SD-карта

Поддержка структуры файловой системы FAT

Реализация протокола обмена SD-карты по SPI

Подключение и инициализация интерфейса SPI

Высокоуровневые библиотеки

TI / msp430 / src /*.*

- /CTS – поддержка функций сенсорной клавиатуры
  - /structure.h – описание используемых структур данных
  - /CTS_HAL.h – функции ядра библиотеки, поддержка методов измерения RO, fRO, RC, установка прерываний таймеров
  - /CTS_Layer.h – слой API, содержит функции отслеживания базового уровня сенсора, определения нажатия каждого сенсора и т.д.
- /F5xx_F6xx_Core_Lib – библиотека ядра
  - /HAL_UCS.h – функции работы с унифицированной системой тактирования — выбор источников сигнала MCLK, SMCLK, ACLK, установка делителя, настройки генераторов XT1, XT2, режим блока FLL

Высокоуровневые библиотеки

- /F5xx_F6xx_Core_Lib – библиотека ядра
  - /HAL_PMM.h – функции работы с менеджером питания
  - /HAL_FLASH.h – библиотека для работы с FLASH-памятью
- /FatFs – стек файловой системы FAT для поддержки SD-карты
  - /diskio.h — базовые операции работы с диском
  - /mmc.h — реализация протокола MMC на базе SPI
  - /ff.h — поддержка файловой системы FAT
- /MSP-EXP430F5529_HAL — библиотека для поддержки основных устройств экспериментальной платы
  - /HAL_Wheel.h – работа с потенциометром
  - /HAL_SDCard.h – работа с SD-картой памяти

Высокоуровневые библиотеки

- /MSP-EXP430F5529_HAL — библиотека для поддержки основных устройств экспериментальной платы
  - /HAL_Dogs102x6.h – работа с ЖКИ экраном, включая простейшие графические функции
  - /HAL_Cma3000.h – работа с акселерометром
  - /HAL_Buttons.h – работа с кнопками
  - /HAL_Board.h – работа со светодиодами
  - /HAL_AppUart.h – работа с USCI в режиме UART
- /USB – стек USB для экспериментальной платы
- /UserExperienceDemo — пример приложения с использованием высокоуровневых библиотек

HAL_UCS.h

- **void LFXT_Start(uint16_t xtdrive)** — *подключение генератора LFXT1 (32 КГц). Параметр: режим после старта*
- **uint16_t LFXT_Start_Timeout(uint16_t xtdrive, uint16_t timeout)** — *запуск генератора LFXT1 с таймаутом. Возвращает 1 при выходе по таймауту. Параметры: режим запуска, таймаут*
- **void XT1_Start(uint16_t xtdrive)** — *подключение генератора XT1. Параметр: режим после старта*
- **uint16_t XT1_Start_Timeout(uint16_t xtdrive, uint16_t timeout)** — *запуск генератора XT1 с таймаутом. Возвращает 1 при выходе по таймауту. Параметры: режим запуска, таймаут*
- **void XT1_Bypass(void)** — *включение режима прямой передачи для генератора XT1*
- **void XT1_Stop(void)** — *выключение генератора XT1*
- **void XT2_Start(uint16_t xtdrive)** - *подключение генератора XT2. Параметр: режим после старта, для MSP430 = 0*

HAL_UCS.h

- **uint16_t XT2_Start_Timeout(uint16_t xtdrive, uint16_t timeout)** — *запуск генератора XT2 с таймаутом. Возвращает 1 при выходе по таймауту. Параметры: режим запуска, таймаут*
- **void XT2_Bypass(void)** — *включение режима прямой передачи для генератора XT2 для MCLK*
- **void XT2_Stop(void)** — *выключение генератора XT2*
- **void Init_FLL(uint16_t fsystem, uint16_t ratio)** — *инициализация FLL. Параметры: частота MCLK в КГц, отношение между MCLK и FLLREFCLK*
- **void Init_FLL_Settle(uint16_t fsystem, uint16_t ratio)** — *аналогична предыдущей, но ожидает вхождения в режим*
- **uint16_t Clear_All_Osc_Flags(uint16_t timeout)** — *сбрасывает все флаги ошибок генераторов. Возвращает состояние флагов. Параметр: таймаут*

HAL_UCS.h

- *Макроопределения:*
- **SELECT_FLLREF(source)** — *выбор источника FLLREF*
- **SELECT_ACLK(source)** — *выбор источника ACLK*
- **SELECT_MCLK(source)** — *выбор источника MCLK*
- **SELECT_SMCLK(source)** — *выбор источника SMCLK*
- **SELECT_MCLK_SMCLK(sources)** — *выбор источника MCLK, SMCLK*
- **ACLK_DIV(x)** — *установка делителя ACLK*
- **MCLK_DIV(x)** — *установка делителя MCLK*
- **SMCLK_DIV(x)** — *установка делителя SMCLK*
- **SELECT_FLLREFDIV(x)** — *установка делителя FLLREF*

HAL_PMM.h

- **uint16_t SetVCore(uint8_t level)** — *установка нового уровня питания. Учитывается требования пошагового переключения уровней. Соблюдается вся цепочка контроля установки уровня. Возвращает 1 в случае ошибки. Параметр: уровень напряжения*
- *Макроопределения для всех порогов монитора и супервизора:*
- **ENABLE_SVSL()** *- разрешение порога SVSL. Для остальных - аналогично*
- **DISABLE_SVSL()** *- запрещение порога SVSL. Для остальных аналогично*
- **ENABLE_SVSL_RESET()** *- разрешение сброса по SVSL. Запрещение аналогично. SVSH - аналогично*
- **ENABLE_SVML_INTERRUPT()** *- разрешение прерывания по SVML. Запрещение прерывания аналогично. SVMH — аналогично*
- **CLEAR_PMM_IFGS()** *- сброс флага прерывания*
- *Набор разрешений и запрещений для режима LPM*

HAL_FLASH.h

- **void Flash_SegmentErase(uint8_t *Flash_ptr)** — *стирает один сегмент flash-памяти, передаваемый указатель должен быть из этого сегмента*
- **uint8_t Flash_EraseCheck(uint8_t *Flash_ptr, uint16_t len)** — *проверка очистки сегмента. Параметры: указатель внутри сегмента, длина проверяемой области*
- **void FlashWrite_8(uint8_t *Data_ptr, uint8_t *Flash_ptr, uint16_t count);**
- **void FlashWrite_16(uint16_t *Data_ptr, uint16_t *Flash_ptr, uint16_t count);**
- **void FlashWrite_32(uint32_t *Data_ptr, uint32_t *Flash_ptr, uint16_t count)** — *три функции записывают соответственно count байт, слов либо двойных слов во flash-память. Параметры: указатель на передаваемые данные, указатель на область внутри flash-памяти для записи данных, количество передаваемых данных*

HAL_FLASH.h

- **void FlashMemoryFill_32(uint32_t value, uint32_t *Flash_ptr, uint16_t count)** — *заливка памяти одинаковым значением (в формате двойного слова). Параметры: устанавливаемое значение, указатель на область памяти для записи, количество записываемых данных*

HAL_Buttons.h

- **void Buttons_init(uint16_t buttonsMask)** — *подключает указанные кнопки в режиме активный уровень — низкий + подтягивающий резистор. Параметр: BUTTON_S2, BUTTON_S1, BUTTON_ALL*
- **void Buttons_interruptEnable(uint16_t buttonsMask)** — *разрешает прерывания по спаду сигнала от указанных кнопок. Параметр аналогичен*
- **void Buttons_interruptDisable(uint16_t buttonsMask)** — *запрещает прерывания от указанных кнопок. Параметр аналогичен*
- *Устанавливает* **обработчики прерываний** *для Port_1 и Port_2*
- *Глобальная переменная* **buttonsPressed** *— флаг нажатия кнопки, изменяется обработчиком прерывания*
- *Использует* **сторожевой таймер** *для защиты от дребезга контактов*
- *Использует* **HAL_Board.h, HAL_Cma3000.h**

HAL_Board.h

- **void Board_init(void)** — *конфигурирует порты (цифровой или периферия + ввод/вывод) для светодиодов, кнопок, сенсорных элементов, акселерометра, ЖКИ экрана, потенциометра, SD-карты*
- **void Board_ledOn(uint8_t ledMask)** — *включает светодиоды в соответствии с маской. Параметр: LED1..LED8, LED_ALL*
- **void Board_ledOff(uint8_t ledMask)** — *отключает светодиоды в соответствии с маской. Параметр аналогичен*
- **void Board_ledToggle(uint8_t ledMask)** — *переключает состояние светодиодов в соответствии с маской. Параметр аналогичен*

HAL_Wheel.h

- **void Wheel_init(void)** — *подключение потенциометра*
- **uint8_t Wheel_getPosition(void)** — *возвращает позицию потенциометра в диапазоне 0..7*
- **uint16_t Wheel_getValue(void)** — *возвращает цифровое значение потенциометра, используется защита от шума*
- **void Wheel_disable(void)** — *отключение потенциометра*
- **void Wheel_enable(void)** — *подключение потенциометра*
- *Используется* **ADC** *с каналом 5 (потенциометр)*
- *Устанавливается* **обработчик прерывания** *для ADC*

HAL_AppUart.h

- **void AppUart_init(void)** — *инициализация UART (USCI_A1)*
- **uint8_t AppUart_getChar(void)** — *прием байта*
- **void AppUart_putChar(uint8_t transmitChar)** — *передача байта. Параметр: передаваемое значение*
- *Использует* **HAL_UCS.h, HAL_AppUart.h, HAL_Dogs102x6.h**

HAL_Dogs102x6.h

- **void Dogs102x6_init(void)** — *подключение и инициализация экрана*
- **void Dogs102x6_backlightInit(void)** — *включение подсветки*
- **void Dogs102x6_setBacklight(uint8_t brightness)** — *установка яркости подсветки. Параметр: уровень яркости*
- **void Dogs102x6_disable(void)** — *отключение экрана*
- **void Dogs102x6_writeCommand(uint8_t* sCmd, uint8_t i)** — *передача серии команд. Параметры: указатель на буфер с командами, количество команд*
- **void Dogs102x6_writeData(uint8_t* sData, uint8_t i)** — *запись данных. Текущий адрес пикселя сдвигается. Параметры: указатель на буфер с данными, длина данных*
- **uint8_t Dogs102x6_getContrast(void)** — *возвращает текущее значение контраста*
- **uint8_t Dogs102x6_getBacklight(void)** — *возвращает текущее значение уровня яркости*

HAL_Dogs102x6.h

- **void Dogs102x6_refresh(uint8_t mode)** — *отображение содержимого памяти на экране и установка режима вывода на экран (сразу или изменения накапливаются в памяти). Параметр: 0 — по запросу, 1 - сразу*
- **void Dogs102x6_setAddress(uint8_t pa, uint8_t ca)** — *установка адреса пискселя. Параметры: номер страницы, номер столбца*
- **void Dogs102x6_setContrast(uint8_t newContrast)** — *установка контраста. Параметр: уровень контраста 0..31, 31 — самый темный*
- **void Dogs102x6_setInverseDisplay(void)** — *инвертирует отображение на экране. Память экрана не изменяется*
- **void Dogs102x6_clearInverseDisplay(void)** — *отключает инвертированный вывод на экран*
- **void Dogs102x6_scrollLine(uint8_t lines)** — *установка режима скроллинга. Параметр: начальная строка 0..63*
- **void Dogs102x6_setAllPixelsOn(void)** — *включение всех пикселей*

HAL_Dogs102x6.h

- **void Dogs102x6_clearAllPixelsOn(void)** — *отмена режима включения всех пикселей*
- **void Dogs102x6_clearScreen(void)** — *очистка экрана. Память также сбрасывается в 0*
- **void Dogs102x6_charDraw(uint8_t row, uint8_t col, uint16_t f, uint8_t style)** — *вывод символа на экран. Параметры: страница (0-7), столбец (0-102), символ, стиль (0 — прямой, 1 — инверсный)*
- **void Dogs102x6_charDrawXY(uint8_t x, uint8_t y, uint16_t f, uint8_t style)**— *вывод символа на экран. Параметры: x (0-102), y (0-63), символ, стиль (0 — прямой, 1 — инверсный)*
- **void Dogs102x6_stringDraw(uint8_t row, uint8_t col, char *word, uint8_t style)** — *вывод строки символов на экран. Параметры: страница (0-7), столбец (0-102), указатель на строку (признак конца строки - 0), стиль (0 — прямой, 1 — инверсный)*

HAL_Dogs102x6.h

- **void Dogs102x6_stringDrawXY(uint8_t x, uint8_t y, char *word, uint8_t style)** — *вывод строки символов на экран. Параметры: x (0-102), y (0-63), указатель на строку (признак конца строки - 0), стиль (0 — прямой, 1 — инверсный)*
- **void Dogs102x6_clearRow(uint8_t row)** — *очистка страницы (стро-ки). Память также изменяется. Параметр: номер страницы 0-7*
- **void Dogs102x6_pixelDraw(uint8_t x, uint8_t y, uint8_t style)** — *вывод пикселя. Параметры: x (0-102), y(0-63), стиль (0 — темный, 1 - светлый)*
- **void Dogs102x6_horizontalLineDraw(uint8_t x1, uint8_t x2, uint8_t y, uint8_t style)** — *вывод горизонтальной линии. Параметры: x1, x2 — начало и конец линии (0-102), y (0-63), стиль (0 — темная, 1 — светлая)*
- **void Dogs102x6_verticalLineDraw(uint8_t y1, uint8_t y2, uint8_t x, uint8_t style)** — *вывод вертикальной линии. Параметры: x (0-102), y1, y2 — начало и конец линии (0-63), стиль (0 — темная, 1 - светлая)*

HAL_Dogs102x6.h

- **void Dogs102x6_lineDraw(uint8_t x1, uint8_t y1, uint8_t x2, uint8_t y2, uint8_t style)** — *вывод линии на экран. Параметры: x1, y1 — начало линии, x2, y2 — конец линии, стиль (0 — темная, 1 - светлая). x1, x2 (0-102), y1, y2 (0-63)*
- **void Dogs102x6_circleDraw(uint8_t x, uint8_t y, uint8_t radius, uint8_t style)** — *вывод окружности на экран. Параметры: x (0-102), y (0-63) — координаты центра, радиус, стиль (0 -темная, 1-светлая)*
- **void Dogs102x6_imageDraw(const uint8_t IMAGE[], uint8_t row, uint8_t col)** — *вывод изображения на экран. Параметры: массив пикселей изображения (первый байт массива — ширина в пикселях, второй — высота в страницах == пиксели / 8), левая верхняя координата вывода: страница (строка) 0-7, столбец (0-102)*
- **void Dogs102x6_clearImage(uint8_t height, uint8_t width, uint8_t row, uint8_t col)** — *очищает область экрана. Параметры: высота в страницах (строках) 1-8, ширина (1-102), левая верхняя координата области: страница (строка) 0-7, столбец (0-102)*

HAL_Dogs102x6.h

- Использует **HAL_buttons.h**
- Использует **USCI_B1, таймер B0**
- Размер шрифта 6 x 8
- Использует глобальные переменные для отслеживания состояния и памяти экрана

HAL_Cma3000.h

- **void Cma3000_init(void)** — *подключение и инициализация акселерометра в режиме 2g / 400Гц*
- **void Cma3000_disable(void)** — *отключение акселерометра*
- **void Cma3000_readAccel(void)** — *получение данных с акселерометра*
- **void Cma3000_setAccel_offset(int8_t xAccel_offset, int8_t yAccel_offset, int8_t zAccel_offset)** — *устанавливает значения смещаний по осям. Параметры: значения смещений по осям*
- **void Cma3000_readAccel_offset(void)** — *получение данных с акселерометра с учетом смещений*
- **int8_t Cma3000_readRegister(uint8_t Address)** — *чтение произвольного регистра акселерометра. Параметр: адрес регистра (DOUTX, DOUTY, DOUTZ, остальные по номерам)*

HAL_Cma3000.h

- **int8_t Cma3000_writeRegister(uint8_t Address, int8_t Data)** — *запись байта в указанный регистр. Параметры: номер регистра (DOUTX, DOUTY, DOUTZ, остальные по номерам), записываемые данные*
- *Используется* **USCI_A0**
- *Значение по осям хранится в* **глобальных переменных** *Cma3000_xAccel, Cma3000_yAccel, Cma3000_zAccel*
- *Использует* **HAL_UCS.h**

CTS_HAL.h

- *Параметры всех функций: первый — структура с описанием сенсора и параметров измерения (структура описана в /structure.h), второй — указатель на буфер для записи результатов измерений*
- *Наименование функции включает: TI_CTS_<Method>_<IO>_<Timer1>_<Timer2>_HAL, где*
- *Method* = RO, fRO, RC
- *IO* = CSIO, COMPB, PINOSC, COMPAp, PAIR
- *Timer1* = TA2, TB0, TA0, SW, TA1
- *Timer2* = WDTA, TA3, TA1, WDTp, SW, TA0
- *Устанавливает обработчики прерываний:* TIMER0_A0, TIMER1_A0, TIMER2_A0, TIMER3_A0

CTS_HAL.h

- void TI_CTS_RO_CSIO_TA2_WDTA_HAL(const struct Sensor *, uint16_t *)
- void TI_CTS_RO_CSIO_TA2_TA3_HAL()
- void TI_CTS_fRO_CSIO_TA2_TA3_HAL()
- void TI_CTS_RO_COMPB_TB0_WDTA_HAL()
- void TI_CTS_RO_PINOSC_TA0_TA1_HAL()
- void TI_CTS_fRO_PINOSC_TA0_TA1_HAL()
- void TI_CTS_RO_COMPAp_TA0_WDTp_HAL()
- void TI_CTS_fRO_COMPAp_TA0_SW_HAL()
- void TI_CTS_fRO_COMPAp_SW_TA0_HAL()
- void TI_CTS_RO_COMPAp_TA1_WDTp_HAL()
- void TI_CTS_fRO_COMPAp_TA1_SW_HAL()
- void TI_CTS_RC_PAIR_TA0_HAL()
- void TI_CTS_RO_PINOSC_TA0_WDTp_HAL()

CTS_HAL.h

- void TI_CTS_RO_PINOSC_TA0_HAL()
- void TI_CTS_fRO_PINOSC_TA0_SW_HAL()
- void TI_CTS_RO_COMPB_TA0_WDTA_HAL()
- void TI_CTS_fRO_COMPB_TA0_SW_HAL()
- void TI_CTS_RO_COMPB_TA1_WDTA_HAL()
- void TI_CTS_fRO_COMPB_TA1_SW_HAL()
- void TI_CTS_RO_COMPB_TA1_TA0_HAL()
- void TI_CTS_fRO_COMPB_TA1_TA0_HAL()

CTS_Layer.h

- **void TI_CAPT_Init_Baseline(const struct Sensor*)** - *инициализация базовой емкости сенсора. Параметр: указатель на структуру сенсора*
- **void TI_CAPT_Update_Baseline(const struct Sensor*, uint8_t)** — *отслеживание базовой емкости сенсора путем усреднения нескольких измерений. Параметры: указатель на структуру сенсора, количество измерений*
- **void TI_CAPT_Reset_Tracking(void)** — *установка алгоритма отслеживания базовой емкости в начальное состояние*
- **void TI_CAPT_Update_Tracking_DOI(uint8_t)** — *установка направления отслеживания алгоритма отслеживания базовой емкости. Параметр: направление (повышение/понижение)*

CTS_Layer.h

- **void TI_CAPT_Update_Tracking_Rate(uint8_t)** — *установка скорости реагирования на изменения алгоритма отслеживания базовой емкости. Параметр: скорость реагирования, биты 4-5 — скорость по направлению изменений: TRIDOI_VSLOW - 00, TRIDOI_SLOW - 01, TRIDOI_MED — 10, TRIDOI_FAST — 11, биты 6-7 — скорость против направления изменений: TRADOI_FAST — 00, TRADOI_MED - 01, TRADOI_SLOW - 10, TRADOI_VSLOW - 11*
- **void TI_CAPT_Raw(const struct Sensor*, uint16_t*)** - *измерение емкости каждого сенсора. Параметры — указатель на структуру сенсора, указатель на буфер данных*
- **void TI_CAPT_Custom(const struct Sensor *, uint16_t*)** - *измерение изменения емкости сенсора. Параметры: структура сенсора для измерения, буфер данных для хранения результата*
- **uint8_t TI_CAPT_Button(const struct Sensor *)** - *определение нажатия кнопки. Результат: 0 — не нажата, 1 — нажата. Параметр: указатель на структуру сенсора для измерения*

CTS_Layer.h

- **const struct Element * TI_CAPT_Buttons(const struct Sensor *)** - *определение нажатия любой кнопки. Результат: структура элементов для каждой кнопки (0 — не нажата, 1 — нажата). Параметр: указатель на структуру сенсора для измерения*
- **uint16_t TI_CAPT_Slider(const struct Sensor*)** - *определение позиции слайдера. Результат — номер позиции или недопустимое значение, если кнопка не нажата. Параметр — структура сенсора для измерения*
- ***uint16_t TI_CAPT_Wheel(const struct Sensor*)*** - *определение позиции кольцевого слайдера. Результат — номер позиции или недопустимое значение, если кнопка не нажата. Параметр — структура сенсора для измерения*
- *Использует* **CTS_HAL.h**

HAL_SDCard.h

- **void SDCard_init(void)** — *подключение линий микроконтроллера и инициализация интерфейса SPI в режиме 3-проводной, Master, MSB, 8-бит, активный уровень CLK – низкий, источник тактирования SMCLK, частота тактирования 397 КГц (при инициализации должна быть менее 400 КГц)*
- **void SDCard_fastMode(void)** — *устанавливает тактовую частоту 12,5 МГц для быстрого обмена*
- **void SDCard_readFrame(uint8_t *pBuffer, uint16_t size)** — *чтение данных из памяти, 1 параметр — указатель на буфер приема, 2 параметр — количество байт*
- **void SDCard_sendFrame(uint8_t *pBuffer, uint16_t size)** — *запись данных в память, 1 параметр — указатель на буфер с данными, 2 параметр — количество байт*
- **void SDCard_setCSHigh(void)** — *установка сигнала выбора в 1*
- **void SDCard_setCSLow(void)** — *сброс сигнала выбора в 0*
- *Использует* **USCI_B1**

mmc.h

- **DSTATUS disk_status (BYTE drv)** — *получение состояния диска. Передается номер диска (0)*
- **DSTATUS disk_initialize (BYTE drv)** — *инициализация диска. Параметры и результат аналогичны предыдущей функции*
- **DRESULT disk_read (BYTE drv, BYTE *buff, DWORD sector, BYTE count)** – *чтение данных с диска. Параметры: номер диска, указатель на буфер для размещения данных, начальный номер сектора для чтения (LBA), количество секторов*
- **DRESULT disk_write (BYTE drv, const BYTE *buff, DWORD sector, BYTE count)** — *запись данных на диск. Параметры аналогичны*
- **DRESULT disk_ioctl (BYTE drv, BYTE ctrl, void *buff)** — *команда управления. Параметры: номер диска, код команды, указатель на буфер для приема/передачи данных команды управления*
- **uint8_t detectCard(void)** — *обнаружение карты и попытка подключения, если карта не обнаружена. Возвращает 1, если карта готова к работе, 0 — карта не обнаружена*
- *использует* **diskio.h, HAL_SDCard.h**

ff.h

- **FRESULT f_mount (BYTE, FATFS*)** — *подключение/отключение логического диска*
- **FRESULT f_open (FIL*, const TCHAR*, BYTE)** — *открытие или создание файла*
- **FRESULT f_read (FIL*, void*, UINT, UINT*)** — *чтение данных из файла*
- **FRESULT f_lseek (FIL*, DWORD)** — *перемещение файлового указателя*
- **FRESULT f_close (FIL*)** — *закрытие открытого файла*
- **FRESULT f_opendir (DIRS*, const TCHAR*)** — *открытие существующего каталога*
- **FRESULT f_readdir (DIRS*, FILINFO*)** — *чтение элементов каталога*
- **FRESULT f_stat (const TCHAR*, FILINFO*)** — *получение состояния файла*

ff.h

- **FRESULT f_write (FIL*, const void*, UINT, UINT*)** — *запись данных в файл*
- **FRESULT f_getfree (const TCHAR*, DWORD*, FATFS**)** — *получение количества свободных кластеров на диске*
- **FRESULT f_truncate (FIL*)** — *усечение файла*
- **FRESULT f_sync (FIL*)** — *очистка кеша для записанных данных*
- **FRESULT f_unlink (const TCHAR*)** — *удаление существующего файла или каталога*
- **FRESULT f_mkdir (const TCHAR*)** — *создание нового каталога*
- **FRESULT f_chmod (const TCHAR*, BYTE, BYTE)** — *изменение атрибутов файла или каталога*
- **FRESULT f_utime (const TCHAR*, const FILINFO*)** — *изменение времени файла или каталога*
- **FRESULT f_rename (const TCHAR*, const TCHAR*)** — *переименование или перемещение файла или каталога*

ff.h

- **FRESULT f_forward (FIL*, UINT(*)(const BYTE*,UINT), UINT, UINT*)** — *помещение данных в поток*
- **FRESULT f_mkfs (BYTE, BYTE, UINT)** — *создание файловой системы на диске*
- **FRESULT f_chdrive (BYTE)** — *смена текущего диска*
- **FRESULT f_chdir (const TCHAR*)** — *смена текущего каталога*
- **FRESULT f_getcwd (TCHAR*, UINT)** — *получение текущего каталога*
- **int f_putc (TCHAR, FIL*)** — *запись символа в файл*
- **int f_puts (const TCHAR*, FIL*)** — *запись строки в файл*
- **int f_printf (FIL*, const TCHAR*, …)** — *запись форматированной строки в файл*
- **TCHAR* f_gets (TCHAR*, int, FIL*)** — *чтение строки из файла*
- *Использует* **diskio.h**

ff.h

- *Макроопределения:*
- f_eof(fp)
- f_error(fp)
- f_tell(fp)
- f_size(fp)

USB