СБОРКА FREERTOS V9.0.0
ДЛЯ ОТЛАДОЧНОГО МОДУЛЯ САЛЮТ «ЭЛ24Д1»

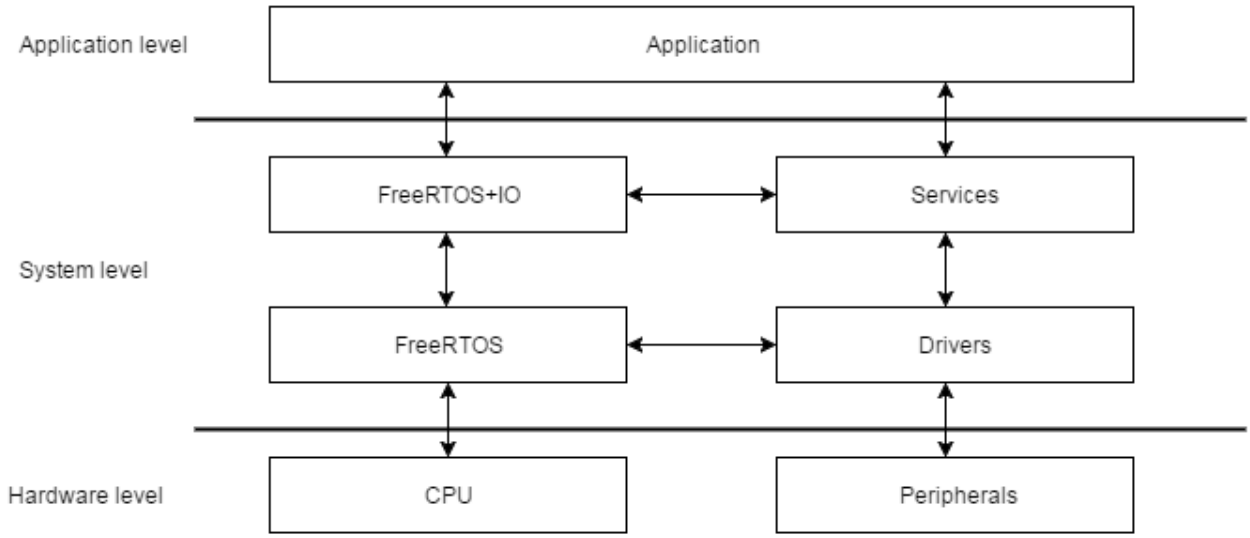
FreeRTOS. Appnote

ОГЛАВЛЕНИЕ

1. ОПИСАНИЕ

ОСРВ FreeRTOS для процессора 1892ВМ14Я основана на исходных кодах FreeRTOS V9.0.0 rc1, содержит изменения на уровне ядра (см. главу Исправления в ядре Freertos). ОСРВ FreeRTOS запускается на отладочных модулях «Салют ЭЛ24Д1 rev 1.3 – rev 1.5». В состав сборки входят драйверы процессора 1892ВМ14Я (см.п. Драйверы), сервисы (см. п. Сервисы), стек TCP/IP, HTTP и FTP сервера. На Рисунок 1 "Иерархия компонентов FreeRTOS" представлена иерархия взаимодействия компонентов в сборке FreeRTOS. Компонент FreeRTOS+IO включает: севера HTTP, FTP, файловую систему FAT, сервер CLI.

Рисунок 1 "Иерархия компонентов FreeRTOS"



В сборке FreeRTOS принято назначать приоритеты задач, в соответствии с следующий таблицей. Приоритет задач, возрастающий (0 – самый низкий).

Таблица 1.1 Уровни приоритетов задач

Приоритет задач	Уровень приоритета задачи в системе
0 - 2	Пользовательские задачи
3 - 5	Службы
>= 6	Сервера (FreeRTOS + IO)

2. АППАРАТУРА

Для запуска и тестирования ОСРВ FreeRTOS используется отладочный модуль «Салют ЭЛ24Д1». Отладочный модуль выполнен на базе процессора 1892ВМ14Я. Отладочная информация выводится с модуля в консоль через переходник USB-UART или COM-UART, переходник подключен к выводам UART0 отладочного модуля. Схема подключения переходника USB-UART к отладочному модулю Салют ЭЛ24Д1 представлена в п.8 документа «Модуль отладочный Салют-ЭЛ24Д1. Руководство пользователя». Хранилищем данных FTP сервера выступают SD карта и/или RAM диск.

Сборка предназначена для загрузки системы с SD карты (процесс описан в документе «FreeRTOS. Руководство системного программиста.pdf») и для запуска с использованием отладчика gdb из командной строки или с использованием среды Netbeans для ОС Linux CentOS7 (https://netbeans.org/downloads/index.html). Для загрузки elf файла сборки используется эмулятор «MC-USB-JTAG».

3. ОПИСАНИЕ СТРУКТУРЫ

Таблица 3.1 Структура проекта

Папка/Файл	Назначение
/docs	Документация по сборке FreeRTOS
/drivers	Драйвера
/freertos	Ядро FreeRTOS v9.0.0, дополнительный функционал FreeRTOS
/nbproject	Конфигурация среды NetBeans
/scripts	Скрипты прошивки SD, SPI, вспомогательные файлы
/services	Сервисы операционной системы
/tasks	Пользовательские задачи
main.c	Файл с точкой входа в main(), основная программа
irq_handlers.c	Обработчики системных событий
platform.c	Инициализация платформы
reset.s	startup файл
mcom2_reg.h	Описание регистров процессора
.gdbinit	Файл GDB
.gitignore	Файл исключений GIT
CMakeLists.txt	Файл CMake с правилами управления проектом
CMakeToolchain.cmake	Файл CMake настроек тулчейна
start	Скрип создания инфраструктуры CMake

4. ИСПРАВЛЕНИЯ В ЯДРЕ FREERTOS

Файл freertos/source/tasks.c:

В структуру tskTaskControlBlock добавлено:

```
#if (configUSE_USER_MODE_SUPPORT == 1)
volatile uint32_t ulTaskASID;
#endif
```

В функцию xTaskGenericCreate() добавлено:

- Добавлен аргумент функции unsigned int ulTaskASID
- проверка: if(!ulTaskASID) configASSERT(pxTaskCode);
- #if (configUSE_USER_MODE_SUPPORT == 0) configASSERT(ulTaskASID == 0); #endif
- #if (configUSE_USER_MODE_SUPPORT == 1) pxNewTCB->ulTaskASID = ulTaskASID; #endif

В функции vTaskStartScheduler() изменен вызов xTaskGenericCreate(), добавлен аргумент.

Файл freertos/source/task.h:

- Добавлен макрос:

```
#define xUserModeTaskCreate( ulTaskASID, pvTaskCode, pcName,
usStackDepth, pvParameters, uxPriority, pxCreatedTask )
xTaskGenericCreate( ulTaskASID, ( pvTaskCode ), ( pcName ), (
usStackDepth ), ( pvParameters ), ( uxPriority ), ( pxCreatedTask ), (
NULL ), ( NULL ), ( NULL ) )
```

Файл freertos/source/timers.c:

- Добавлен параметр к вызову функции xTaskGenericCreate().

Файл freertos/source/FreeRTOS_TCP/FreeRTOS_ARP.c:

- Замена второго аргумента функции vARPRefreshCacheEntry() на &pxARPHeader->ulSenderProtocolAddress

5. ДРАЙВЕРЫ

5.1 Драйвер dbgled

Таблица 5.1 Открытое API драйвера dbgled

Функция	Назначение
void setDBGLED(int num);	Зажечь светодиод
void unsetDBGLED(int num);	Погасить светодиод

5.2 Драйвер emac

Таблица 5.2 Открытое API драйвера emac

Функция	Назначение
int emac_init(gemac_speed_t speed, gemac_mode_t mode, const unsigned char* mac_addr, int bFlowControl);	Инициализировать контроллер Ethernet.
void emac_work_start();	Запустить контроллер Ethernet
int emac_dev_write(int create_header, int use_shadow_buffer, const unsigned char* addr, unsigned char* pData, int size);	Записать данные в передатчик Ethernet

5.3 Драйвер gic

Таблица 5.3 Открытое API драйвера gic

Функция	Назначение
void enable_IRQ(void);	Разрешить системные прерывания
void disable_IRQ(void);	Запретить системные прерывания
void enable_FIQ(void);	Разрешить быстрые прерывания
void disable_FIQ(void);	Запретить быстрые прерывания
void attach_irq_handler(unsigned int ID, handler_funct_ptr funct_pointer, unsigned int core);	Зарегистрировать обработчик прерывания
void attach_exception_handler(enum EXCEPTIONS ID, handler_funct_ptr funct_pointer, unsigned int core);	Зарегистрировать обработчик исключения

5.4 Драйвер l2cc

Таблица 5.4 Открытое API драйвера l2cc

Функция	Назначение
void init_l2cc();	Инициализировать кэш L2
void sync_l2cc();	Синхронизация барьера инструкций и данных кэша и pipeline процессора
void flush_and_invalidate_cache_l2cc();	Сбросить и инвалидировать кэш
void flush_cache_l2cc();	Сбросить кэш
void invalidate_cache_l2cc();	Инвалидировать кэш

5.5 Драйвер mmu

Таблица 5.5 Открытое API драйвера mmu

Функция	Назначение
mmu_op_status_t mmu_set_context(mmu_ctx_t *ctx, void *ttb1_16kb);	Сохранение контекста
mmu_ctx_t * mmu_get_ctx();	Восстановление контекста
mmu_op_status_t mmu_set_global_1mb_region(unsigned int va, unsigned int pa, int mb_count, int policy, int access, int non_execute);	Установить глобальный регион памяти с настройками policy
mmu_op_status_t mmu_set_global_1mb_region_pagetable_64k(unsigned int va, unsigned int *pte);	Установить page table entry
mmu_op_status_t mmu_set_global_1mb_region_pagetable_4k(unsigned int va, unsigned int *pte);	Установить page table entry
mmu_op_status_t mmu_set_global_page_64k(unsigned int va, unsigned int pa, int policy, int access, int non_execute);	Установить глобальный регион памяти с настройками policy
mmu_op_status_t mmu_set_global_page_64k_address(unsigned int va, unsigned int pa, int non_execute);	Заполнить page table entry

Функция	Назначение
mmu_op_status_t mmu_set_global_page_4k(unsigned int va, unsigned int pa, int policy, int access, int non_execute);	Установить глобальный регион памяти с настройками policy
mmu_op_status_t mmu_set_global_page_4k_address(unsigned int va, unsigned int pa, int non_execute);	Заполнить page table entry
mmu_op_status_t mmu_set_non_global_1mb_region(mmu_non_global_header_t* ng_header, int policy, int access, int non_execute);	Установить глобальный регион памяти с настройками policy

5.6 Драйвер pll

Таблица 5.6 Открытое API драйвера pll

Функция	Назначение
unsigned int getCPUVoltage(unsigned int MHz);	Получить рабочее напряжение процессора Cortex-A9
unsigned int getDSPVoltage(unsigned int MHz);	Получить рабочее напряжение процессора DSP-кластера DELcore 30M
unsigned int getCurrentCPUFreq();	Получить частоту ядра Cortex-A9
unsigned int getCurrentDSPFreq();	Получить частоту ядра DSP кластера DELcore 30M
unsigned int getCurrentSystemFreq();	Получить частоту системной шины
int setCPUFreq(unsigned int MHz);	Установить частоту ядра Cortex-A9
int setDSPFreq(unsigned int MHz);	Установить частоту ядра DSP-кластера DELcore 30M
int setSystemFreq(unsigned int MHz);	Установить частоту системной шины

5.7 Драйвер ptimers

Таблица 5.7 Открытое API драйвера ptimers

Функция	Назначение
void initPrivateTimer(unsigned int Hz, int bAuto, int bInt);	Инициализация private таймера процессора

Функция	Назначение
unsigned int getPrivateTimerCounter();	Получить счетчик private таймера
void resetPrivateTimerInt();	Сбросить private таймер
void disablePrivateTimer();	Сбросить и остановить таймер
void initPrivateWDT(unsigned int ms, int bAuto, int bInt);	Инициализировать Watch Dog Timer
unsigned int getPrivateWDTCounter();	Получить счетчик Watch Dog Timer
void refreshPrivateWDT();	Сбросить Watch Dog Timer
void disablePrivateWDT();	Сбросить и остановить Watch Dog Timer
void initTimerModePrivateWDT(unsigned int Hz, int bAuto, int bInt);	Установить режим работы Watch Dog Timer
void resetTimerModePrivateWDTIntr();	Сбросить режим работы Watch Dog Timer

5.8 Драйвер scu

Таблица 5.8 Открытое API драйвера scu

Функция	Назначение
void init_scu(void);	Инициализировать и включить scu

5.9 Драйвер sd

Таблица 5.9 Открытое API драйвера sd

Функция	Назначение
SDError_t xSDCardInit(sd_card_t *sd);	Инициализация контроллера SDMMC и SD карты, контроллер и SD карта настраивается на работу в режиме DEFAULT_SPEED
SDError_t xSDWrite(sd_card_t *sd, uint32_t startBlock, block_t *pData, uint32_t numBlocks);	Запись на SD карту
SDError_t xSDRead(sd_card_t *sd, uint32_t startBlock, block_t *pData, uint32_t numBlocks);	Чтение с SD карты

5.10 Драйвер SPI

Таблица 5.10 Открытое API драйвера spi flash

Функция	Назначение
void ConfigSPI0();	Инициализировать SPI0 контроллер, контроллер работает в режиме: CPOL = 0, CPHA = 0, частота 36 МГц.
void DisableSPI0();	Отключить тактирование контроллера SPI0, вернуть в состояние после сброса
unsigned char ReadStatusRegisterSPI0Flash();	Прочитать регистр статуса SPI Flash
void SendOpcodeSPI0(unsigned char opcode);	Отправить код операции SPI Flash. Возможные коды: OP_READ 0x3 OP_WREN 0x6 OP_WRDIS 0x4 OP_PAGEPROG 0x2 OP_READSTATUS 0x5 OP_BLOCKERASE 0xd8 OP_CHIPERASE 0xc7
void BlockEraseSPI0Flash(boot_config_t* cfg);	Стереть страницу SPI Flash
void WriteSPI0Flash(unsigned int address, unsigned char* bytes, unsigned int size, boot_config_t* cfg);	Записать на SPI Flash
int ReadSPI0Flash(unsigned int address, unsigned char* bytes, unsigned int size, boot_config_t * cfg);	Прочитать с SPI Flash

5.11 Драйвер UART

Таблица 5.11 Открытое API драйвера uart

Функция	Назначение
void uart_config(unsigned int uartNum, uart_baudrate_t baud, uart_data_len_t bits, uart_stop_bit stopBit, uart_parity_t parity);	Инициализировать UART
void uart_disable(unsigned int uartNum);	Отключить тактирование контроллера UART, вернуть в состояние после сброса
int uart_init_interrupt(unsigned int uartNum, unsigned int rftl);	Включить прерывания UART
void uart_interrupt_handler(int ulICCIAR, void(*callback)(int, int, int));	Зарегистрировать обработчик прерывания UART

Функция	Назначение
int uart_send_data(unsigned int uartNum, const char* src, int size);	Отправить данные в UART
int uart_receive_data(unsigned int uartNum, void* dst, int size, int linesplit);	Получить данные из UART
int is_tx_empty(unsigned int uartNum);	Проверка отсутствия данных в буфере передачи
int is_rx_empty(unsigned int uartNum);	Проверка отсутствия данных в буфере приема

6. СЕРВИСЫ

6.1 Сервис netsrv

Сервис реализует запуск FTP и HTTP серверов и UDP сервиса

Таблица 6.1 API сервиса netsrv

Функция	Назначение
void vNetsrvStartTCPServer(void);	Запуск FTP и HTTP серверов
void vNetsrvStopTCPServer(void);	Остановка FTP и HTTP серверов
BaseType_t vStartUDPService(void);	Запуск UDP сервиса
void vStopUDPService(void);	Остановка UDP сервиса
int32_t xNetsrvUDPRecvfrom(void *pvBuffer, uint32_t xBufLen, struct freertos_sockaddr *pxSourceAddr);	Прочитать данные из внутреннего буфера UDP сервиса
int32_t xNetsrvUDPSendto(void *pvBuffer, uint32_t xBufLen, struct freertos_sockaddr *pxDestAddr);	Послать данные по UDP

6.2 Сервис sdsrv

Сервис реализует интерфейс чтения, записи и инициализации SD-карты (SDHC/SDXC). При запуске сервиса на SD-карте создается файловая система FAT32.

Таблица 6.2 API сервиса sdsrv

Функция	Назначение
int vInitSDService(uint8_t ucSDMMC);	Инициализация контроллера SDMMC0, инициализация SD карты, запуск сервиса
SDError_t xSDServiceWrite(sd_card_t *sd, uint32_t startBlock, Block_t *pData, uint32_t numBlocks);	Запись данных на SD карту
SDError_t xSDServiceRead(sd_card_t *sd, uint32_t startBlock, Block_t *pData, uint32_t numBlocks);	Чтение данных с SD карты
int vInitSDService(uint8_t ucSDMMC);	Инициализация контроллера SDMMC0, инициализация SD карты, запуск сервиса

Функция	Назначение
SDError_t xSDServiceWrite(sd_card_t *sd, uint32_t startBlock, Block_t *pData, uint32_t numBlocks);	Запись данных на SD карту

6.3 Сервис uartsrv

Сервис реализует доступ к порту UART0. По умолчанию сервис запущен.

Таблица 6.3 API сервиса uartsrv

Функция	Назначение
void vInitUARTService(unsigned int uartNum, uart_policy_t pol);	Запуск сервиса с настройками: BAUD115200, UART_8BIT, UART_STOPBIT1, UART_NOPARITY
int vUARTServicePrintStr(unsigned int uartNum, const char* str);	Вывод в UART строки в стиле C. Функция не потокобезопасна
int vUARTServicePrintStrn(unsigned int uartNum, const char* str, int size);	Вывод в UART size символов строки в стиле C. Функция не потокобезопасна
int vUARTServicePrintFormat(unsigned int uartNum, const char* str, ...);	Вывод в UART форматированной строки. Функция потокобезопасна
int vUARTServiceStringPrintFormat(char* s, int n, const char* str, ...);	Форматированный вывод в строку s. Аналог функции int vsnprintf (char * s, size_t n, const char * format, va_list arg); Функция потокобезопасна
unsigned char vUARTServiceRead(unsigned int uartNum);	Чтение байта из UART. Функция потокобезопасна
int vUARTServiceReadLine(unsigned int uartNum, char* str, int size);	Чтение строки данных размером size в строку на которую указывает str из UART. Если в принимаемой строке найден символ возврата каретки или символ возврата, то функция завершает работу. Функция потокобезопасна
void vUARTServiceReadBlock(unsigned int uartNum, char* dst, int size);	Чтение блока данных размером size в строку на которую указывает str из UART. Функция потокобезопасна
void vUARTServiceWrite(unsigned int uartNum, unsigned char ch);	Вывод в UART символа. Функция использует внутренний буфер. Функция потокобезопасна

Функция	Назначение
void vUARTServiceWriteBlock(unsigned int uartNum, void* src, int size);	Вывод в UART строки. Функция использует внутренний буфер. Функция потокобезопасна
void vUARTServiceFlushWrite(unsigned int uartNum);	Вытолкнуть буфер Tx в UART

7. СЕРВЕРА

7.1 HTTP и FTP

HTTP и FTP сервера контролирует сервис netsrv. По умолчанию HTTP и FTP сервера выключены, для того чтобы их включить необходимо в терминале выполнить команду server-start. FTP-сервер обеспечивает доступ к файловым системам расположенным на SD карте и RAM диске. Сервер поддерживает все стандартные операции над файлами (чтение файла, запись файла, удаление файла, копирование файла, перемещение файла, создание директории, переименование директории, удаление директории, копирование директории).

7.2 CLI

Сборка FreeRTOS содержит сервер CLI (Command Line Interface). Сервер по умолчанию включен и доступен из консоли. Сервер позволяет пользователю использовать консоль для интерпретации команд, поддерживаемых компонентами сборки. Для просмотра списка доступных команд необходимо выполнить команду «help».