Санкт-Петербургский государственный университет

Кафедра Системного программирования

Группа 22.Б15-мм

Портирование ОСРВ Embox на netduino plus 2

Перевалов Ефим Сергеевич

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к.ф.-м.н., Д. В. Луцив

Консультант:
ген. директор, ООО “Ембокс”, А. В. Бондарев

Санкт-Петербург
2024

Оглавление

Введение

Встраиваемые системы получают всё большую популярность, в частности, за счёт своей способности автоматизировать как устройства, используемые в бытовых целях, так и более сложные оборудования в промышленности. Этот тренд также требует соответствующего развития программного обеспечения.

Embox [1] представляет из себя ОС реального времени, которая имеет поддержку на различных платформах. В силу его разнообразия, существуют некоторые проблемы и возможность для расширения функциональности, такая как доработка драйверов.

Необходимо изучить, как Embox взаимодействует с различными архитектурами, например, с ARM и x86.

Главным подходом является решение уже существующих проблем, связанных с поддержкой платформ и микроконтроллеров, и изучение внутренней работы драйверов самой системы на возможность ошибок, которые могут возникнуть в дальнейшем.

1. Постановка задачи

Целью работы является портирование ОСРВ Embox на платформу netduino plus 2. Для ее выполнения были выделены следующие задачи:

1. Добавление необходимых модулей;
2. Добавление поддержки STM32F405;
3. Портирование ОСРВ Embox на netduino plus 2;
4. Тестирование.

2. Обзор

2.1. Разработка драйверов

Разработка драйверов зависит от конкретной ОС и платформы. Это включает в себя знание API и архитектуры ОС. Необходимо понимание спецификаций и возможностей аппаратных устройств, с которыми драйвер взаимодействует. Это может быть работа с процессорами, шинами, периферийными устройствами, сетевыми картами и другими компонентами. Разработка драйверов требует управления ресурсами, такими как управление памятью, обработка прерываний, взаимодействие с устройствами ввода-вывода. Драйверы подвергаются интенсивному тестированию и требуют систематической отладки, поскольку близко взаимодействуют с железом. Не менее важна эффективная работа с аппаратурой, например знание аппаратных интерфейсов, протоколов передачи данных и спецификаций устройств.

2.2. Embox

Embox [1] представляет собой открытую операционную систему реального времени, которая не зависит от платформы. Одним из возможных применений могут быть встраиваемые системы. Embox написан на языке программирования C. Выбором послужил именно этот язык, так как имеет относительно низкий уровень и прямую работу с памятью. Архитектурно зависимый код вынесен в отдельные модули, которые могут быть организованы и включены с помощью системы Mybuild [3]. Этот подход позволяет добавлять в сборку только необходимую функциональность для решения конкретных задач, избегая затрат на излишние модули, что особенно важно в контексте микроконтроллеров, где ресурсы сильно ограничены.

В ОСРВ Embox уже существует поддержка некоторого количества архитектур, в число которых входят ARM, x-86, RISC-V и несколько других. Одним из представителей ARM архитектуры являются платы семейства STM32 [6]. Они представляют собой 32-битные микро-

контроллеры использующие одинаковое ядро в рамках одной серии.

2.3. Netduino

На основе STM32F205RF существует платформа netduino 2 [4]. На данной плате доступно 22 цифровых входов/выходов, 4 из которых поддерживают UART [7], необходимый для обеспечения связи с устройствами. Помимо этого поддерживаются SPI [5], обеспечивающий связь с периферией, и I2C [2] для внутреннего взаимодействия. Объём оперативной памяти составляет 60 КБ, а для хранения исполняемого кода может быть использовано 192 Кб.

Netduino plus 2 схож с netduino 2, но использует STM32F405RG, и имеет разъём RJ45, который используется для Ethernet подключения. Помимо этого может читать карты MicroSD так как обладает кардридером. Объем оперативной памяти составляет свыше 100 КБ, а для исполняемого кода может быть использовано 384 Кб.

В свою очередь, STM32F405 - это производительный 32-битный микроконтроллер, основанный на базе ядра ARM Cortex-M4, благодаря чему может поддерживать операции с плавающей точкой. Может быть использован как для медицинских, так и для промышленных или потребительских целей.

3. Реализация

3.1. Поддержка платформы

Для реализации портирования ОСРВ Embox на netduinoplus2 было необходимо добавить поддержку платформы, а именно:

1. Модуль, который нужен для сборки самого Embox (arch). В нём находятся необходимые настройки для данной платы. Для этого потребовалось дополнительно изучить репозиторий STM32F4-Discovery [13] и использовать оттуда функцию SystemClock_Config, отвечающую за настройку системного времени.

2. Настройки для сборки, флагов, необходимых связей с другими частями программы и нужных для платы модулей (Mybuild).

3. Заголовочный файл (stm32f4xx_hal_conf), необходимый для сборки, который можно найти в репозитории STM32F4-Discovery [13]. Необходимо добавить только его, так как в Embox уже реализована поддержка STM32Cube, упрощающего работу с драйверами.

4. Вспомогательный файл stm32cube_compat, потребовавшийся для правильной работы STM32Cube.

3.2. Описание базового темплейта

После добавления новой платфоры стало необходимым добавление базового темплейта, так как на данном этапе отсутствует периферия и для проверки работоспособности нужна дополнительная функциональность. Для этого имеющаяся функциональность была дополнена следующими компонентами:

1. Описание для карты памяти (Ids.conf). Содержит следующие конфигурации:

ROM (0x08000000, 1024K) — определяет область флэш-памяти с начальным адресом 0x08000000 и размером 1024 Кб.

RAM (0x20000000, 128K) — оперативная память, имеющая начальный адрес 0x20000000 и размер 128 Кб.

text (ROM) — секция, отвечающая за машинные инструкции, размещенная в области ROM.

rodata (ROM) — представляет изменяемые данные, находящиеся во флэш-памяти.

data (RAM, ROM) — размещена как в RAM, так и в ROM. Отвечает за глобальные и статические данные.

bss (RAM) — отвечает за глобальные и статические данные, которые при старте содержат нулевые значения и размещена в RAM.

2. Описание, необходимое для компилятора, и флаги сборки. В нем указаны архитектура для запуска, платформа и какие-то дополнительные данные. (build.conf)

3. Описание самой системы, такое как системная частота и контроллер прерываний. (mods.conf)

3.3. Периферия

Следующим этапом идёт добавление описания периферии. Необходимо, чтобы Embox поддерживал порты, имеющиеся у платы netduino plus 2, и правильно работал с драйверами. Описание периферии находится в netduinoplus2.conf.h и содержит в себе:

1. UART - на его основе реализован базовый ввод - вывод. Для поддержки были определены следующие настройки:

   - Назначение номера прерывания, помогающего процессору определить, какой обработчик прерывания должен быть вызван.
   - Подключение портов для передачи(TX) и приёма(RX) данных.

- Установки тактовой частоты.
- Установка скорости передачи данных.

```
.dev = {
        .name = "USART1",
        .irqs = {
                VAL("", 37),
        },
        .pins = {
                PIN("TX", PB, PIN_6, AF7),
                PIN("RX", PB, PIN_7, AF7),
        },
        .clocks = {
                VAL("TX",   CLK_GPIOB),
                VAL("RX",   CLK_GPIOB),
                VAL("UART", CLK_USART1),
        }
},
.baudrate = 115200,
```

2. Также нужно обеспечить поддержку SPI.

- Назначение номера прерывания.
- Установка порта, отвечающего за общий частотный сигнал(SCK).
- Подключение порта приёма данных ведущим (MISO).
- Пин отправки данных от ведомого к ведущему (MOSI).
- Чип селект, использующийся для определения ведущего устройства (CS).
- Указание тактовой частоты.

```
.dev = {
        .name = "SPI1",
        .pins = {
                PIN("SCK", GPIO_PORT_A, PIN_5, AF5),
                PIN("MISO", GPIO_PORT_A, PIN_6, AF5),
                PIN("MOSI", GPIO_PORT_A, PIN_7, AF5),
                PIN("CS",   GPIO_PORT_C, PIN_13, NOAF),
        },
        .clocks = {
                VAL("SCK",  CLK_GPIOA),
                VAL("MISO", CLK_GPIOA),
                VAL("MOSI", CLK_GPIOA),
                VAL("CS",   CLK_GPIC),
                VAL("SPI",  CLK_SPI1),
        }
},
```

3. I2C.

   - Назначение номера прерывания для ошибок и событий.
   - Подключение порта частотного сигнала (SCL).
   - Подключение порта для работы с данными (SDA).
   - Указание тактовой частоты.

```
.dev = {
        .name = "I2C1",
        .irqs = {
                VAL("EVENT", 31),
                VAL("ERROR", 32),
        },
        .pins = {
                PIN("SCL", GPIO_PORT_B, PIN_8, AF4),
                PIN("SDA", GPIO_PORT_B, PIN_9, AF4),
        },
        .clocks = {
                VAL("I2C", CLK_I2C1),
        }
},
```

Для ознакомления с портами для подключения использовалась документация [9] для данной платформы.

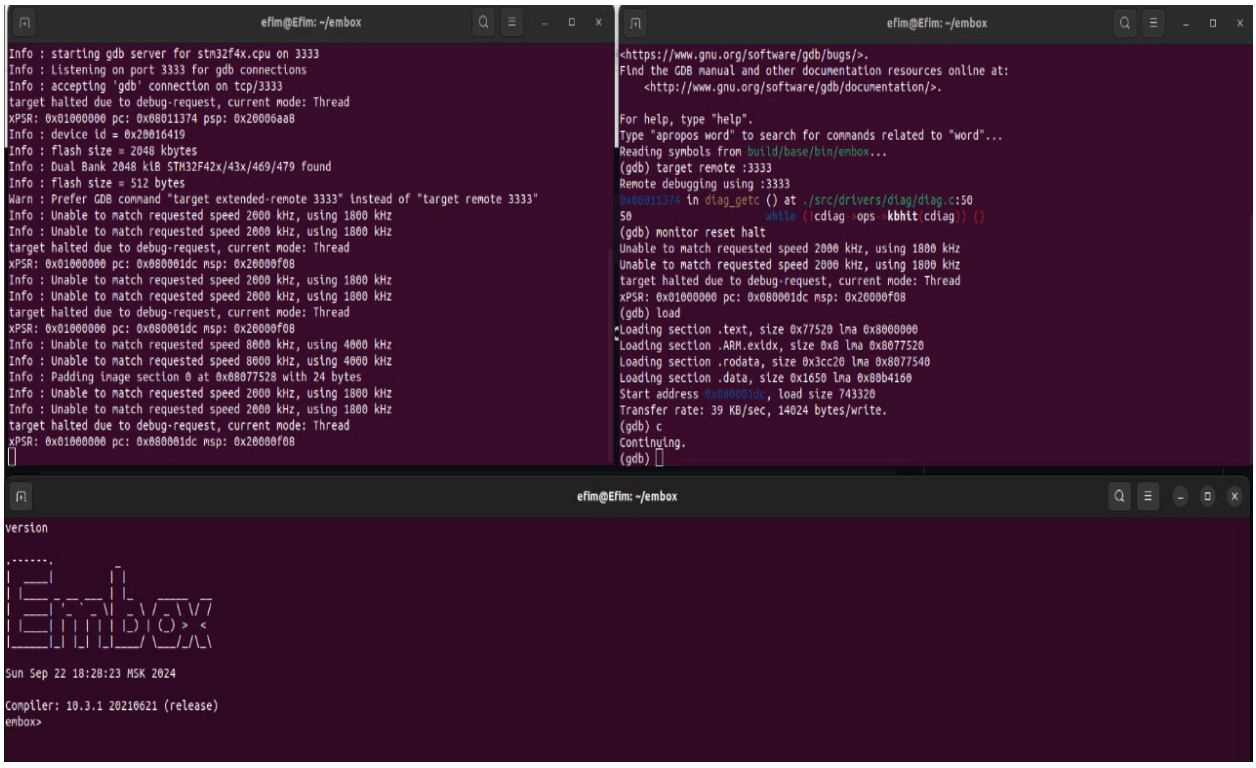
3.4. Настройка системной частоты

Помимо этого в процессе работы произошло столкновение с ошибкой, которая связана с настройкой системной частоты. Для её нахождения потребовалось взаимодействие с отладчиком GNU Debugger, широко используемом для программ, написанных на языке C. Для выявления изменения в корректности работы в качестве примера использовалась STM32F429ZI. Для ознакомления с подключением к плате её необходимо было прошить, и проверить работоспособность. Это производилось с использованием нескольких системных утилит:

1. OpenOCD, предоставляющая возможность отлаживать и прошивать различные микроконтроллеры;

2. Minicom, который даёт возможность взаимодействовать с подключёнными устройствами и представляющий собой терминальный эмулятор.

3. Gdb, благодаря которому можно было отследить каждый шаг загрузки Embox и найти различия в работе с netduino plus 2.



Таким образом были выявлены проблемы с регистрами RCC.

3.5. Ошибки с другими платами

В процессе работы также были найдены и исправлены следующие ошибки:

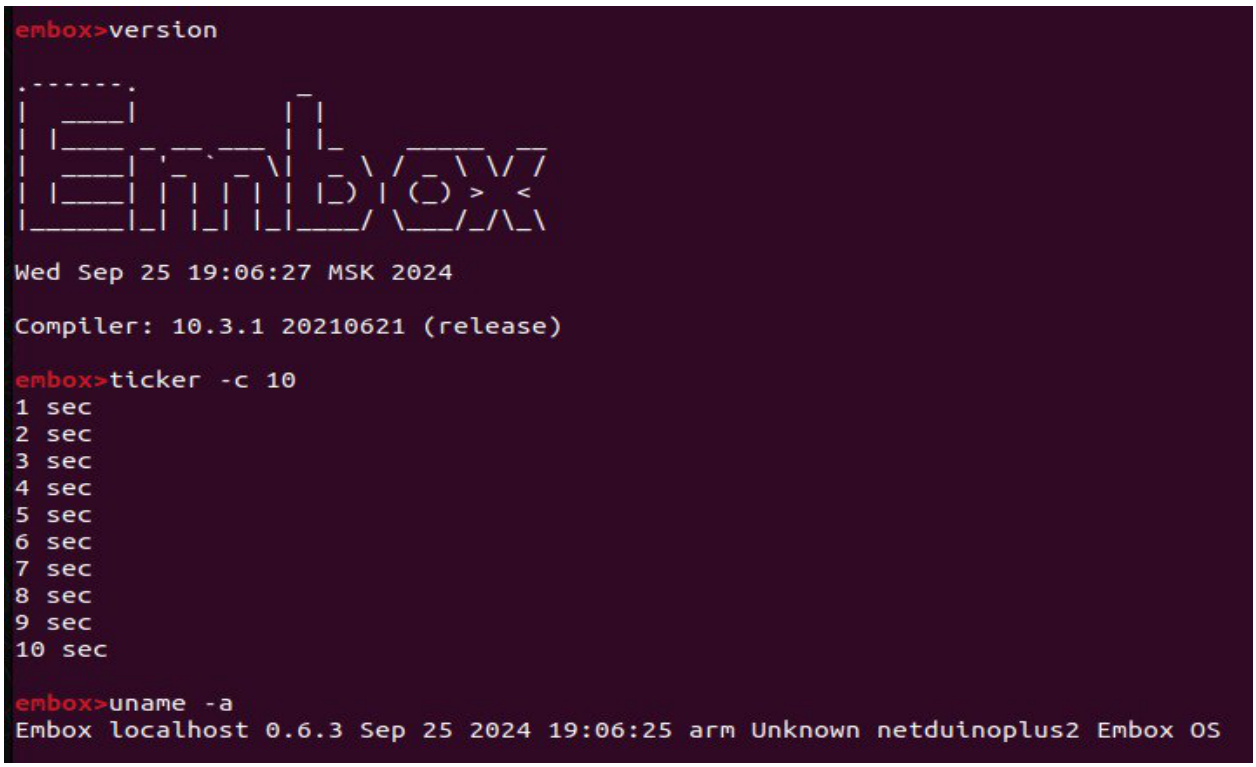
1. В nucleo_f207zg.conf.h USART1 подключение происходило по порту PB9, хотя должен был использоваться PA9.
2. В nucleo_f207zg.conf.h I2C2 указывался CLK_I2C1 вместо CLK_I2C2.
3. В stm32f4discovery.conf.h I2C2 использовался CLK_I2C1, хотя должен был CLK_I2C2.

Все вышеперечисленные ошибки, в следствии неправильной обработки, могли привести к отсутствию отклика операционной системы на используемые порты. Для проверки корректности портов использовалась документация для STM32F4DISCOVERY [10] и STM32F207ZG [8]

Работа велась в репозитории [12], основной репозиторий [11] .

4. Апробация

В качестве апробации была произведена сборка и запуск ОСРВ Embox на портированной платформе netduino plus 2. Для того, чтобы проверить корректность работы в условиях, максимально приближенных к реальным, без необходимости непосредственного доступа к физическому устройству был использован эмулятор QEMU.



Можно пронаблюдать, что запуск команд через интерфейс UART подтверждает, что все основные компоненты функционируют корректно. Также видно, что была произведена проверка работы таймера и системы прерываний. Для этого использовалась команда ticker, которая позволяет генерировать прерывания с заданной частотой. Данная проверка подтверждает, что таймер функционирует стабильно, а система прерываний реагирует на события, как и ожидалось.

Заключение

Результатом практики весеннего семестра стало:

- Были добавлены модули, необходимые для netduino plus 2;
- Добавлена поддержка STM32F405;
- Портирование ОСРВ Embox на netduino plus 2;
- Произведена сборка и запуск портированной системы;
- Исправлены небольшие баги в STM32F207ZG и STM32F4DISCOVERY.

Список литературы

[1] Embox. — URL: https://ru.wikipedia.org/wiki/Embox (дата обращения: 21 сентября 2024 г.).

[2] I2C. — URL: https://en.wikipedia.org/wiki/I2C (дата обращения: 21 сентября 2024 г.).

[3] Mybuild. — URL: https://non-descriptive.github.io/embox-mdbook/ru/embox_user_manual_ru.html (дата обращения: 21 сентября 2024 г.).

[4] Netduino. — URL: https://ru.wikipedia.org/wiki/Netduino (дата обращения: 21 сентября 2024 г.).

[5] SPI. — URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface (дата обращения: 21 сентября 2024 г.).

[6] STM32. — URL: https://ru.wikipedia.org/wiki/STM32 (дата обращения: 21 сентября 2024 г.).

[7] UART. — URL: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter (дата обращения: 21 сентября 2024 г.).

[8] Документация STM32F207ZG. — URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f207zg.html (дата обращения: 21 сентября 2024 г.).

[9] Документация STM32F405xx. — URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f407vg.html#documentation (дата обращения: 21 сентября 2024 г.).

[10] Документация STM32F4DISCOVERY. — URL: https://www.st.com/en/evaluation-tools/stm32f4discovery.html#documentation (дата обращения: 21 сентября 2024 г.).

[11] Основной репозиторий. — URL: https://github.com/embox/embox (дата обращения: 21 сентября 2024 г.).

[12] Рабочий репозиторий. — URL: https://github.com/Urtix/embox (дата обращения: 21 сентября 2024 г.).

[13] Репозиторий STM32F4-Discovery. — URL: https://github.com/STMicroelectronics/STM32CubeF4/tree/master/Projects/STM32F4-Discovery (дата обращения: 21 сентября 2024 г.).