COMPUTER ORGANIZATION (ELE408) LAB 1

Introduction to Freescale TWR-K70F120M tower system and Design Tools

1. Pre-lab Report

- Read Freescale MQX RTOS design platform training materials carefully
- Read Freescale MQX design tool Reference Manual.
- *Read this lab handout carefully. Take notes and prepare the programs required in this handout.
- Study the Codewarrior design platform and Freescale MQX RTOS tool.

Items marked by "*" must be included in a written pre-lab report. The pre-lab must be shown to the Teaching Assistant at the beginning of each lab section. The prelab must be initialed by the TA and attached as an appendix in the final report. This rule applies to all labs.

2. Objectives of this Lab

This lab is intended to familiarize you with the Freescale's Kinetis evaluation board with ARM Cortex M4 microprocessor, the TWR-K70F120M, and several modern engineering tools for development of embedded computer systems. You will apply your knowledge of mathematics, science, and engineering that you have learnt in your previous classes to the lab experiments. Examples of such knowledge include different number systems, programming languages (assembly & C), hardware concepts, and programming skills. Your knowledge together with the modern tools available in the lab settings will enable you to design and perform simple experiments, as well as to analyze and interpret data.

Specifically, you will learn and exercise the basics of

- Freescale MQX RTOS system development set,
- ARM Cortex M4 processor addressing modes,
- ARM Cortex M4 processor memory organization,
- dBUG operating system,
- Codewarrior and Freescale MQX

Freescale Kinetis boards can be used either in stand-alone mode or tower system mode. You will learn different clock settings for different work modes.

LEDs and Switches are connected to GPIO pins of the corresponding microprocessor.

Basically, each LED and switch wired to a resistor and a GPIO pin serially. To turn on a LED, you need to first assign its related pin as output, and then set the pin to Logic 1. This is a common output control pattern that you will use often. Setting a pin for output requires first writing a "1" to the appropriate bit in the Ports DDR. Next, to put the LED in ON state, a "1" is written to the Ports Data Register.

Why does the LED turn ON when its control pin is at logic 1 level?

3. Basics

In this lab, you will learn how to initialize and start MQX design platform, and familiarize yourselves with the basics of MQX tasks through hand on experiments on GPIO on K70 board. Specifically, you will learn how to program and access LEDs and switch on the board.

- Initializing and starting MQX:

Set up and initialize data that MQX uses internally including the default memory pool, ready queues, the interrupt stack, and task stacks.

1) initialize the hardware (for example, chip selects).
2) enable timers.
3) set the default time slice value.
4) create the Idle task, which will be active if no other task is ready.
5) create tasks that the task template list defines as autostart tasks.
6) starts scheduling the tasks.

- Task Template list

The task template list, which is a list of task templates (**TASK_TEMPLATE_STRUCT**), defines an in itial set of templates, from which tasks can be created on the processor.

At initialization, MQX creates one instance of each task, whose template defines it as an autosta rt task. while an application is running, it can create other tasks using a task template that eithe r the task template list defines or the application defines dynamically.

The end of the task template list is a zero - filled task template.

```
typedef struct task_template_struct
{
        _mqx_uint TASK_TEMPLATE_INDEX;
        void _CODE_PTR_ TASK_ADDRESS)(uint_32);
        _mem_size TASK_STACKSIZE;
        _mqx_uint TASK_PRIORITY;
        char _PTR_ TASK_NAME;
        _mqx_uint TASK_ATTRIBUTES;
        uint_32 CREATION_PARAMETER;
        _mqx_uint DEFAULT_TIME_SLICE;
} TASK_TEMPLATE_STRUCT, _PTR_ TASK_TEMPLATE_STRUCT_PTR;
```

When you assign task priorities in the task template list, be aware that:

1) MQX creates one ready queue for each priority to the lowest priority (highest number)
2) While an application is running, it cannot create a task that has a lower priority (a higher number) than the lowest priority task in the task template list.
3) If you assign priority zero to a task, the task runs with interrupts disabled.

You can assign any combination of the following attributes to a task:

1) Autostart — when MQX starts, it creates one instance of the task.
2) DSP —MQX saves the DSP coprocessor registers as part of the task's context.
3) Floating point — MQX saves floating point registers as part of the task's context.
4) Time slice —
   MQX uses round robin scheduling for the task (the default is FIFO scheduling).

Example:

```
TASK_TEMPLATE_STRUCT MQX_template_list[] =
{
        {   MAIN_TASK, world_task, 0x2000, 5, "world_task", MQX_AUTO_START_TASK, 0L, 0},
        {       HELLO,  hello_task, 0x2000, 5,   "hello_task", MQX_TIME_SLICE_TASK   , 0L, 100},
        {           0,          0,      0, 0,           0,                          0, 0L, 0 }
};
```

In the lab

1. Set workspace as "C:\Freescale\Freescale MQX 3.8\demo", Import a given MQX project name in ELE408_lab1_example in this workspace
2. Follow the instructions in this link, follow the lab handout step by step, understand how to define and use an MQX task.
    1) Draw a flowchart of the status task in your lab report.
    2) Understand how the switches and LEDs work. Draw a logic relationships between switches and LEDs in your lab report.
    3) Report the result of this demo project in your lab report (screen print)
    4) Give your understanding of lwevent?
3. Modify the example project to satisfy the following requirements.
    1) Switch the function of two temperature switches.
    2) Read tsi.c and tsi.h file in this project. Add TSI functions to this project (by callingsome functions in tsi.c in your tasks).
    3) Touch the LEDs and find the function of TSI and add the following functions to touch sensors: (Hint, just call some function in tsi.c file)
        - When LED1 is touched, set the HVAC to HVAC_Heat mode.
        - When LED3 is touched, set the HVAC to HVAC_Auto mode.
        - When LED1 is touched, set the HVAC to HVAC_Cool mode.

    Make a screen shot and print your result in your lab report and discuss the results

5. Lab Report Requirements

In your lab report, you should discuss your designs, trade-offs between performance and power, Explanation and interpretation of your results are very important. The lab report will be graded based on your report and discussions. Total mark for the report is 100 points.

- Prelab report: 20 points
- Successful experiment: 50 points
- Results analysis, interpretation, and discussions on your design and engineering constraints: 30 points

In the following items, numbers inside each bracket indicates the point you will earn on a satisfactory report and discussions.

In your discussion and explanations of your results, you should consider the following constraints:

- What knowledge of mathematics, science and engineering have you applied in this lab and what tools have you used in this lab?[5]
- Economic Constraint (performance/cost ratios) [5]
- Manufacturability, Modularity and Expandability Constraints, Environmental Constraint (power consumption) [5]
- Sustainability: Is your design and implementation sustainable? [5]
- What is the potential impact of your design on real time applications? [5]
- How would you utilize the memory hierarchy available to you to design real time applications? [5]

For each of the above programs hand in the debugged source code with comments; the machine code is not necessary. Be very specific with your comments that explain what you are doing and why you are doing it.