NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4258 Energy Efficient Computer Design
Laboratory Report

Exercise 2

Group 5:

Bakken, Alexander
Hov, Einar
Lindgren, Joakim

October 19, 2015

Abstract

1 Introduction

In this report we will talk about and discuss how to make sound effects by interacting with different buttons using the C programming language.

For this exercise, we were supplied with a motherboard, a prototyping board and the GNU C Compiler; GCC with the corresponding documentation, explaining how C files could be linked with the gcc command. We were also supplied with a custom gamepad and manuals for all of these parts. In addition to this, we got a thorough explanation on how to implement interrupts handling and other useful tips for HW timers in C.

The exercise specified that we should write a C program that runs directly on the development board without the support of an operating system and which plays different sound effects when different buttons are pressed. Three different sound effects should be made. In order to complete this exercise we needed to understand the the underlying fundamental principles and functions of the sound generator: Digital to Analog Converter (DAC). The EFM32GG reference manual provided was in good stead.

2 Background and Theory

The problem is to be solved on the EFM32GG-DK3750 Development Kit from Energy Micro. The kit contains the EFM32GG990F1024 MCU, a motherboard with various peripherals, and a prototyping board for connecting external devices. [**?**] Further we are also using a custom gamepad made specifically for the course. To develop and profile the software for the kit we are using the GNU toolchain and selected tools from the energyAware software suite.

2.1 The MCU

The MCU contains a CPU core from the ARM Cortex-M3 family, 1MB of Flash memory, 128kB of SRAM, a debug interface, an interrupt controller and various peripherials. Amongst the peripherials are clocks, I/O ports and timers. For solving this problem we will only need peripherals from these categories, but the MCU also contains serial interfaces, analog interfaces, an energy management unit and hardware acceleration for AES encryption. The Flash memory, SRAM and peripherals are all mapped to the CPU's system memory map. [**?**]

The MCU operates in different energy modes, named EM0 through EM4. At EM0, called Run Mode, the CPU and all peripherals are active. At EM1, called Sleep Mode, the CPU is halted in a sleep mode, but all peripherals are still active. In higher modes, sets of peripherals are deactivated to further lower energy consumption. Entering a lower energy mode must be done in software, using the wfi and wfe instructions. Returning to EM0 and waking the CPU is initiated with an event or interrupt to the CPU. [**?**]

2.2 Motherboard

The kit's motherboard hosts the MCU module and the prototyping board, along with various other peripherals. Of relevance to this exercise is the Advanced Energy Monitoring (**AEM**) system, which provides tracking of energy consumption. There is also an LCD display, buttons and a joystick which are used to configure the kit and display an energy consumption graph. Energy consumption can also be tracked from a computer connected over USB. [**?**]

2.3 Prototyping Board and Controller

The prototyping board provides contacts for connecting external circuits to the dev kit. [**?**] For this exercise we are connecting the controller to two sets of GPIO pins on the

board, each set consisting of 8 pins. One set of pins is used for input from the controller's 8 buttons, while the other set is used for setting 8 LED lights on the controller board.

2.4 Software

As mentioned two sets of software were primarily used to develop the program for the dev kit. The GNU toolchain is a set of tools for building executable code from source code. The toolchain also provided a debugger that was used in conjunction with a debugging server from the energyAware software suite to debug the program on the MCU. Other tools from the energyAware software suite was used to upload the program to the MCU's flash memory, and to read energy consumption information from the AEM. [**?**]

3 Methodology

4 Results

5 Conclusion