CT Praktikum

Interrupt

1 Einleitung

In diesem Praktikum wenden Sie Interrupts an. Sie programmieren und nutzen den User-Button des ST Discovery Boards (blauer Knopf) als manuelle Interrupt-Quelle.

2 Lernziele

- Sie können eine Interrupt Service Routine (Interrupt Handler) implementieren.
- Sie können den entsprechenden Hardware Interrupt initialisieren.
- Sie verstehen, wie mit Hilfe von Interrupts auf externe Ereignisse reagiert werden kann.

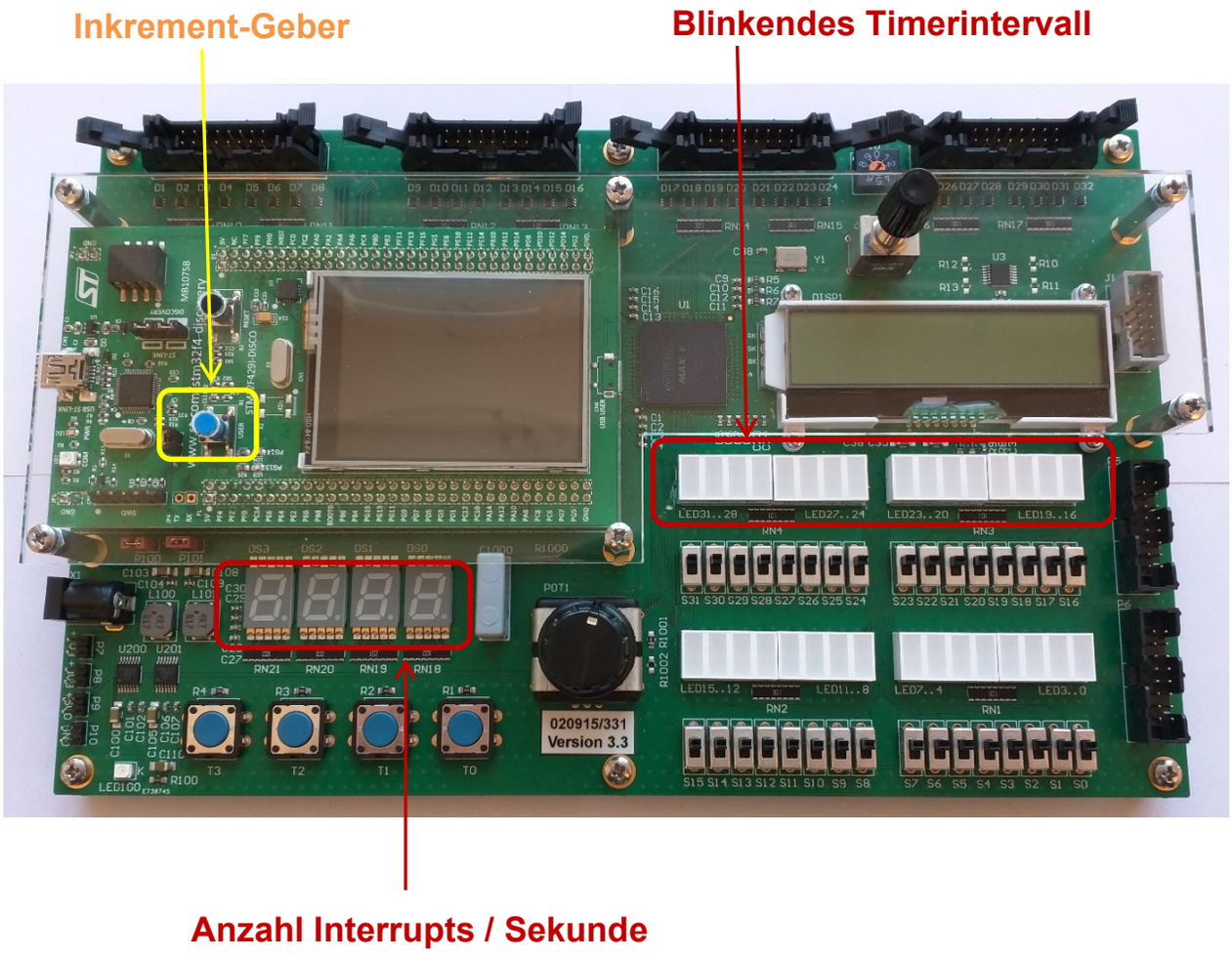


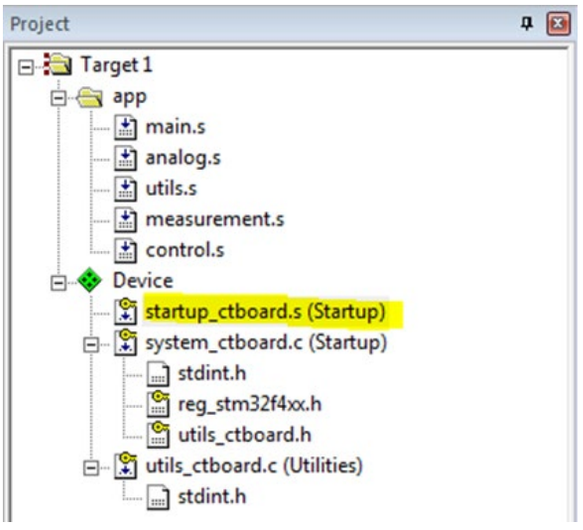
Abbildung 1: Übersicht

3 Aufgaben

3.1 Suchen der Vektortabelle

Das Signal des Inkrementgebers (User Button) dient als Interrupt Quelle. Für die Implementation müssen Sie ausschliesslich das File `main.s` editieren.

a) Öffnen Sie den vorgegebenen Programmrahmen in uVision. Suchen Sie die Vektortabelle im File `startup_ctboard.s`. Diese bildet die für den STM32F429I Chip spezifische Verdrahtung der Interruptquellen ab.



b) Das vorgegebene Programm konfiguriert den Eingang PA.0 (*Signal blauer User-Button*) als Quelle für den *EXTI Line 0* Interrupt.

Wie heisst das Label für den zugehörigen Interrupt Handler? Sie finden die Antwort in der Vektortabelle in `startup_ctboard.s`

EXTI0_IRQHandler

Welcher IRQ Nummer (IRQn) entspricht dies? Bestimmen Sie n durch abzählen in der Vektortabelle.

22te Stelle im Array → IRQ22

22 · 4 = 88 → 0x58

Im Reference Manual zur F4 Familie (uVision, Books Tab) auf Seite 373ff befindet sich eine Tabelle mit denselben Informationen (Kanal Nummer → Position) wie im File `startup_ctboard.s`.

c) Schalten Sie die Interrupt Quelle *EXTI Line 0* im Register `SETENAx` des Interrupt Controllers (NVIC) frei.

Freischalten Interrupt Quelle

```
; -------------------------------------------------------------
; -- Main
; -------------------------------------------------------------
main            PROC
                ...
                ; Configure NVIC (enable interrupt channel)
                ; STUDENTS: To be programmed
```

Verwenden Sie die Slides zum Thema „Exceptional Control Flow“ und die Seite „Nested Vector Interrupt Controller“ auf dem CT-Wiki

d) Implementieren Sie nun einen rudimentären Interrupt Handler. Verwenden Sie dafür das oben bestimmte Label EXTI0_IRQHandler. Setzen Sie innerhalb der Handler Routine den Interrupt Request IRQ_EXTI0 zurück. Rufen Sie dafür die vorgegebene Subroutine auf: `BL clear_IRQ_EXTI0.`

Interrupt Handler für EXTI0

```
; -------------------------------------------------------------
; Handler for EXTI0 interrupt
; -------------------------------------------------------------
                ; STUDENTS: To be programmed
```

Da die Vektortabelle nicht in `main.s` liegt, müssen Sie das Label mit der Assembler Direktive ‘EXPORT’ ausserhalb von `main.s` bekannt machen.

```
<label_example> PROC
                EXPORT <label_example>
                ...
                ENDP
```

e) Setzen Sie einen Breakpoint am Anfang der Handler Routine und prüfen Sie mit dem Debugger, dass die Handler Routine aufgerufen wird.

f) Erweitern Sie Ihre Handler Routine. Erstellen Sie einen Zähler, der die Anzahl Interrupts aufsummiert. Speichern Sie den Zähler in einer von Ihnen zu definierenden Variable, in der gegebenen ‘Data Section’. Das Zurücksetzen des Interrupt Requests verbleibt am Schluss der Handler Routine.

Data Bereich für Variablen

```
; -------------------------------------------------------------
; -- Variables
; -------------------------------------------------------------

                AREA myVars, DATA, READWRITE

                ; STUDENTS: To be programmed
```

g) Erweitern Sie die Endlosschlaufe im main und zeigen Sie die durch den Handler gespeicherte Zählvariable, auf der 7-Segment Anzeige an.

> Sie können den Binärwert der Zählvariable direkt über das Binär-Interface der 7-Segmentanzeige ausgeben (siehe CT-Wiki: „7 Segment Binary Interface“).

h) Testen Sie Ihre Lösung. Ist der Wert Ihrer Zählvariable korrekt?

3.2 Messen der Interrupts in einem Zeitintervall

Nun soll die Anzahl der Interrupts ermittelt werden, welche innerhalb des voreingestellten Timer-Intervalls (2 Sekunden) ausgelöst wurden. Diese werden dann auf der 7-Segment-Anzeige ausgegeben.

Dazu erhöht der bestehende EXTI0 Interrupt Handler weiterhin bei jedem Aufruf Ihre Zählvariable. Der Wert der Zählvariablen wird alle zwei Sekunden durch die Timer Interrupt Service Routine in eine andere Variable kopiert und danach wieder auf Null zurückgesetzt. Dadurch erhält man die Druckzahl des Knopfes pro zwei Sekunden. Das Hauptprogramm gibt in seiner Endlos Schleife den kopierten Wert auf die 7-Segment-Anzeige.

Dies geschieht in unserem Fall mit dem Timer TIM2, der alle zwei Sekunden einen Interrupt auslöst.

a) Bestimmen Sie die Interrupt-Nummer des Timer Interrupts.

Wie heisst das Label für den zugehörigen Interrupt Handler? Sie finden die Antwort in der Vektortabelle in `startup_ctboard.s`

TIM2_IRQHandler

Welcher IRQ Nummer (IRQn) entspricht dies? Bestimmen Sie n durch abzählen in der Vektortabelle.

44te Stelle im Array → IRQ44

44·4 = 176 → 0x80

b) Schalten Sie den Timer 2 als Interruptquelle im Register `SETENAx` frei. Passen Sie einfach die entsprechende Zeile aus Aufgabe 3.1 c) an.

c) Implementieren Sie einen einfachen Interrupt Handler für den Timer 2. Lassen Sie sich den Aufruf des Interrupts auf LED31..16 durch ein Blinken im Timer Takt (alle LEDs ein bzw. ausschalten) anzeigen. Verwenden Sie für das Rücksetzen des Timer Interrupt Requests die vorgegebene Subroutine `clear_IRQ_TIM2`.

Interrupt Handler für TIM2 in „main.s“:

```
; -------------------------------------------------------------
; Handler for TIM2 interrupt
; -------------------------------------------------------------
                ; STUDENTS: To be programmed
```

d) Erstellen Sie in der ‘Data Section’ eine zusätzliche Puffervariable für Ihren aktuellen Zähler. Ergänzen Sie den Timer 2 Interrupt Handler. Kopieren Sie den Wert der Zählvariable in Ihre Puffervariable und setzen Sie die Zählvariable auf ‘0’ zurück.

e) Passen Sie im „Main Loop“ die Ausgabe auf die 7-Segmentanzeige an. Diese soll nun den gepufferten Wert ausgeben.

> Die Ausgabe des gepufferten Wertes geschieht im „Main Loop“. Im Interrupt Handler TIM2 wird lediglich die Zählvariable gepuffert und zurückgesetzt.

3.3 Bewertung

Kapitel / Aufgabe	Bewertungskriterien	Gewichtung
4.1	Das Programm erfüllt die in Aufgabe 4.1 geforderte Funktionalität.	1/4
4.2	Das Programm erfüllt die in Aufgabe 4.2 geforderte Funktionalität.	3/4