AUFGABE 3: BLINKY ASM

AUFGABENSTELLUNG:

Erstellen Sie ein ARM Assembler Programm, dass die RGB LED des Boosterpacks wie in Abbildung 1 gezeigt in abwechselnden Farben blinken lässt. Die Blinkreihenfolge soll nach drücken des Tasters SW1 kontinuierlich in einer Endlosschleife blinken.

Optional (Für volle Punktzahl notwendig):
Nachdem Sie die Aufgabe gelöst haben, schreiben Sie Ihre Matrikelnummern in die Register R5, bzw. R7 und lassen Sie anschließend Die Register R4, bzw. R6 hochzählen, bis der Wert der jeweiligen Matrikelnummer erreicht ist. Im Anschluss sollen die Registerwerte von R4, bzw. R6 wieder bis Null heruntergezählt werden.

Abbildung 1: Blinkreihenfolge

VORGEHENSWEISE:

1. Öffnen Sie die Anwendung **KEIL µVISION 5** und öffnen über *Project und Open Project* die Projektdatei **Blinky.uvprojx**

2. Bearbeiten Sie die Dateien *„Blinky.s"* und ggfs. *„Startup.s"*

Tipp:
Schauen Sie sich zur Erstellung des ASM Codes Ihre VL Unterlagen an und studieren Sie den Anhang dieses Dokuments.

Auch im Dokument *„Einführung_Assembler-und-Programmablaufplan"* finden Sie wertvolle Informationen.

ABGABE:

1. Präsentieren Sie Ihr Programm im Labor einem Tutor

2. Markieren Sie den Quellcode mit Ihrem vollen Namen und Ihrer Matrikelnummer

3. Kommentieren Sie Ihre Codezeilen **sinvoll** und fertigen Sie ein Programmablaufplan (PAP) an (siehe Dokument Einführung_Assembler-und-Programmablaufplan.pdf).

Speichern Sie das PAP als JPEG oder PDF. (Freie Tool-Wahl zum Erstellen des PAP, ggfs. auch händisch zeichnen und danach fotografieren/ scannen)

Laden Sie Ihren Code und das PAP bei Stud.IP hoch **(Bitte auf korrekte Urheberrechtseinstellungen im Stud.IP achten!)**

ZUSATZ Matrikelnummer-Aufgabe:

Zeigen Sie lauffähiges Programm im Debug-Modus, wie der Zähler die benannten Register hoch-, bzw. runterzählt und die Matrikelnummern in den Registern korrekt hinterlegt sind.

Fügen Sie zu diesem Zweck vor und nach den Zähleraufrufen einen Breakpoint ein.

Auch dieser Codeabschnitt ist für die Abgabe **sinvoll** zu kommentieren.

Anhang A: Tipps und Hinweise

Taster:

Der Taster ist angeschlossen über GPIO-PORT PF4. Diesmal handelt es sich allerdings um eine digitale Eingabe. Dies bedeutet, das Direction-Register muss für Pin 4 auf 0 gesetzt werden, um den Pin als Eingang-Pin zu nutzen. Im Digital-Enable-Register muss das passende Bitmuster für den Taster gesetzt werden. Der Eingang-Pin befindet sich in einem undefinierten Zustand, daher muss ein Pull-Up-Widerstand eingesetzt werden um die Leitung auf einen definierten Wert von 1 (HIGH) zu setzen.

Dies geschieht indem das Bitmuster 0x10 für den Port PF4 im Pull-Up-Register gesetzt wird. Nun geht es ans Auslesen: da es sich um einen Eingang-Pin handelt, wird nicht ins Data-Register geschrieben. Das Bit für den Port PF4 ist ‚1' wenn der Taster nicht gedrückt wird. Sobald der Taster gedrückt wird, steht im Data-Register an Bit 4 (0x10) eine ‚0' (LOW).

Dies sollte mit einer „Vergleichsfunktion (if –Anweisung in Assembler) " vor dem Toggeln der LEDs in Assembler realisiert werden.

Ansteuerung Ein-/Ausgabe-Ports:

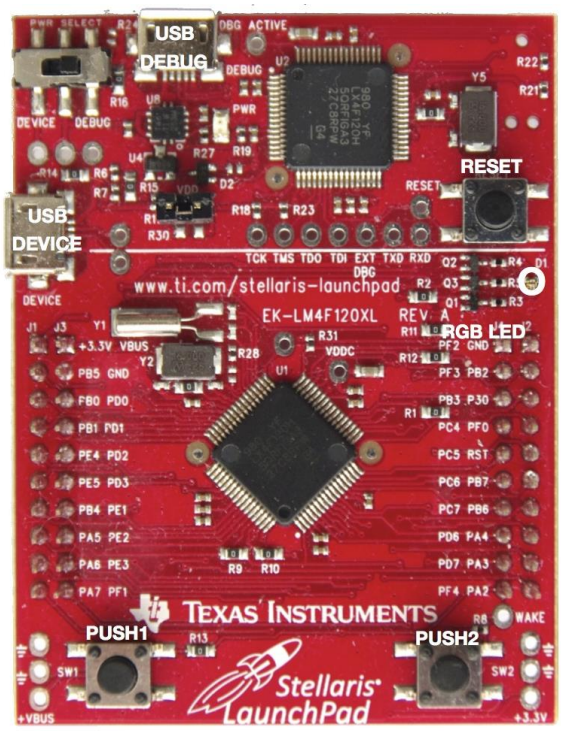

Abbildung 2: TI-Launchpad

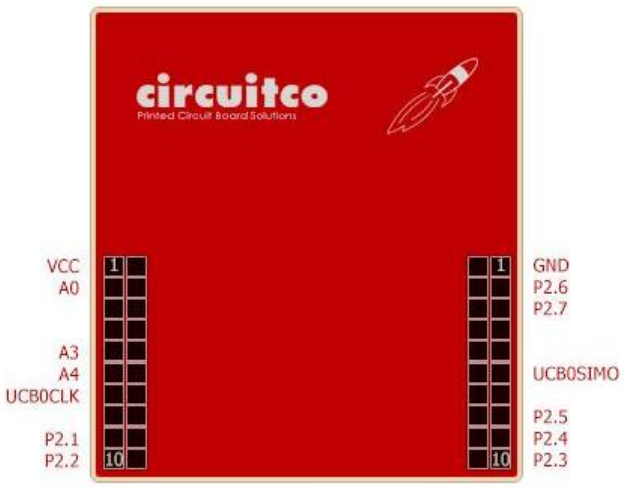

Abbildung 3: Boosterpack (Rückseite)

Register	**Adresse**	**Beschreibung**
GPIO_PORTF_DATA_R	0x400253FC	Data-Register von PORT F
GPIO_PORTF_DIR_R	0x40025400	Direction-Register von PORT F
GPIO_PORTF_PUR_R	0x40025510	Pull-Up-Register von PORT F
GPIO_PORTF_DEN_R	0x4002551C	Digital-Enable-Register von PORT F
SYSCTL_RCGC2_GPIOF	0x00000020	Bitmuster für Clock-Aktivierung

Boosterpack-Pin	**Stellaris-Launchpad-Pin**	**Funktion (Bitmuster)**
P2.1	PA6	RGB-Led Rot (0x40)
P2.2	PA7	RGB-Led Grün (0x80)
P2.4	PA3	RGB-Led Blau (0x8)
	PF4	PUSH1-Button (0x10)

Um im RGB Farbkreis sowohl die Primärfarben (**R**ot, **G**rün, **B**lau) als auch die Sekundärfarben (Magenta, Gelb, Cyan) zu erhalten, müssen jeweils zwei Primärfarben gemischt werden.
(Beispiel: Grün (0x80) + Rot (0x40) = Gelb (0xC0))

Kombination	**Bitmuster**	**Farbe**
RGB	0xC8	Weiß
RG	0xC0	Gelb
R	0x40	Rot
RB	0x48	Magenta
G	0x80	Grün
GB	0x88	Cyan
B	0x08	Blau