Add a flattened device tree-based initialization for Beagle BSP

Google Summer of Code Program 2023 Project Proposal

Aryan Sachin Karawale
askarawale@gmail.com
Veermata Jijabai Technological Institute
Mumbai, India
+91 7977488078
+91 9322086009
Discord - Aryan_Karawale#1731
Github
Linkedin

Project Abstract:

The FDT(Flattened Device Tree) is a data structure that describes the hardware platform and its configuration, providing flexibility and portability. Refactoring the FDT support for these will enhance the performance and reliability of the Beagle BSP by eliminating hardcoded values and simplifying driver implementation. Previously some work has been done by implementing some of Open FirmWare API and modeling an FDT inspired by the FreeBSD FDT. But the performance of all of the OFW Api wasn't completed. Moreover, the newly ported FDT Structure has only been refactored for the I2C driver.

This project aims to refactor the FDT for the GPIO, IRQ, SPI, and QEP drivers to the FDT. This will remove the unnecessary manual initialization of the registers making the complete utilization of the FDT. After the completion of the same, I intend to develop some unit tests to benchmark the performance of the FDT support.

Project Scope:

Medium approximately 175 hours

Project Description:

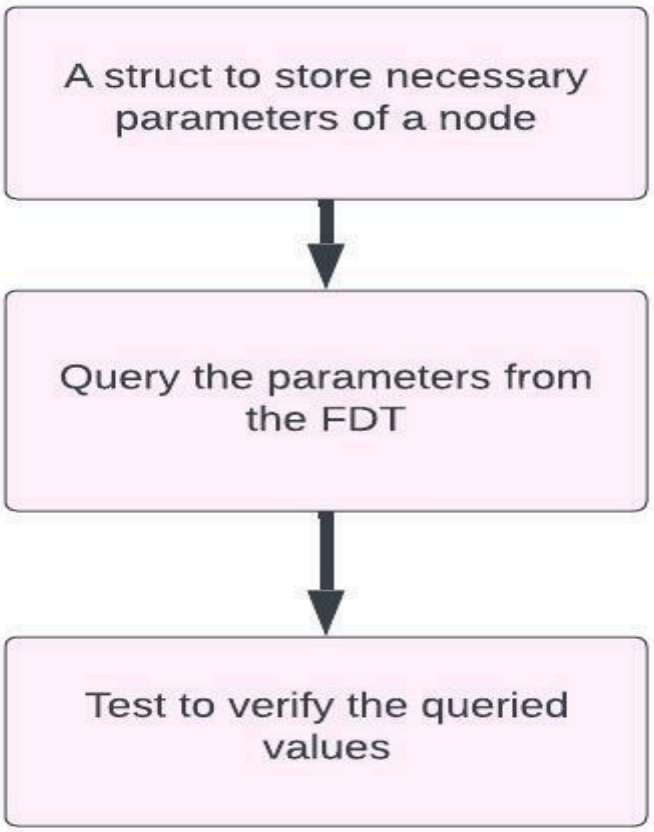
This project aims to achieve the following:

- Refactor the GPIO, IRQ, SPI, and QEP drivers to the FDT using the OFW API
- This will remove the need to initialize the registers of the drivers manually by automatically initializing them.
- Study the work done on the implementation of Open Firmware functions implemented and continue on the previous work done.
- Developing a test to benchmark the working of the FDT
- Documentation regarding the working FDT for the implemented drivers
- Tests to generate more intuitive error messages for robust debugging.

- The *bus_alloc_resources* function does not allocate Interrupt Requests to the node, which can be configured to do so. It will remove the need for local functions like (*am335x_i2c_irq_number*) in different drivers

Project Deliverables:

- Refactor the GPIO,SPI and QEP drivers to the implemented FDT
- The workflow for all these drivers can be summarised as per the following flow chart with some changes



- Provide documentation for the ofw.c file containing the Open Firmware Api ported to RTEMS
- Port the OFW Functions that still haven't been ported yet

Proposed Schedule:

April 27 - May 4 (Acceptance Waiting Period)

- Familiarize with the Beagle Bsp Codebase
- Research the possibilities to approach the project idea
- Interact with the community to develop a further understanding of the project

May 4 - May 28 (Community Bonding Period)

- Discussion on resources and tools relevant to the project with the mentors
- Understanding the working of the FDT and getting my doubts cleared
- Preparing an updated goal plan along with the mentor's

May 29 - July 14 (First Half)

- Start with refactoring the GPIO driver, this will include
    1. Writing a *am335x_gppio_fill_registers* structure to contain the necessary information about the specific GPIO drivers. It would contain the following information
        - ❖ Base reg values
        - ❖ “gpio-ranges" property describes the mapping between the GPIO controller's memory-mapped registers and the GPIO pins used by the controller.
    2. Write a function to query the register values for the GPIO driver
    3. Write a test to check whether the queried register values are correct
        - ❖ Check if the base register values are valid and belong to the same node
        - ❖ Check if the node is compatible

- Understanding the various functions implemented from OFW and explore the possibility of porting other OFW functions to further improve the working of the BSP
- Implementation of an Interrupt Request parameter to *bus_alloc_resources*
- Documentation for the drivers refactored. Along with maintaining a blog to record my weekly progress and the conversation revolving around the same
- Get code reviews from mentors and prepare for evaluation.

July 14 - September 4 (Second Half)

- Executing the improvements suggested by the mentors for evaluation 1
- I would work on refactoring the SPI driver, the implementation of this driver would be quite similar to that of the I2C driver with some minor changes,i.e the parameters in the structure which contains the parameters of any spi register it would contain additional felids like a chip select (which will contain the register value of the device that the controller is communicating with)
- Finally would start working on the QEP driver and refactor it using a method similar to that used for GPIO driver
- Add functions to display error messages for the refactored drivers to allow for easier debugging.
- Work on implementation of the porting of any other OFW functionality as
- Provide documentation for the implemented OFW API as it currently lacks one
- Completed documentation of my project
- Get feedback,code reviews by the mentors and implement the same

Future Improvements

On completion of the project, I would have gained a much better knowledge of the FDT used and hence can try out my hand at the more ambitious prospect of the project that is reallocating memory resources using FDT.

Continued Involvement

After the completion of this project I plan to contribute towards the general improvement of the beagle bsp (#2891 ticket) which is a logical next step for me. I also am interested in contributing to any other RTEMS projects which align with my interests.

Conflict of Interests or Commitment

I have no commitments during the span of the project and hence have no conflict of interest. If any do arise I would contact my mentors to figure a way out.
I have my college end-semester examinations from 15 May - 28 May and hence would be unable to contribute to the project then

Eligibility

- I have completed the Hello World Task which is a pre-requisite for contribution. https://lists.rtems.org/pipermail/devel/2023-March/074768.html
- I also possess a *Beagle Bone Black* required for testing

Major Challenges foreseen

The biggest challenge that I feel I would face is porting the OFW API to RTEMS. Moreover, refactoring of IRQ would be challenging as I feel the mapping of the interrupt numbers of the FDT to those used by the system would be quite a task.

Relevant Background Experience

I am an embedded systems enthusiast and proficient in c and c++. I have previously worked on ESP32, STM32, RasPi, and MSP432 for my previous projects. Working on these projects got me interested in FreeRTOS and its development. I have preliminary knowledge of the software and hardware components required for the testing and implementation required for the project.

Personal:

I am Aryan Sachin Karawale, an undergraduate student of Electronics and Telecommunications Engineering at Veermata Jijabai Technological Institute Mumbai, India. I have a keen interest in the inner workings of microcontrollers.

I have been curious about FreeRTOS and its inner workings which motivated me to work on an RTEMS project. I have also been wanting to work on beagle bone for quite a long time now so the Project for me is kind of a best-of-both-worlds situation. I am a beginner to the open source community but the whole idea of open source contributions fascinates me, adding fuel to my fire to make some serious open source contributions.

Experience:

Personal Projects

Esp32 Group Chat
Set upped a group chat communication between 3 ESP32s to facilitate a self-healing network for internetless transfer of data over the ESP Mesh Protocol
https://github.com/VanshPanchal0308/Esp32_Grp_chat

Competitions

Functional RoadBot
Built a pothole detection and filling bot using functional programming for the E-Yantra Robotics Competition conducted by IIT Bombay. My team is selected as one of the four finalists in the world for our theme

Language Skill Set
- C/C++
- Python
- Elixir

References

- https://gs-niteesh.github.io
- https://wiki.freebsd.org/FlattenedDeviceTree#More_powerful_use_cases
- https://elinux.org/Device_Tree_What_It_Is
- https://octavosystems.com/app_notes/osd335x-design-tutorial/osd335x-lesson-2-minimal-linux-boot/linux-device-tree/
- https://octavosystems.com/app_notes/osd335x-design-tutorial/osd335x-lesson-2-minimal-linux-boot/linux-device-tree/