NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4258 Low-Level Programming
Laboratory Report

Exercise 1

Group 7:

Vegard Seim Karstang
Johannes Andersen
Daniel Fedai Larsen

September 16, 2016

1 Overview

In this exercise we were tasked with programming the EFM32GG-DK3750 to allow the user to control the LEDs on the gamepad provided. We implemented this by writing assembly code for the ARM Cortex-M3 and using the virtual machine provided by the course to upload and debug the code we wrote. The requirements for the exercise were not only that the LEDs should be controllable, but that it should first be implemented using polling, and later, if we we were able to, with interrupts which the ISA for the ARM Cortex-M3 provides. Laslty, if we had time, we were to activate sleep modes to drastically decrease the power consumption.

1.1 Baseline Solution

Out baseline solution sets up the initial state by first enabling the GPIO clock. The drive strength of the GPIO A pins are set to HIGH (20 mA), and output is enabled on pin 8 - 15 on GPIO A. Afterwards input is enabled on pin 0 - 7 on GPIO C, and the internal pull-up resistors on GPIO C are enabled because the gamepad does not contain these. The system then changes state and starts polling the button states of the gamepad. During polling it first turns off all lights. Then it fetches the state of all buttons and turns on the correct lights according to which buttons are pressed.

1.2 Improved Solution

The improved soultion starts with setting up get GPIO in the same way the baseline solution does. However it does not start polling. Instead it enables GPIO interrupts for both press and release of buttons. When an interrupt is received the lights are reset and the state of the buttons are read. It then turns on the correct lights according to which buttons are pressed. We also added going into deep sleep while waiting for interrupts.

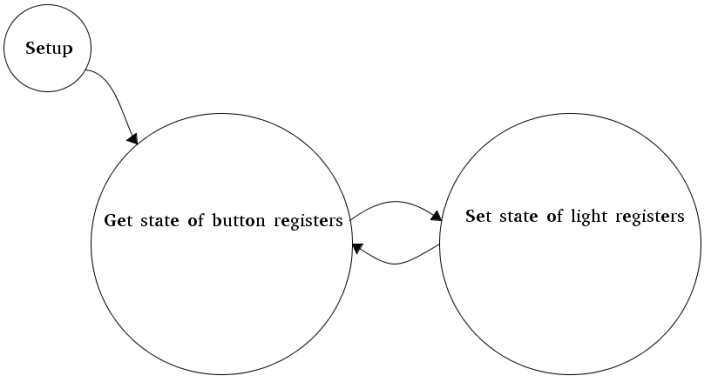

Figure 1.1: FSM of the baseline solution (polling)

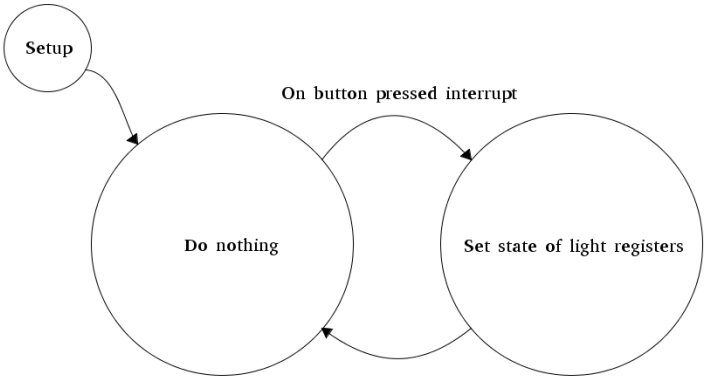

Figure 1.2: FSM of the improved solution (interrupts)

2 Energy Measurements

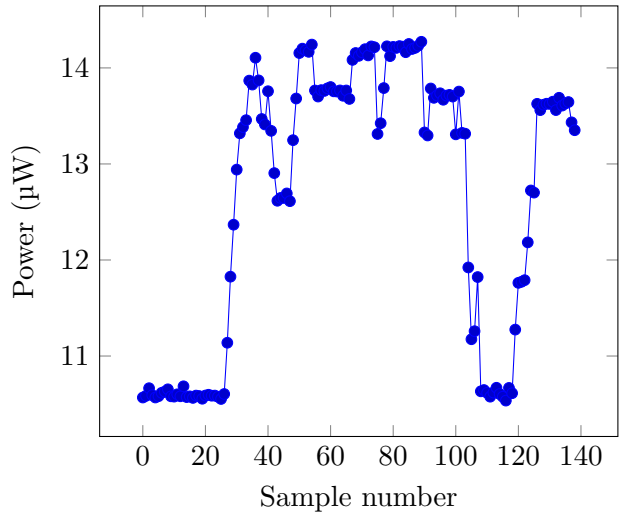

Figure 2.1: Power consumption plot generated from eAProfiler data when using polling.

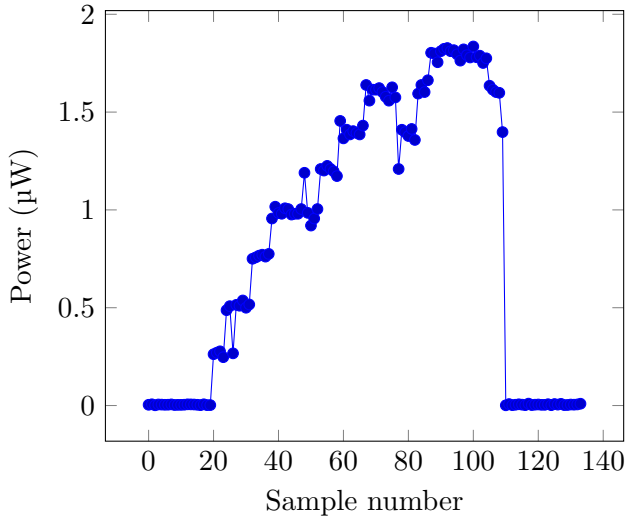

Figure 2.2: Power consumption plot generated from eAProfiler data when using interrupts combined with deep sleep.

If we use Figure 2.1 as a baseline when using polling to control the LEDs, we see

massive improvements in power consumption when using interrupts with sleep activated. When looking at our EFM32GG microcontroller, the consumtion went from 3mA to 1.4 µA. It is worth noting that we are using the highest available drive mode when turning on the LEDs, which may result in a little bit more power consumption (even though we have the pin set to not take this into consideration). The reason for the drastic change in power consumption is caused by deep sleep. When using interrupts, but without deep sleep, we still see a decrease in power consumption, but only by approximately 2 mA. We can therefore say that deep sleep, in combination with interrupts, is the most important distinction and is very important to implement.