Assignment 2 - Exploration

Andrew N. Sloss

Abstract

This Assignment is the **Exploration** stage. The main objective is to create a *theoretical* baseline of capabilities for the target-board. This means determining the details about *target-board* and software, list out the features, options and potential limitations.

Contents

1 Objective

This is the **Exploration** stage. The main objective is to create a *theoretical* baseline of capabilities. This means determining the *target-board* (and software) features, options and limitations. This assignment builds upon what was learnt from Assignment 1. The prerequisite is that Assignment 1 was successful, which means you can create basic C code, have a working development environment (Mac/PC/Ubuntu/...) and can compile or cross compile 'C' code targeting the Raspberry Pi 3. In this assignment we're going to characterize the target-board hardware and software by using the 'C' programming language and search for extra information (e.g. using manuals, internet, books, etc.).

You are building a one off prototype for this course but put down some thoughts if you had to scale-up the prototype to say 1000 units (short production run) or even 10,000 units (a mass production run). And how this would affect the questions being raised.

The expected output is 'C' source code and a report. Record all the procedures and observations include both successes and failures.

2 Background

The characterization of hardware and software on a *target-board* is an essential part of Embedded Systems. The procedure provides a method to determine capability, limitations and fit-for-purpose. This is experimental "scientific" exploration, use your background knowledge to make an assessment of the target-board. We are specifically interested in the hardware and software attributes that affect Embedded Systems and Real-Time responses. This information is used to determine whether a target-board can satisfy a set of requirements.

2.1 Estimated Time

It is estimated that the time to complete this assignment is around 5-6 hours. Make sure you read through entire document before focusing on the details of the exercise.

2.2 Terminology

Device	An external piece of hardware
Peripheral	An hardware function built either on the SoC or board
Distribution	A wrapped collection of Linux software, packages and management
HAT	Hardware Attached on Top
NOOBS	New Out Of Box Software

2.3 Nomenclature

$\mathbf{P}_x$	Pin X on the 40-pin Raspberry Pi header.

3 Feedback

If you see an error or anything that requires further explanation or is plain wrong please let me know so that others may benefit from your observations. Engineering is about constant improvement.

4 Important

Don't just pull the power from the device. This is a Linux device and pulling the power randomly risks corrupting the file system. Interesting aside, the ext2 filesystem we will use is a filesystem with journaling turned on which has *some* measure of fault tolerance. The best practice is to force a software halt first and

then pull the power. The halt forces Linux to be placed in a dormant safe state for power removal. Use this command before pulling the power but always remember to wait a few minutes until Linux has fully shutdown.

```
$ sudo halt
```

Be aware that the pins on the Raspberry Pi are highly sensitive. It is easy to damage the board if you connect the wrong electrical inputs to a pin, or even static discharge. You might consider adding a **HAT** board to protect the *Raspberry Pi* against potential future damage.

5 Recommendations

Even-through this assignment is individual, work as a team to gain information. Make sure you note and record the source of the information. Try to approach this assignment in a systematic scientific way and where possible implement your own ideas to determine the characteristics. A good method is to think about how would you assess this board for use in your own company or for your own personal use. Make notes on the time-taken and any challenges or difficulties encountered. It is expected that you follow an Embedded C Coding Style (see section [6]).

Focus on parts of the target-board from the perspective of *memory-hierarchy*, *hardware and hardware extensions*, *communication*, *response-time*, *software-tools plus libraries*, *power-management*, *reliability*, *safety-critical* and/or *improvements to the development environment*. Explore!

Note: Use this time to help you focus on your main project (next assignment), pick the parts which are important for you to understand.

6 Coding Standards

Try to keep to one coding standard and specify the standard that you have adopted.

- JPL Coding Standards
- MISRA C
- C99 Standard

You can use your own coding standard but be careful to specify the rules in detail. Be consistent with the decisions made. Consistency is very important with Embedded Systems.

7 Pin Control Library

There are many libraries available that can be used to control the *Raspberry Pi 3* pins. Three of the more popular libraries are *BCM2835*, *WiringPi* and *PiGPIO*. Note not all PIN libraries are the same. It will be worth your time taking a look at these libraries to determine viability, capabilities and usefulness for your major project. Maybe worth creating a table of features. Note if you find any other PIN libraries for Raspberry Pi, explore and determine whether it has any useful features.

7.1 Example: BCM2835 Pin Library

The rest of this section explains how to download and install the *BSM2835* library (note, this library isn't necessarily this best library but provides an example). This library was designed for Raspberry Pi 2 and works on the Raspberry Pi 3, it provides a simple control interface to the **GPIO** pins. The quality of this library is unknown. The library states that it provides control of **SPI**, **I²C** and **PWM**. The next steps are to download the library and configure.

If you choose to download and configure an alternative PIN library, make sure you record the procedure and all the necessary resource links.

7.1.1 Read library description

Description of the library can be found from this URL:

http://www.airspayce.com/mikem/bcm2835/

7.1.2 Download library

Download the library from this URL:

http://www.airspayce.com/mikem/bcm2835/bcm2835-1.50.tar.gz

7.1.3 Build the library

Follow this following procedure. First download the latest version of the library and copy it over to the target-board, say bcm2835-1.xx.tar.gz (ungzip the file) then:

```
$ tar xvf bcm2835-1.xx.tar
$ cd bcm2835-1.xx
$ ./configure
$ make
$ sudo make check
$ sudo make install
```

You now should have a built library. Now configure the Raspberry Pie to allow the exposure of the Device Tree. This will involve you going back into the desktop mode and running the following command within the desktop.

```
$ sudo startx
```

The standard desktop should appear and now configure the Raspberry Pi with Device Tree. Note it maybe already enable but double check.

```
$ sudo raspi-config
```

Under the advanced options choose “enable Device Tree”. Once enabled and saved reboot the Raspberry Pi 2.

Other useful links:

http://elinux.org/RPi_Low-level_peripherals
http://www.scribd.com/doc/101830961/GPIO-Pads-Control2
http://www.airspayce.com/mikem/bcm2835
https://github.com/momenso/node-dht-sensor

8 Characterization questions

First at a high level, decide what information you want to obtain about the target board e.g. hardware type, memory speed, real-time response capability, software libraries availability, quality of PWM control, Pin functions, Power consumption, etc.

Once you have decided on the high-level information you want to capture, the next step is to define the questions you want to answer. The following questions are a set of examples you could ask about the target-board. These questions are in no particular order (no logical structure) and are not exhaustive. Treat them as a guideline example for characterization questions. Part of the goal here is for you to create your own questions and each persons should have an unique list due to experience and interest.

1. What SoC is used on the Board? What version and manufacturer of the SoC?
2. What features are present on the board/SoC?
3. What size caches are present?
4. What is the pin layout on this specific version of the target board?
5. How does the board boot? What is the process?
6. What software is available to enhance the pins? Test the software?
7. How much memory is available on the board?
8. What is the maximum memory configuration?
9. How many different types of memory are on the board?
10. What is the performance of the memory?
11. What version of the Operating System is being used?
12. What version of the compiler is being used?
13. How long does it take for a context switch?
14. What is involved in adding an *on-switch* to the Raspberry Pi?
15. What is the interrupt latency for the board? E.g. from an interrupt occurring to it being serviced?
16. How much time does it take to copy 1 KB, 1 MB, and 1GB in bytes, half words, and words in RAM?
17. How much time does it take to copy 1 KB, 1 MB, and 1GB in bytes, half words, and words on the filing system?
18. How long does it take between reboot and an active board? Is it deterministic?
19. How long does it take to halt the board? Is it deterministic?

20. Is the board reliable? If so how reliable? Code examples?
21. Find and run some benchmarks?
22. Determine the speed of integer arithmetic with your own benchmark?
23. Determine the speed of floating point arithmetic with your own benchmark?
24. Write a multithreaded example application that illustrates the producer-consumer algorithm with protected data running on multiple processors?
25. What are the operating temperature ranges?
26. What is the power consumption without load and with load?
27. What would be a useful stress test to determine reliability?
28. Does temperature effect performance?
29. Determine how much energy is being used to run a benchmark?
30. How deterministic is the board? How real-time? Code examples?

Once you have the created a question-list, attempt to programmatically obtain the answers by writing 'C' programs and using manual references to get this information. You may use other hardware like an Logic Analyzer, Oscilloscope or breadboard circuits but it depends on your precise focus & questions.
For questions that are too difficult to answer it is important to note those questions down and provide a reason why the cannot be answered. Honesty in characterization is fundamentally important because a lot of Embedded and Real-Time Systems go into Safety-Critical Devices. Knowing what we don't know can be of paramount importance.

9 Submission

Submit a report which includes the objective, the areas which you have decided to explore for a baseline, the list of associated questions, the answers to those questions and finally a conclusion plus all the references used. Include any software developed for exploration. Make sure you also include some thoughts about limited or mass production.

Also note time-taken.

10 Assessment

This assignment is worth 20 marks of the total assessment.

The assignment will be assessed as follows:

- **(5)** points for a clear objective and structured document
- **(8)** points for actual knowledge gained i.e. characterization-level
- **(5)** points the quality of the 'C' code created
- **(2)** points for insights into *limited* (1000 unit) or *mass* (over 10,000 unit) production