***Efficient Embedded Course***

**LAB 3**

**INTERRUPT LAB EXERCISE:**

**STACK USE AND TIMING BEHAVIOUR**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc87593058)

[1.1 Lab overview 1](#_Toc87593059)

[2 Learning Outcomes 1](#_Toc87593060)

[3 Requirements 1](#_Toc87593061)

[4 Hardware Setup 2](#_Toc87593062)

[5 Analysis 2](#_Toc87593063)

[5.1 CPU Behaviour 2](#_Toc87593064)

[5.1.1 CPU state when entering handler 2](#_Toc87593065)

[5.1.2 CPU State After Exiting Interrupt 3](#_Toc87593066)

[5.2 Timing 3](#_Toc87593067)

[5.2.1 Observe Overall CPU Timing Behaviour 3](#_Toc87593068)

[5.2.2 Observe Detailed CPU Timing Behaviour 3](#_Toc87593069)

[5.2.3 Observe Even More Detailed CPU Timing Behaviour 3](#_Toc87593070)

[5.2.4 Preemption of Main Code 4](#_Toc87593071)

# Introduction

## Lab overview

In this lab, you will evaluate the behaviour of a system with an interrupt. You will use the interrupt demonstration code from lecture.

# Learning Outcomes

* Write an interrupt-driven program using an interrupt service routine (ISR).
* Analyse CPU timing behaviour via debug signals
* Use a debug tool to observe the state of the CPU when entering interrupt handling state.

# Requirements

In this lab, we will be using the following hardware and software:

* **Keil µVision5 MDK IDE**
  + Please see the included Getting Started with Keil guide on how to download and install Keil.
* **STM32 Nucleo-L552ZE-Q**
  + For more information, click [here](https://www.st.com/en/evaluation-tools/nucleo-l552ze-q.html).
* **Logic Analyzer or Oscilloscope** 
  + Required to monitor the interrupt signals.

# Hardware Setup

Connect the switch signal to the GPIO port input on the MCU as shown in table below. Connect the debug signals and the switch signal to a logic analyzer or oscilloscope. This matches the pins used in the supplied code.

Table 1. Signals and connections

|  |  |  |  |
| --- | --- | --- | --- |
| Signal Name | Description | Direction | MCU |
| SW1 | Switch Input | Input to MCU | PC\_13 |
| VSS | Ground |  |  |
| DBG\_Main | Main Thread Debug Output | Output from MCU | PC\_6 |
| DBG\_ISR | ISR Debug Output | Output from MCU | PC\_8 |

Please see the Nucleo-L552ZE-Q User manual for the pinout of the Arduino-included Zio connectors for CN7, CN8, CN9 and CN10 using this link: https://www.st.com/resource/en/user\_manual/um2581-stm32l5-nucleo144-board-mb1361-stmicroelectronics.pdf

# Analysis

* Compile and load the code onto the board.
* Start the debugger.
* Enable the disassembly window (View->Disassembly Window)
* Set a breakpoint at start of handler function.
* Run the program, and then press the switch SW1.

## CPU Behaviour

### CPU state when entering handler

Examine the stack and CPU registers with the debugger.

1. Complete the table below to show the values of the CPU registers and state information.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Register** | **Value** | **Register** | **Value** | **Register/State** | **Value** |
| R0 |  | R8 |  | xPSR |  |
| R1 |  | R9 |  | MSP |  |
| R2 |  | R10 |  | PSP |  |
| R3 |  | R11 |  | PRIMASK |  |
| R4 |  | R12 |  | CONTROL |  |
| R5 |  | R13 (SP) |  | Mode |  |
| R6 |  | R14 (LR) |  | Privilege |  |
| R7 |  | R15 (PC) |  | Stack |  |

1. Complete the table below to show what information is on the stack. Open a memory window (View->Memory Windows->Memory 1) and enter SP as the address. Right-click on the window and specify Unsigned->Int as the display format.

|  |  |  |
| --- | --- | --- |
| Address | Value | Description |
| (SP) |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### CPU State After Exiting Interrupt

Single step out of the handler, verify return address.

## Timing

* Now connect the debug signals to a logic analyzer or oscilloscope.
* Disable the breakpoint in the handler function, and other breakpoints you may have added.
* Resume program execution.

### Observe Overall CPU Timing Behaviour

Use the falling edge of the Switch input to trigger the data capture/sweep. Set the time base of the logic analyzer (or oscilloscope) so that a switch press covers about one fourth of the screen. Capture a screenshot showing the switch signal, DBG\_ISR and DBG\_MAIN.

1. How long was the switch pressed down?
2. Is there any noticeable delay between the switch being pressed and the ISR running?
3. Does the DBG\_MAIN indicate that main stops running at any time?

### Observe Detailed CPU Timing Behaviour

Now zoom in so that the screen displays about 100 us, centered on the ISR.

1. How long is the DBG\_ISR signal asserted?
2. Is there any noticeable delay between the switch being pressed and the ISR running?
3. Does the DBG\_MAIN signal indicate that main stops running at any time?

### Observe Even More Detailed CPU Timing Behaviour

Now zoom in so that the screen displays about 10 us centered on the ISR.

1. How long is the DBG\_ISR signal asserted?
2. Is there any noticeable delay between the switch being pressed and the ISR running? How many clock cycles does this correspond to? How does this compare to what you would expect?
3. Does the DBG\_MAIN signal indicate that main stops running at any time? If so, calculate for how long.

### Preemption of Main Code

Now zoom in so that the screen displays about 100 us centered on the ISR.

1. For how long is the main function delayed? First measure the pulse width of the DBG\_MAIN output signal before the switch is pressed. Then measure the pulse width when main is preempted by the ISR. The difference indicates the total preemption time.
2. How long is the total preemption in comparison with the duration of the DBG\_ISR signal? If the two times aren’t the same, explain why.