Embedded Systems
Toolchain
Praktikum 3

Fachhochschule Bielefeld
Campus Minden
Studiengang Informatik

Beteiligte Personen:

Name	Matrikelnummer
Jan-Hendrik Sünderkamp	1153536
Peter Dick	1050185

---

20. Mai 2018

Inhaltsverzeichnis

1 Aufgabe 1

1.1 Aufgabenstellung

- Installieren und konfigurieren Sie die Entwicklungsumgebung Energia für das Tiva C Series LaunchPad gemäß der beigelegten Anleitung oder Anweisungen auf der Energia-Webseite.
- Schließen Sie das Launchpad an und wählen Sie in der Entwicklungsumgebung die entsprechende Plattform (Tools -> Board -> "Launch Pad (Tiva C) w/ TM4C123 (80MHz)") aus. Ggf. müssen Sie das Board über den Boardverwalter installieren (Tools -> Board -> Boards Manager).
- Wählen Sie aus dem Menü der Entwicklungsumgebung das Programm "Blink" (findet sich unter File -> Examples -> 1. Basics) aus.
- Ändern Sie in dem Programm das Zeitintervall von 1s auf 4s. Vermeiden Sie "magic numbers".
- Schalten Sie unter "File -> Preferences" den "verbose output" für "compilation" und "upload" ein.
- Prüfen Sie Ihr Programm mittels der Entwicklungsumgebung ("Verify code after upload").
- Laden Sie das geänderte Programm auf das Board hoch (Pfeilsymbol "Upload").

1.2 Durchführung

Zunächst wurde die Entwicklungsumgebung *Energia* für das *Tiva C Series LaunchPad* nach der gegebenen Anleitung, auf einem Windows 10 PC, installiert.

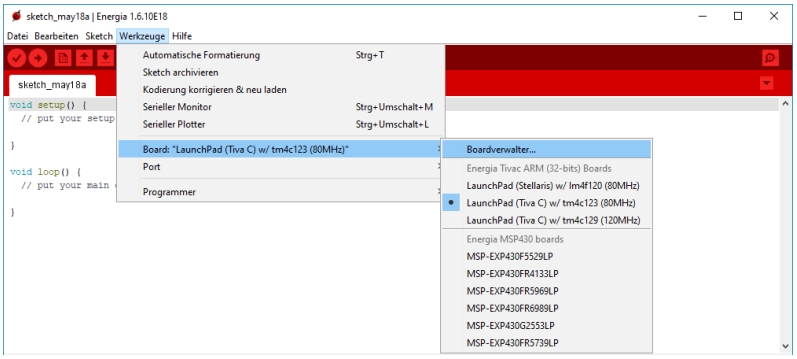


Abbildung 1: Öffnen des Board Manager

Wie oben in Abbildung 1 gezeigt wurde der Board Manager gestartet und durch ihn dann das *TM4C123* installiert.

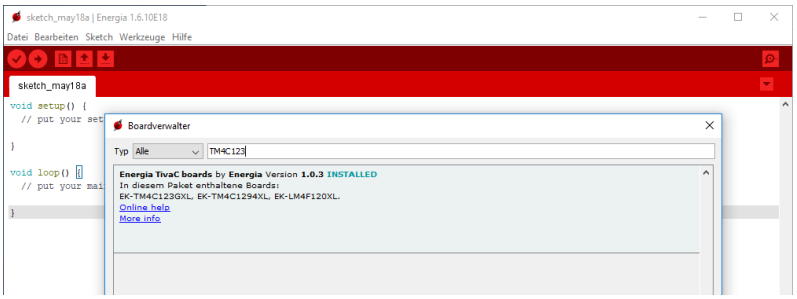


Abbildung 2: Suchen des TM4C123

Wie in Abbildung 1 zu sehen ist das *LaunchPad (Tiva C) w/ tm4c123 (80MHz)* bereits ausgewählt.

Als nächstes wurde das Beispiel *Blink* überarbeitet und das Zeitintervall des Blink-Programms von 1s auf 4s gesetzt. Dazu wurde die Variable $BLINK\_INTERVALL$ erstellt um *magic numbers* zu vermeiden. $BLINK\_INTERVALL$ wurde mit dem Wert 4000 definiert und an den den Befehl *delay* übergeben.

```
/*
  Blink
  The basic Energia example.
  Turns on an LED on for one second, then off for one second, repeatedly.
  Change the LED define to blink other LEDs.

  Hardware Required:
  * LaunchPad with an LED

  This example code is in the public domain.
*/

// most launchpads have a red LED
#define LED RED_LED
#define BLINK_INTERVALL 4000

//see pins_energia.h for more LED definitions
//#define LED GREEN_LED

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(LED, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(LED, HIGH);    // turn the LED on (HIGH is the voltage level)
```

```
  delay(BLINK_INTERVALL);      // wait for 4 seconds
  digitalWrite(LED, LOW);      // turn the LED off by making the voltage LOW
  delay(BLINK_INTERVALL);      // wait for 4 seconds
}
```

Zuletzt wurden die Einstellungen anhand der Aufgabenstellung, wie in Abbildung 3, angepasst, das Programm mit der Entwicklungsumgebung geprüft und auf das Board geladen.

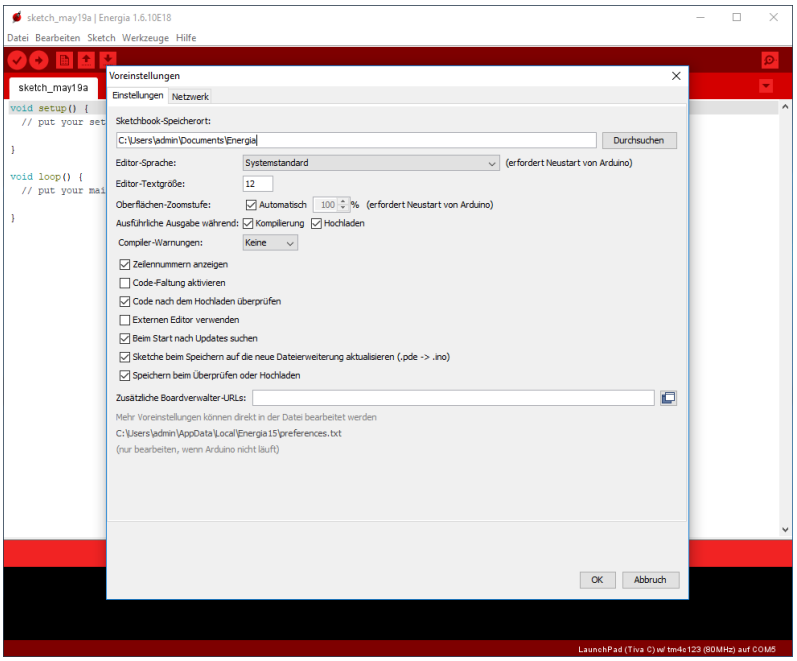


Abbildung 3: Einstellungen

2 Aufgabe 2

Welche generellen Schritte führt die hinter der Energia-Entwicklungsumgebung liegende Toolchain aus (Tipp: Betrachten Sie dafür den "verbose output")?

Zuerst werden die .cpp und .c zu .cpp.o und c.o Dateien compiliert. Dann werden die einzelnen .cpp.o und .c.o zu einer core.a Datei compiliert. Danach werden die Libs zusammen gelingt. Und auf das Board geflasht.

Aus welchen Tools besteht die Toolchain (der letzte Schritt ist das Hochladen auf das Board)?

Die Toolchain besteht aus folgenden Tools:

- arm-none-eabi-g++ : Compiliert .cpp Dateien.
- arm-none-eabi-gcc : Compiliert .c Dateien.
- arm-none-eabi-ar : Compiliert .cpp.o und .c.o Dateien zu einer core.a Datei
- arm-none-eabi-objcopy : Generiert die Binär-Datei.
- dslite : Flasht die Binär-Datei auf das Board.

3 Aufgabe 3

Lokalisieren Sie in den Verzeichnissen der Energia-Installation das Linker-Skript "lm4fcpp_blizzard" (Tipp: Schauen Sie Sich den Linker-Schritt in der Konsole an).

C:\Users\admin\AppData\Local\Energia15\packages\energia\hardware\tivac\1.0.3\variants\EK-TM4C123GXL\lm4fcpp_blizzard.ld

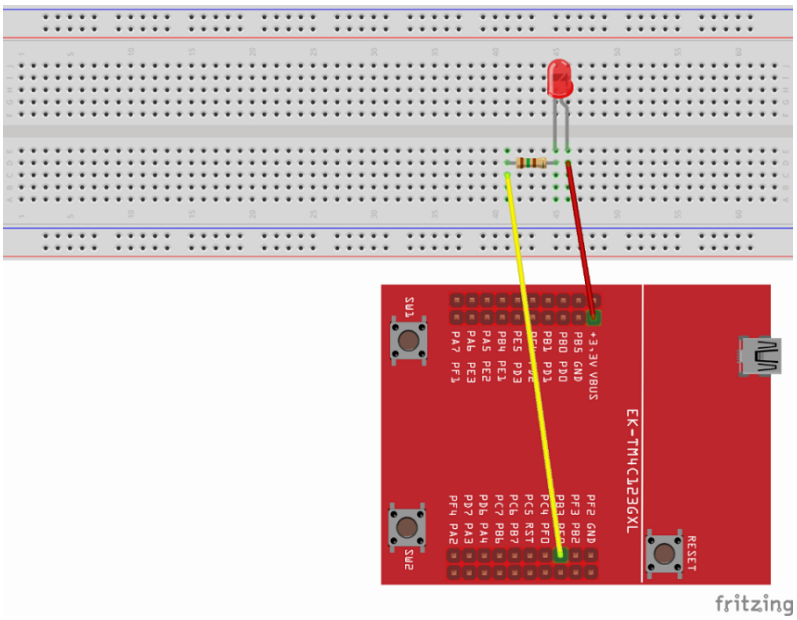
Welche Aufgaben erfüllt ein Linker-Skript?

Ein Linker-Skript konfiguriert die Einstellungen für ein Energia C++ Programm

Was beschreiben die einzelnen Zeilen in dem Block *Memory*?

In dem Block wird die Startadresse, die Speichergroße und die Lese-Schreib-Rechte für den Flash-Speicher und den RAM definiert. Der Flash-Speicher darf Lesen und Ausführen, beginnt bei der Adresse 0x00000000 und endet bei der Adresse 0x00040000 und die Speichergroße beträgt 256kB. Der RAM darf Lesen, Schreiben und Ausführen, beginnt bei der Adresse 0x20000000 und endet bei der Adresse 0x20008000 und die Speichergroße beträgt 32kB.

4 Aufgabe 4



Schreiben Sie ein Programm, das mittels einer LED ein SOS signalisiert. Bauen Sie dafür die abgebildete Schaltung auf (150Ω Widerstand, LED wird angeschlossen an den Pin PB_3) oder verwenden Sie wie oben die Onboard-LED. Strukturieren Sie Ihr Programm dabei *sinnvoll* zur Wiederverwendung mittels Verwendung von Funktionen.

Als erstes wurde der Angesprochene Pin PB_3 und die Geschwindigkeiten $SPEED\_S$ und $SPEED\_O$ definiert. In der Funkton *setup* wird das Ansprechen des Pins vorbereitet. Der Funktion *character* muss ein Integer übergeben werden, dieser bestimmt wie lange die LED an bleibt. Zum Schluss wird in der Funktion *loop* in for-Schleifen die Funktion *character* aufgerufen um S.O.S zu blinken.

```
#define PIN PB_3
#define SPEED_S 500
#define SPEED_O 2000

void setup() {
  // put your setup code here, to run once:
  pinMode(PIN, OUTPUT);
}

void character(int speed) {
  digitalWrite(PIN, LOW);
  delay(speed);
  digitalWrite(PIN, HIGH);
  delay(speed);
}
```

```
void loop() {
  for (int x = 1; x <= 3; x++) {
    character(SPEED_S);
  }
  for (int x = 1; x <= 3; x++) {
    character(SPEED_O);
  }
  for (int x = 1; x <= 3; x++) {
    character(SPEED_S);
  }
  delay(2000);
}
```

5 Aufgabe 5

Lokalisieren Sie in der Energia-Umgebung die folgenden Dateien:

- Energia.h
- main.cpp

und vollziehen Sie die Funktionsweise des Programmablaufs (s. main.cpp) nach.

Die Energia.h und die main.cpp Datei befinden sich in folgenden Ordner
C:\Program Files\energia-1.6.10E18\hardware\energia\msp430\cores\msp430
In *Energia.h* befinden sich nur Definitionen (z. B. HIGH und LOW). Dort finden sich auch Definitionen von Funktionen wie *delay* und *sleep*. In der *main.cpp* wird die *Energia.h* Datei inkludiert und die *main* Funktion definiert. In der *main* Funktion wird zuerst die init() Funktion aufgerufen. Dann die vom Programmierer erstellte *setup()* Funktion aufgerufen, so wie die *loop()* Funktion, welche sich in einer Endlosschleife befindet.

Warum reicht es aus, nur die Funktionen *setup* und *loop* für ein Energia-Programm (Sketch) zu implementieren?

Durch die Beschreibung der *main* Funktion wird auch klar wieso es reicht wenn man nur *setup* und *loop* implementiert. Diese beiden Funktionen werden in der *main* Funktion aufgerufen.