Операционная система FreeRTOS

Курсовая работа

Евгений Зеленин

2 мая 2025 г.

1. Постановка задачи

Условия задачи.

В качестве курсового проекта предлагается сделать несложный логический контроллер. Контроллер имеет 4 входа (кнопки) и 4 выхода (светодиоды). Помимо входов и выходов контроллер должен иметь UART для приема команд. Реализуйте простейшие команды управляющие логикой контроллера. Пример команд: 1. Опрос состояние входов - In? 2. Состояние выходов - Out 1 on 3. Маска поведения выхода в зависимости от состояния входа - Mask XXXX В Качестве результата работы предоставьте видео с демонстрацией и исходные коды проекта.

2. Построение логики работы и архитектуры ПО

Для выполнения задания будем использовать отладочную плату MCUINSIDE v1. В целях ознакомления с возможностями FreeRTOS внесем некоторые изменения в задание. Для управления ПЛК будем использовать Virtual COM Port и настроим МК на режим работы USB CDC. Также, предусмотрим возможность сохранения и загрузки настроек из внешней EEPROM по протоколу I_2C (микросхема памяти AT24C32). Назначение выводов МК показано на рисунке 1.

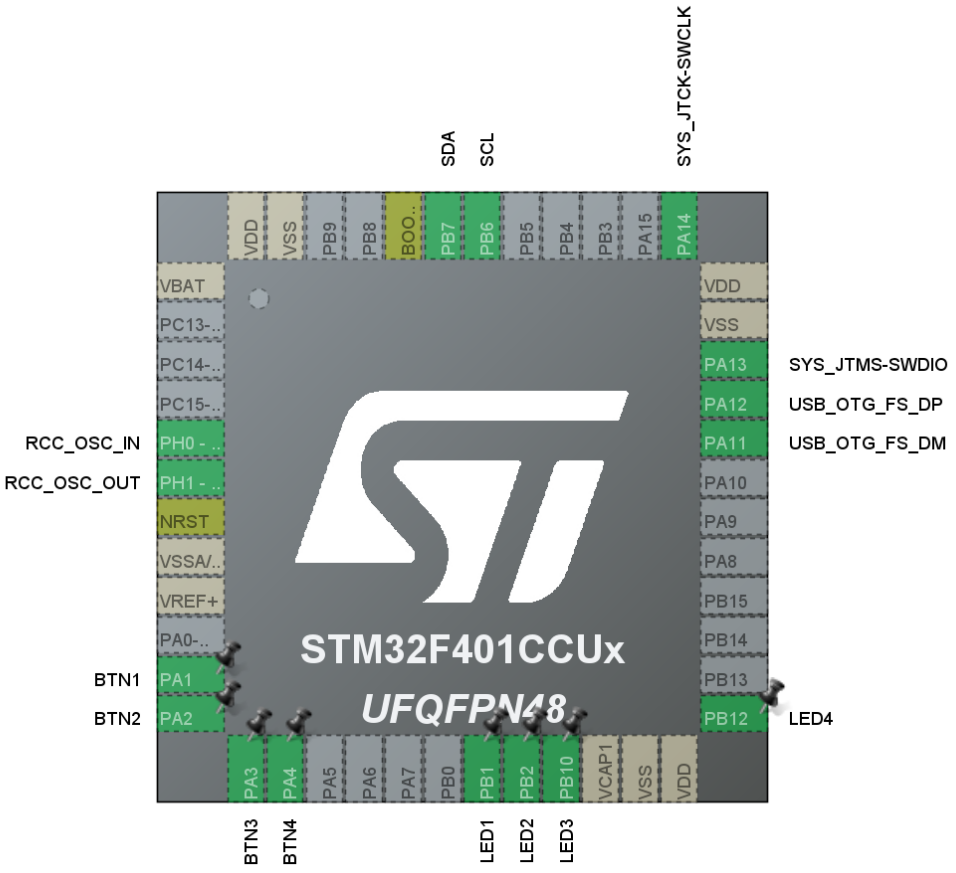

Рис. 1: Назначение выводов в CubeMX

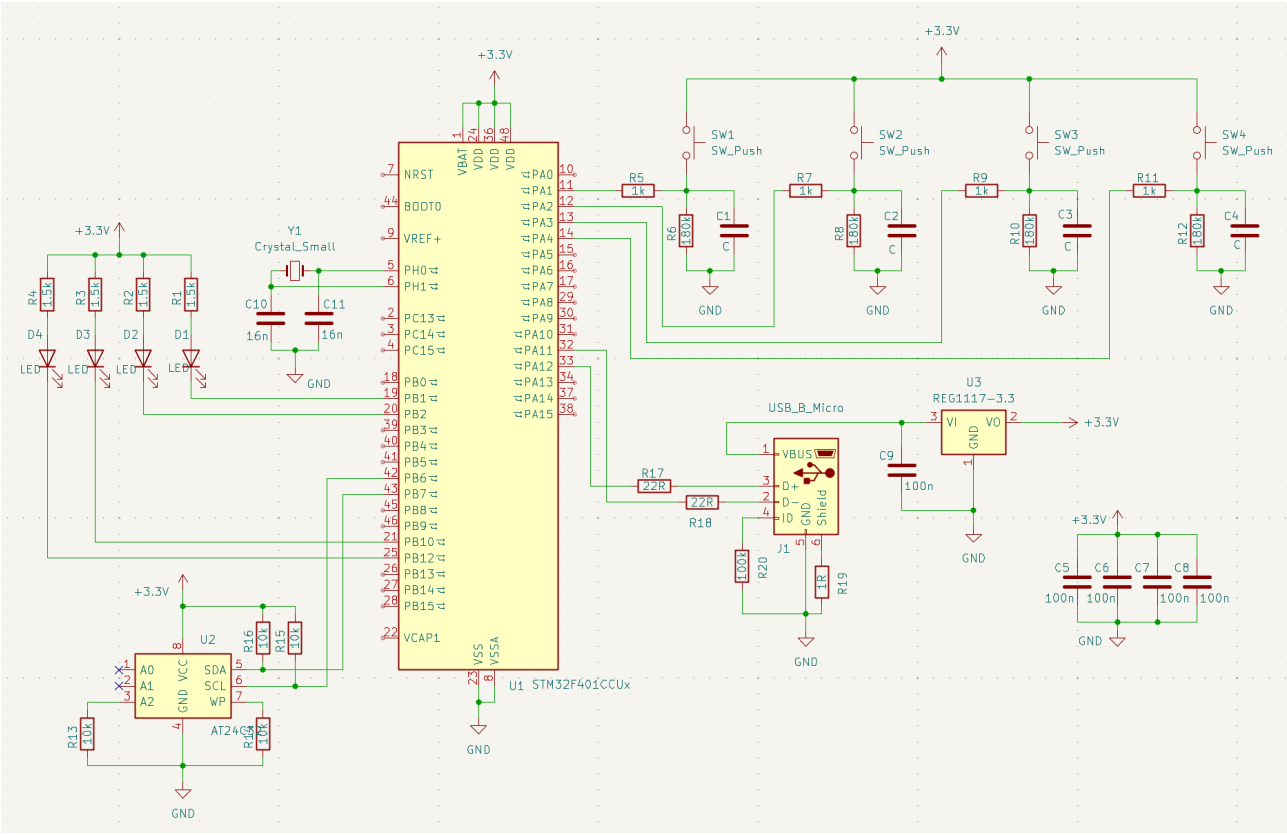

Рис. 2: Схема подключений периферии

Для выполнения такого количества функций требуется использовать достаточно большое количество задач. Поэтому, главной задачей, отвечающей за управление остальными (точкой синхронизации) является функция StartSync(). Она отвечает за синхронизацию потоков данных между остальными задачами. Это достигается с помощью разнообразных механизмов: используются очереди, семафоры, нотификация, события, мьютексы. Подробная схема взаимодействия между модулями программы показана на рисунке 3.

Обработка нажатий кнопок

Каждой кнопке соответствует отдельная задача, которая в момент создания получает номер кнопки через параметр. Функции StartButtons() отвечают за установку/снятия бита соответствующего кнопке в группе событий. Одновременно может быть установлено несколько бит, относящихся к разным кнопкам. Когда происходит событие нажатия/отпускания кнопки, устанавливаются или снимаются биты в группе событий и передается уведомление задаче синхронизации StartSync(). В этой задаче сравнивается текущее состояние группы событий с установленной маской, после чего принимается решение о том какой светодиод требуется зажечь или погасить. Когда решение принято, соответствующей задаче-обработчику светодиода отправляется нотификация с кодом команды (зажечь или погасить светодиод).

Задачи, отвечающие за установку выходов в нужное состоянии также помещают/стирают соответствующее значение в группе событий по выходам, это дает возможность удобно считать статусы выходов и входов по команде из терминала.

Все задачи, отвечающие за обработку нажатий создаются из одной и той же функции. Аналогичным образом реализованы задачи для управления светодиодами.

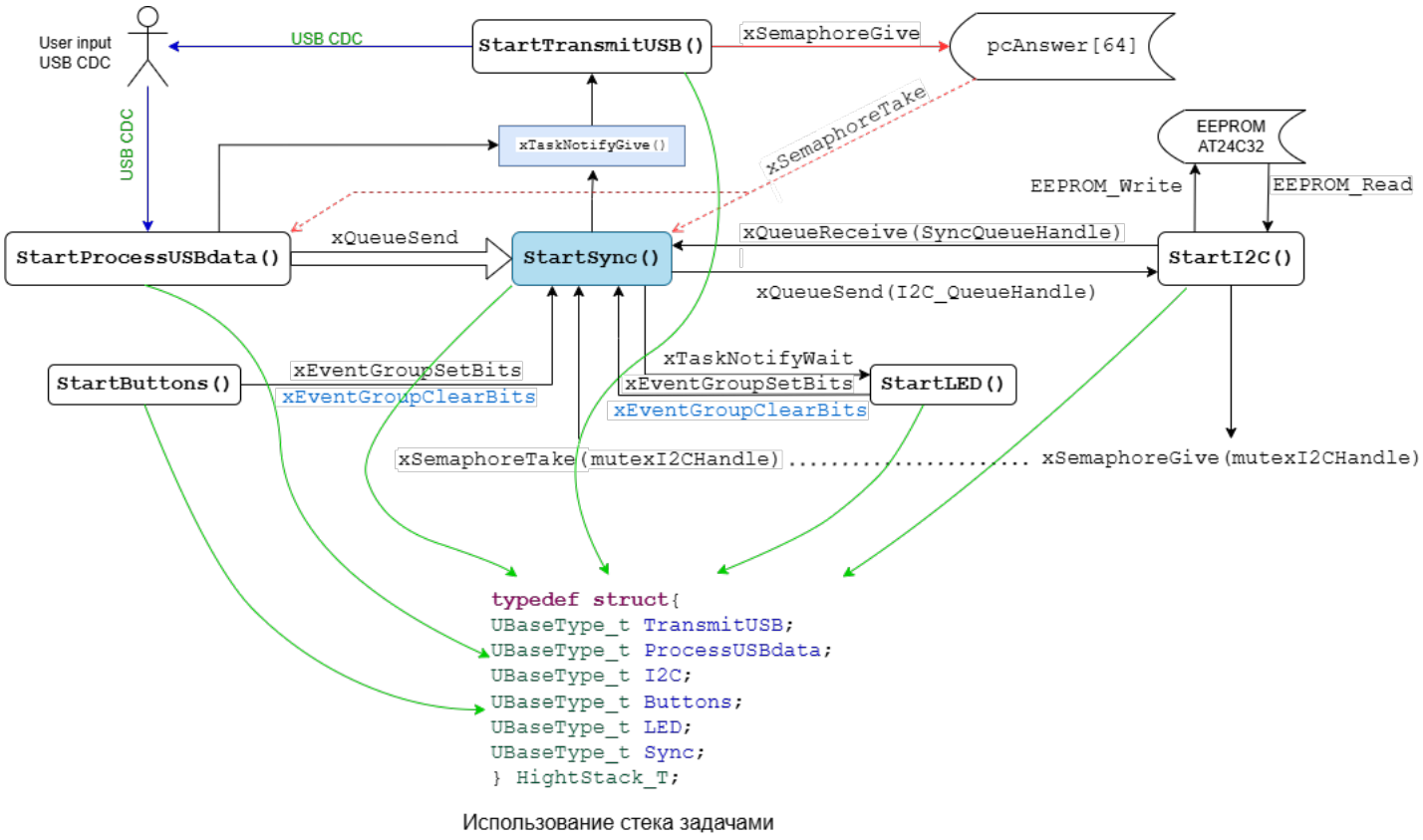

Рис. 3: Логика взаимодействия задач

Прием команд через Virtual COM Port

Взаимодействие с пользователем осуществляется через USB, через Virtual COM Port. Для управления ПЛК используется следующий набор команд:

- Save - Сохранить конфигурацию в EEPROM
- Load - Загрузить конфигурацию из EEPROM
- Show - Показать текущие настройки
- Reset - Перезагрузить ПЛК
- Status - Показать статус входов и выходов

В программе используется отложенная обработка данных, полученных через USB: в функции CDC_Receive_FS() устанавливается семафор и вызывается макрос portYIELD_FROM_ISR(). После чего обработка буфера осуществляется в задаче StartProcessUSBdata().

В функции StartProcessUSBdata() осуществляется разбор команд их последующая передача через очередь задаче StartSync(), а так же заполнение буфера сообщениями для ответа через Virtual COM Port.

Примеры работы с системой команд показаны на рисунке 4.

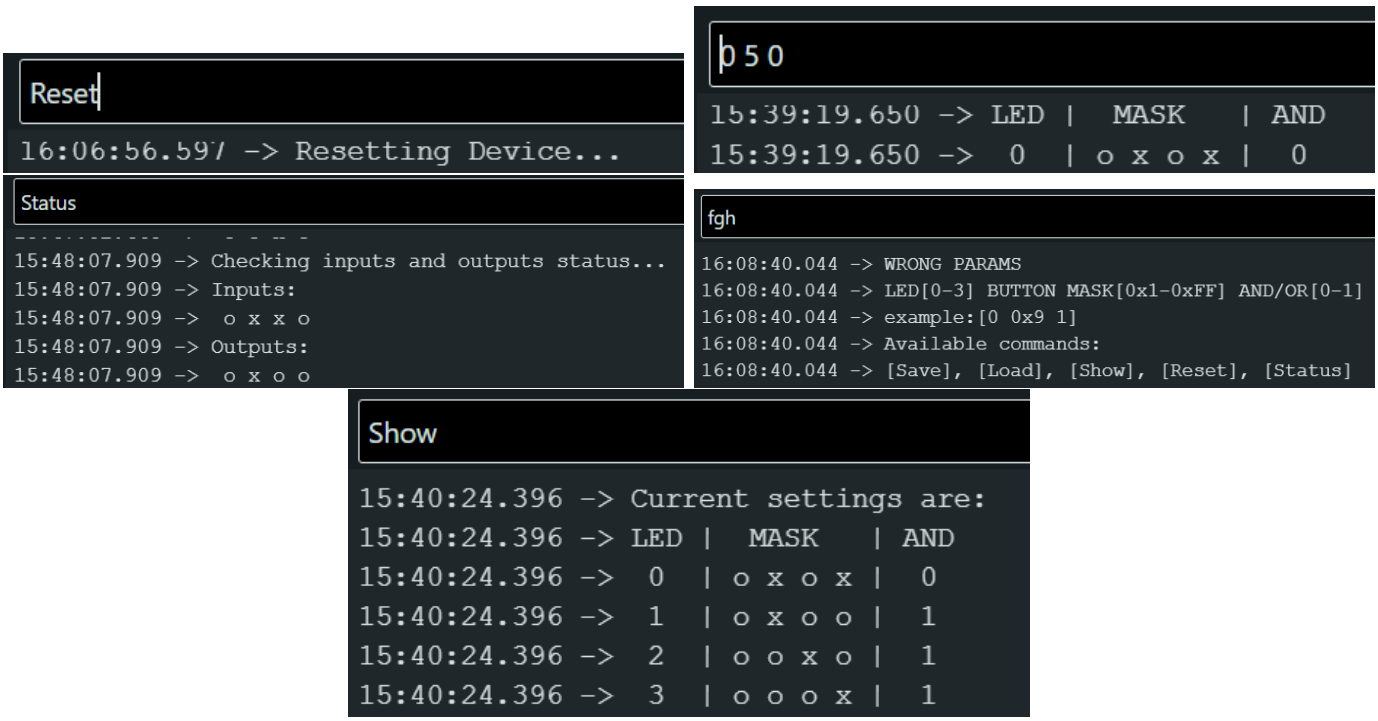

Рис. 4: Примеры взаимодействия с системой команд

Отправка сообщений через Virtual COM Port

Отправка сообщения на ПК осуществляется в задаче StartTransmitUSB(). Чтобы исключить наслоение данных, доступ к буферу блокируется с помощью семафора, а разрешение на передачу данных передается задаче через нотификацию после заполнения буфера. После отправки данных, семафор возвращается задачей StartTransmitUSB.

Таким образом, перед отправкой сообщения, задаче-отправителю требуется взять семафор, заполнить буфер и отправить нотификацию задаче StartTransmitUSB(). После отправки сообщения, семафор возвращается и у других задач появляется доступ к буферу для отправки сообщений.

Операции записи/чтения EEPROM

За операции чтения/записи EEPROM отвечает задача StartI2C(). Управляющие команды и данные передаются через очередь с помощью структуры вида:

```
/*Data structure to send via Queue*/
typedef struct {
  uint32_t *data; //Data pointer
  uint8_t size;   //Data size
  uint8_t cmd;    //Command
} QDATA_T;
```

В этой структуре передается указатель на данные (указатель на массив структур или на одиночное значение), размер данных и команда (чтение/запись).

Первым значением в EEPROM записывается количество байт, а далее идут сами данные. Аналогично, во время операции чтения сначала считывается размер данных, а потом сами данные.

Анализ стека

Для анализа использования стека сформирована структура следующего вида:

```
typedef struct{
  UBaseType_t TransmitUSB;
  UBaseType_t ProcessUSBdata;
  UBaseType_t I2C;
  UBaseType_t Buttons;
  UBaseType_t LED;
  UBaseType_t Sync;
  } HightStack_T;
```

Далее, в каждой задаче в процессе выполнения заполняются значения этой структуры с помощью функции uxTaskGetStackHighWaterMark().

```
xUserStack.XXXXX = uxTaskGetStackHighWaterMark(NULL);
```

Этот прием позволяет в процессе отладки наблюдать за использованием стека.

3. Заключение

В ходе выполнения курсового проекта были использованы основные приемы работы с FreeRTOS: очереди, семафоры, мьютексы, события, нотификаиция, отложенная обработка прерываний. Задача по созданию ПЛК выполнена успешно, весь заявленный функционал реализован в полном объеме.

Следует отметить, что FreeRTOS в разы ускоряет и упрощает разработку ПО со сложной архитектурой. FreeRTOS позволяет гибко настраивать процессы взаимодействия различных модулей программы и синхронизировать потоки данных.

4. Дополнительные материалы

Видео-демонстрация, схема, исходные коды и другие материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к курсовому по FreeRTOS