EEE3095/6S: Practical 3

STM32 Profiling and Benchmarking

September 5, 2025

1 Overview

In this practical, you will switch to STM32F4, which offers greater processing power compared to STM32F0.

This practical is more open ended(You will notice there are few todos in the code as compared to the previous pracs) and will involve more benchmarking and comparison of the performance between STM32F0 and STM32F4.

The clock has been configured to run at:

- 120Mhz for the STM32F4
- 48Mhz for the STM32F0

Since this practical is open ended you are welcome to change that in the ioc file, as long as you do not break the code.

In line with that, if you find the provided files insufficient you are welcome to create your own STM32 CUBE IDE project to do the benchmarking, but just be sure you know what you are doing.

Hardware Setup

1. Solder the pin headers onto your STM32F4 module. Don't solder pins to the highlighted row headers shown below in Figure 1:

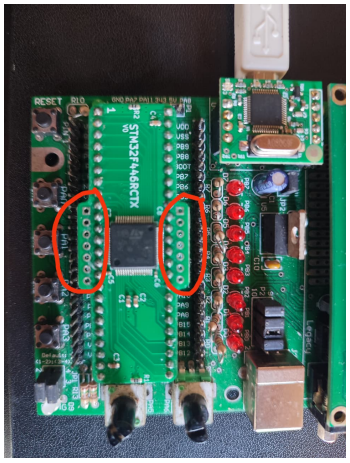

Figure 1: Out of Bounds Row Headers

2. Plug the STM32F4 into your development board, matching the orientation of the STM32F0 socket. (Pin 1 of STM32F4 should plug in exactly where pin 1 of STM32F0 was plugged)

   Note: *Most GPIOs map directly from the F0 to the F4, apart from I2C and LCD pins.(You don't need to worry about that for now, as you won't be using it for this practical)*

3. Once plugged in, the development board should resemble the one in Figure 2 below:

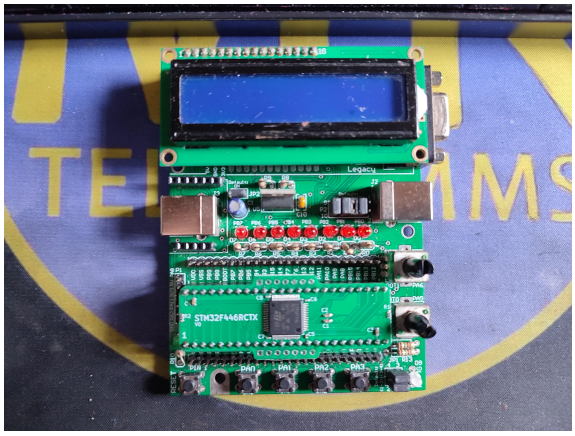

Figure 2: STM32F4 Development Board

4. To verify your board is functional, flash the "Hello World" code in folder Practical_3A to your board, and you should see the LEDs toggle. Check HelloWorld

2 Outcomes and Knowledge Areas

In this practical, you will profile the performance of a simple Mandelbrot run on the STM32F0 and the STM32F4.

The objectives of the practical are:

1. Run the same tasks as Practical 1B on STM32F4.
2. Benchmark how max iterations affects Mandelbrot computation performance.

3. Extend execution time measurement through measuring clock cycles, compute throughput (pixels/sec).
4. Test scalability (increasing image size up to HD/Full HD or until memory limits).
5. Measure effects of enabling/disabling FPU and using float vs double on the STM32F4.
6. Test compiler optimisation levels.
7. Compare fixed-point arithmetic scaling choices.
8. (Advanced) Attempt to measure power usage.

3 Deliverables

For this practical, you must:

- Develop the code required to meet all objectives specified in the Tasks section
- Push your completed code to a shared repository on GitHub
- Write a short report documenting your implementation. This must be in PDF format and submitted on Gradescope with the naming convention(**If you do not adhere to the naming convention, there will be a penalty**):

  EEE3096S 2025 Practical 3 Hand-in STDNUM001 STDNUM002.pdf

  Check the Appendix for Report Structure 5.

Note: Your code (and GitHub link) should be copy-pasted into your short report so that the text is fully highlightable and searchable;**do NOT submit screenshots of your code or you will be penalised.**

4 Getting Started

The procedure is as follows:

1. Clone or download the Git repository(The practical folder(s) of interest is Practical3_F0 and Practical3_F4):
   git clone https://github.com/EEE3096S-UCT/EEE3096S-2025.git
2. - Open **STM32CubeIDE**, then navigate through the menus:

     `File` → `Import` → `Existing Code as Makefile Project`

   - Click `Next`.
   - In the dialog, click `Browse...` and select your project folder.
   - Under **Toolchain**, choose `MCU ARM GCC`.
   - Click `Finish`.
3. In the IDE, navigate and open the **main.c** file under the Core/src folder, and then complete the Tasks below.

5 Tasks

Have one commit per task as it will make it easier for the tutors to evaluate your code for each task.

Task 1: Welcome to Practical 3 (5%)

To be done on only the **STM32F4**.

1. Port the Mandelbrot code from **Practical 1B** to the **STM32F4** platform.
2. Run the code on STM32F4 and **log the performance benchmarks** (execution time and checksum) for the **same image sizes as in Practical 1B**.

Task 2: Impact of Maximum Iteration Variable (10%)

To be done on both the **STM32F4** and **STM32F0**

1. Choose at least **5 MAX_ITER values** between **100** and **1000** (e.g., 100, 250, 500, 750, 1000).
2. For each MAX_ITER value and for each image size used in **Practical 1B**, run the Mandelbrot program and record:
    - Execution time
    - Checksum

Task 3: Extended Execution Time Measurement (15%)

To be done on both the **STM32F4** and **STM32F0**

1. In addition to wall-clock time, measure and log:
    - CPU clock cycles
    - Throughput in pixels per second
2. Use any valid approach such as the DWT cycle counter.
3. **Constraints:** MAX_ITER = 100; Image sizes must match those from Practical 1B.

NB: The execution time refers to the execution tome of the Mandelbrot function.

Task 4: Scalability Test (20%)

To be done on both the **STM32F4** and **STM32F0**

1. Gradually increase image size up to Full HD (1920 × 1080).
2. If memory is insufficient:
    - Document the method used to split the image and process it in parts.
    - Explain why splitting was necessary.
3. **Constraint:** MAX_ITER = 100.

Task 5: FPU Impact (10%)

To be done on only the **STM32F4**.

1. Build and run the Mandelbrot code with FPU enabled and then with FPU disabled on STM32F4.
2. Perform this test using:
    - `float` variables in the Mandelbrot function.
    - `double` variables in the Mandelbrot function.
3. Compare and log:
    - Accuracy differences
    - Speed-up

- Execution Time

4. Modify the Makefile in the line starting with **FPU** = to enable/disable FPU.
5. **Constraints:** MAX_ITER = 100; Image sizes must match those from Practical 1B.

Task 6: Compiler Optimisations (10%)

To be done on both the **STM32F4** and **STM32F0**

1. Test at least 3 compiler optimisation levels (e.g., `-O0`, `-O2`, `-Os` or `-O1`, `-O2`, `-O3`).
2. You can achieve this by:
   - Editing the Makefile in the line starting with **OPT** =, or
   - Using IDE settings (Project → Properties → C/C++ Build → Settings → MCU Settings).
3. For each optimisation level, record:
   - Binary File size
   - Runtime measurements(Total Time it takes to run the whole program)
4. **Constraints:** MAX_ITER = 100; Image sizes must match those from Practical 1B.

Task 7: Fixed Point Arithmetic Scaling Factor (10%)

To be done on both the **STM32F4** and **STM32F0**

1. Implement the Mandelbrot algorithm using Fixed-Point Arithmetic.
2. Test at least 3 different scaling factors (e.g., 10^3, 10^4, 10^6).
3. For each scaling factor, document:
   - Effect on precision
   - Risk of overflow
   - Execution speed
4. **Constraints:** MAX_ITER = 100; Image sizes must match those from Practical 1B.

Task 8: Power Measurement Attempt (10%)

To be done/explained for both the **STM32F4** and **STM32F0**

1. Attempt to measure the power consumption during Mandelbrot benchmarking on STM32F0 and STM32F4 for one test case.
2. Use the collected data to extrapolate for other test cases.
3. If measurement is not possible, explain:
   - Why it cannot be done
   - What tools or setup would be required

Report Structure

Structure your report as follows:

1. Introduction
2. Task Number and Title .Check Table 1 for guidance
3. Conclusion
4. AI clause

Component	Weight	Short description
Objective	10%	Clear task objective description
Methodology & Setup	20%	Detailed steps/approach taken to achieve the objectives, relavant code snippets etc
Results	30%	Measured data in labeled tables and/or figures
Analysis & discussion	40%	Correct interpretation, quantitative comparison, explanation of anomalies and limitations etc

Table 1: Task Report Section Structure

5. References
6. Code(The last commit version)

The page limit for this report is 8 pages excluding Appendices. You are still required to use the IEEE Conference template provided on Amathuba Report Templates.

Report Mark Allocation

Task	**Mark Allocation(%)**
Introduction	4
Task 1	5
Task 2	10
Task 3	15
Task 4	20
Task 5	10
Task 6	10
Task 7	10
Task 8	10
Conclusion	4
AI Clause	2