beide Boards für ihre früher gelesenen Schalterstellungen den gleichen Speicher verwenden.

Dieser Nachteil kann behoben werden, wenn der Speicher aus der Funktion herausgelöst wird. Die aufrufende Funktion kann den Speicher bei sich allozieren und einen Pointer darauf übergeben. Dadurch entsteht eine generische Funktion, welche für beliebig viele Zusatzboards parallel verwendet werden kann.

Nachteil 2 – Umkopieren von Samples

Diesen Nachteil kann man durch einen Ringbuffer beheben. Man merkt sich, wo im Array das älteste Element durch das neue Sample ersetzt werden soll. Ein entsprechender **`struct`** ist bereits im Programmrahmen als Typ vordefiniert.

Aufgabe

Schauen Sie sich das Interface der Funktion **`generic_debounce()`** und die zugehörigen Kommentare im Code an. Wie funktioniert die Funktion? Welche Parameter benötigt sie?

Implementieren Sie die Funktion mit dem angedachten Ringbuffer. Allozieren Sie Speicherplatz für den Ringbuffer im Hauptprogramm. Rufen Sie die Funktion an der markierten Stelle im Hauptprogramm auf (Default case im switch statement). Vor dem Aufruf muss das Hauptprogramm den gewünschten GPIO Port einlesen und als Parameter an die Funktion übergeben.

Testen Sie die Funktion an Port B. Eventuell müssen Sie die Anzahl Samples für das Debouncing erhöhen um stabile Ergebnisse zu erhalten.

5 Bewertung

Das Praktikum wird mit maximal 4 Punkten bewertet:

- 3 Einführung in SVN inklusive Anlegen des Repositories und Beantworten der Fragen — 1 Punkt
- 4.2 Flankendetektion ohne Entprellen — 1 Punkt
- 4.3 Flankendetektion mit Entprellen — 1 Punkt
- 4.4 Flankendetektion und Entprellen für mehrere Boards — 1 Punkt

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.

6 Anhang: Schema Zusatzboard