



ОСРВ MULTEX-ARM

Руководство Программиста



ООО "Сэт Код"

195248, Россия, Санкт-Петербург,

Новомалиновская дорога, 6А

8 (921) 971-00-80

set-code.ru

Содержание

1	Общее описание	21
1.1	Операционная система жесткого реального времени MULTEX-ARM	21
1.2	Наборы библиотек	22
1.3	История версий	23
2	Сборка проекта	24
2.1	Среда сборки	24
2.1.1	Подготовка WSL	24
2.1.2	Установка пакетов	25
2.1.3	Настройка переменных окружения	25
2.2	Описание структуры проекта пользователя	25
2.3	Файл конфигурации config.h	26
2.3.1	Выбор платформы	26
2.3.2	Сопроцессор	26
2.3.3	Кэш память	27
2.3.4	Отладочная консоль	27
2.3.5	Файловая система ОС MS-DOS	27
2.3.6	Сеть	27
2.3.7	Процедуры пользователя	28
2.3.8	Пример написания config.h	28
2.4	Конфигурация аппаратной части	29
2.4.1	Приоритет записей в списке параметров конфигурации	30
2.4.2	Файл конфигурации	30
2.5	Файл сборки Makefile	31
2.5.1	Имя выходного файла	32
2.5.2	Определение флагов компилятора	32
2.5.3	Распределение памяти	32
2.5.4	Настройка путей	33
2.5.5	Настройка подключаемых библиотек	34
2.5.6	Подключение примеров и тестовых функций	34
2.5.7	Пример написания Makefile	34
2.6	Сборка проекта (библиотеки)	35
2.6.1	Вызов справки	35
2.6.2	Отладочная сборка	35

2.6.3	Сборка поставочной версии	35
2.6.4	Сборка библиотек	36
2.6.5	Прочие цели сборки	36
2.7	Запуск на целевой платформе	36
3	Интерпретатор команд SHELL	38
3.1	Справка Shell	39
3.2	Справка работы с диском	40
3.3	Дамп памяти	40
3.4	Модифицировать дамп памяти	40
3.5	Просмотр задач	41
3.6	Открытые файлы, сокеты, устройства	41
3.7	Установленные устройства	41
3.8	Информация о сети	42
3.9	Удалить задачу	42
3.10	Изменить приоритет задачи	42
3.11	Запустить задачу	43
3.12	Выполнить процедуру	43
3.13	Аппаратная перезагрузка	43
3.14	Работа с переменными	43
3.14.1	Просмотр переменной	43
3.14.2	Изменение переменной в десятичном виде	43
3.14.3	Изменение переменной в шестнадцатеричном виде	44
3.14.4	Изменение строковой переменной	44
3.15	Работа с диском	44
3.15.1	Просмотр каталога	44
3.15.2	Смена каталога / диска	45
3.15.3	Удаление файла	45
3.15.4	Копирование файла	45
3.15.5	Удаление каталога	46
3.15.6	Создание каталога	46
3.15.7	Проверка диска	46
3.16	Встроенный текстовый редактор	46
3.16.1	Просмотр файла	46
3.16.2	Редактирование файла	47
3.17	Поддерживаемые сочетания клавиш	47

3.17.1	История команд	47
3.17.2	Навигация	47
3.17.3	Редактирование	47
4	Ядро операционной системы	48
4.1	Многозадачность и межзадачное взаимодействие	48
4.1.1	Управление прерываниями	49
4.1.2	Приоритеты прерываний и задач	50
4.1.3	Таймеры	51
4.1.4	Сигналы	52
4.1.5	Семафоры	52
4.1.6	Очереди сообщений	52
4.2	Базовая система ввода / вывода	53
4.2.1	Блочные устройства и файловые системы	54
4.3	Диспетчер памяти	54
4.4	Работа с встроенной КЭШ-памятью процессора	55
4.5	Нелокальные переходы	55
4.6	Работа с ini-файлами	55
4.7	Работа с межпроцессорными каналами.	56
4.8	PLL – распределение тактовых частот	56
5	Аппаратные интерфейсы	57
5.1	GPIO – Порты ввода/вывода	57
5.2	PWM (ШИМ)	57
5.3	SPI	58
5.4	I2C	59
5.5	UART	59
6	Аппаратная поддержка мультимедиа	61
6.1	Графическая подсистема	61
6.1.1	Общее описание	61
6.1.2	Инициализация графического адаптера	62
6.1.3	Работа с поверхностями	62
6.1.4	Известные ошибки работы с поверхностями	71
6.2	Работа с оверлеями	72
6.2.1	Пример работы с оверлеем	72
6.3	Аппаратный TV-декодер	72

6.4	Поддержка шрифтов FreeType	73
6.4.1	Пример работы со шрифтами	73
6.5	Работа с AVI-файлами	74
6.5.1	Пример воспроизведения AVI файла через overlay	75
6.5.2	Пример воспроизведения AVI файла через 2D акселератор	76
6.6	Кодер/декодер видео h.264 CEDRUS	77
6.7	Звуковая подсистема	77
7	Программный вывод графики	78
7.1	Графика на простых процессорах	78
7.1.1	Общее описание	78
7.1.2	Работа с поверхностями	78
8	Сетевая подсистема	80
8.1	Подключение к проекту	80
8.2	Протокол UDP	80
8.3	Протокол TCP/IP, сокет TCP	81
9	Подсистема USB	82
9.1	Подключение к проекту	82
9.2	Общее описание	82
9.3	Получение дескрипторов из структуры usb_device	83
9.4	Использование интегрального параметра pipe	83
10	Поддержка CSI (Camera Sensor Interface)	85
10.1	Работа с цифровыми видекамерами	85
11	Ошибки	86
12	Список устаревших определений и описаний	87
13	Список экспериментальных опций	88
14	Алфавитный указатель групп	89
14.1	Группы	89
15	Алфавитный указатель структур данных	90
15.1	Структуры данных	90
16	Список файлов	94

16.1	Файлы	94
17	Группы	99
17.1	SCI (Camera Sensor Interface)	99
17.1.1	Подробное описание	99
17.2	USB	100
17.2.1	Подробное описание	100
17.3	Драйвера интерфейсов	101
17.3.1	Подробное описание	101
17.4	Мультимедиа	102
17.4.1	Подробное описание	102
17.5	Стандартные типы	103
17.5.1	Подробное описание	103
17.6	Ядро MULTEX-ARM	104
17.6.1	Подробное описание	105
18	Структуры данных	106
18.1	Структура blk_cache	106
18.1.1	Подробное описание	106
18.1.2	Поля	106
18.2	Структура blk_dev	108
18.2.1	Поля	108
18.3	Структура complex	110
18.3.1	Поля	110
18.4	Структура date_time	111
18.4.1	Подробное описание	111
18.4.2	Поля	111
18.5	Структура device_header	113
18.5.1	Подробное описание	113
18.5.2	Поля	113
18.6	Структура Display	114
18.6.1	Поля	114
18.7	Структура div_t	116
18.7.1	Подробное описание	116
18.7.2	Поля	116
18.8	Структура dtcompact	117

18.8.1	Подробное описание	117
18.8.2	Поля	117
18.9	Структура <code>env_var</code>	119
18.9.1	Поля	119
18.10	Структура <code>exit_st</code>	120
18.10.1	Подробное описание	120
18.10.2	Поля	120
18.11	Структура <code>ffblk</code>	121
18.11.1	Подробное описание	121
18.11.2	Поля	121
18.12	Структура <code>FILE</code>	123
18.12.1	Поля	123
18.13	Структура <code>file_fcb</code>	124
18.13.1	Подробное описание	124
18.13.2	Поля	124
18.14	Структура <code>g2d_blt</code>	126
18.14.1	Поля	126
18.15	Структура <code>g2d_fillrect</code>	128
18.15.1	Поля	128
18.16	Структура <code>g2d_image</code>	129
18.16.1	Подробное описание	129
18.16.2	Поля	129
18.17	Структура <code>g2d_rect</code>	130
18.17.1	Подробное описание	130
18.17.2	Поля	130
18.18	Структура <code>g2d_stretchblt</code>	131
18.18.1	Поля	131
18.19	Структура <code>imaxdiv_t</code>	132
18.19.1	Поля	132
18.20	Структура <code>in_addr</code>	133
18.20.1	Подробное описание	133
18.20.2	Поля	133
18.21	Структура <code>iniBinaryArray</code>	134
18.21.1	Подробное описание	134
18.21.2	Поля	134

18.22 Структура iniCoords	135
18.22.1 Подробное описание	135
18.22.2 Поля	135
18.23 Структура iniIntArray	136
18.23.1 Подробное описание	136
18.23.2 Поля	136
18.24 Структура iniRect	137
18.24.1 Подробное описание	137
18.24.2 Поля	137
18.25 Структура ip_packet	138
18.25.1 Подробное описание	138
18.25.2 Поля	138
18.26 Структура jmp_buf	139
18.26.1 Подробное описание	139
18.26.2 Поля	139
18.27 Структура ldiv_t	140
18.27.1 Подробное описание	140
18.27.2 Поля	140
18.28 Структура listNode	141
18.28.1 Подробное описание	141
18.28.2 Поля	141
18.29 Структура msgQID	142
18.29.1 Поля	142
18.30 Структура REG_SET	144
18.30.1 Подробное описание	144
18.30.2 Поля	144
18.31 Структура sDisplayInfo	146
18.31.1 Поля	146
18.32 Структура seekblk	147
18.32.1 Подробное описание	147
18.32.2 Поля	147
18.33 Структура Sem_Id	148
18.33.1 Подробное описание	148
18.33.2 Поля	148
18.34 Структура sigaction	150

18.34.1	Подробное описание	150
18.34.2	Поля	150
18.35	Структура siginfo	151
18.35.1	Подробное описание	151
18.35.2	Поля	151
18.36	Объединение sigval	152
18.36.1	Поля	152
18.37	Структура sList	153
18.37.1	Подробное описание	153
18.37.2	Поля	153
18.38	Структура sockaddr	154
18.38.1	Подробное описание	154
18.38.2	Поля	154
18.39	Структура sockaddr_in	155
18.39.1	Подробное описание	155
18.39.2	Поля	155
18.40	Структура sTtfFont	156
18.40.1	Подробное описание	156
18.40.2	Поля	156
18.41	Структура sVector	157
18.41.1	Подробное описание	157
18.41.2	Поля	157
18.42	Структура tagSURFACE	158
18.42.1	Поля	158
18.43	Структура TCB	159
18.43.1	Поля	160
18.44	Структура tDrvBit	164
18.44.1	Подробное описание	164
18.44.2	Поля	164
18.45	Структура tDrvBitGroup	165
18.45.1	Подробное описание	165
18.45.2	Поля	165
18.46	Структура tDrvGpio	166
18.46.1	Подробное описание	166
18.46.2	Поля	166

18.47 Структура textRect	167
18.47.1 Поля	167
18.48 Структура timespec	168
18.48.1 Подробное описание	168
18.48.2 Поля	168
18.49 Структура tm	169
18.49.1 Подробное описание	169
18.49.2 Поля	169
18.50 Структура tMapIterators	171
18.50.1 Подробное описание	171
18.50.2 Поля	171
18.51 Структура tRingBuffer	172
18.51.1 Поля	172
18.52 Структура tScreenDeviceMode	173
18.52.1 Подробное описание	173
18.52.2 Поля	173
18.53 Структура udp_hdr	175
18.53.1 Подробное описание	175
18.53.2 Поля	175
18.54 Структура udp_service	176
18.54.1 Поля	176
18.55 Структура usb_class_abstract_control_descriptor	177
18.55.1 Поля	177
18.56 Структура usb_class_atm_networking_descriptor	178
18.56.1 Поля	178
18.57 Структура usb_class_call_management_descriptor	180
18.57.1 Поля	180
18.58 Структура usb_class_capi_control_descriptor	181
18.58.1 Поля	181
18.59 Структура usb_class_country_selection_descriptor	182
18.59.1 Поля	182
18.60 Структура usb_class_descriptor	183
18.60.1 Поля	183
18.61 Структура usb_class_direct_line_descriptor	186
18.61.1 Поля	186

18.62 Структура <code>usb_class_ethernet_networking_descriptor</code>	187
18.62.1 Поля	187
18.63 Структура <code>usb_class_extension_unit_descriptor</code>	189
18.63.1 Поля	189
18.64 Структура <code>usb_class_function_descriptor</code>	190
18.64.1 Поля	190
18.65 Структура <code>usb_class_function_descriptor_generic</code>	191
18.65.1 Поля	191
18.66 Структура <code>usb_class_header_function_descriptor</code>	192
18.66.1 Поля	192
18.67 Структура <code>usb_class_hid_descriptor</code>	193
18.67.1 Поля	193
18.68 Структура <code>usb_class_mdldm_descriptor</code>	194
18.68.1 Поля	194
18.69 Структура <code>usb_class_mdldm_descriptor</code>	195
18.69.1 Поля	195
18.70 Структура <code>usb_class_multi_channel_descriptor</code>	196
18.70.1 Поля	196
18.71 Структура <code>usb_class_network_channel_descriptor</code>	197
18.71.1 Поля	197
18.72 Структура <code>usb_class_protocol_unit_function_descriptor</code>	198
18.72.1 Поля	198
18.73 Структура <code>usb_class_report_descriptor</code>	199
18.73.1 Поля	199
18.74 Структура <code>usb_class_telephone_call_descriptor</code>	200
18.74.1 Поля	200
18.75 Структура <code>usb_class_telephone_operational_descriptor</code>	201
18.75.1 Поля	201
18.76 Структура <code>usb_class_telephone_ringer_descriptor</code>	202
18.76.1 Поля	202
18.77 Структура <code>usb_class_union_function_descriptor</code>	203
18.77.1 Поля	203
18.78 Структура <code>usb_class_usb_terminal_descriptor</code>	204
18.78.1 Поля	204
18.79 Структура <code>usb_config</code>	206

18.79.1	Подробное описание	206
18.79.2	Поля	206
18.80	Структура <code>usb_configuration_descriptor</code>	207
18.80.1	Подробное описание	207
18.80.2	Поля	207
18.81	Структура <code>usb_descriptor</code>	209
18.81.1	Поля	209
18.82	Структура <code>usb_device</code>	211
18.82.1	Поля	211
18.83	Структура <code>usb_device_descriptor</code>	215
18.83.1	Подробное описание	215
18.83.2	Поля	215
18.84	Структура <code>usb_endpoint_descriptor</code>	218
18.84.1	Подробное описание	218
18.84.2	Поля	218
18.85	Структура <code>usb_generic_descriptor</code>	221
18.85.1	Поля	221
18.86	Структура <code>usb_interface</code>	222
18.86.1	Подробное описание	222
18.86.2	Поля	222
18.87	Структура <code>usb_interface_descriptor</code>	223
18.87.1	Подробное описание	223
18.87.2	Поля	223
18.88	Структура <code>usb_string_descriptor</code>	225
18.88.1	Подробное описание	225
18.88.2	Поля	225
19	Файлы	226
19.1	Файл <code>a20graph.h</code>	226
19.1.1	Подробное описание	229
19.1.2	Макросы	229
19.1.3	Типы	232
19.1.4	Перечисления	232
19.1.5	Функции	240
19.1.6	Переменные	251
19.2	Файл <code>arch.h</code>	253

19.2.1	Подробное описание	254
19.2.2	Функции	254
19.3	Файл archdef.h	262
19.3.1	Подробное описание	262
19.3.2	Макросы	262
19.4	Файл assert.h	265
19.4.1	Подробное описание	265
19.4.2	Макросы	265
19.4.3	Функции	265
19.5	Файл avi.dox	267
19.6	Файл avilib.h	268
19.6.1	Подробное описание	268
19.6.2	Макросы	269
19.6.3	Типы	269
19.6.4	Функции	269
19.7	Файл blkcache.h	274
19.7.1	Типы	274
19.7.2	Функции	274
19.8	Файл cache.h	278
19.8.1	Подробное описание	278
19.8.2	Функции	278
19.9	Файл cedrus.h	281
19.9.1	Подробное описание	281
19.9.2	Перечисления	282
19.9.3	Функции	282
19.10	Файл console.h	288
19.10.1	Подробное описание	288
19.10.2	Функции	288
19.11	Файл crc32.h	291
19.11.1	Функции	291
19.12	Файл crc8.h	292
19.12.1	Функции	292
19.13	Файл crt.h	294
19.13.1	Макросы	295
19.13.2	Функции	298

19.14	Файл <code>csi.h</code>	303
19.14.1	Подробное описание	304
19.14.2	Макросы	304
19.14.3	Перечисления	304
19.14.4	Функции	309
19.15	Файл <code>ctype.h</code>	313
19.15.1	Подробное описание	313
19.15.2	Макросы	313
19.15.3	Функции	314
19.16	Файл <code>datetime.h</code>	321
19.16.1	Типы	321
19.16.2	Функции	321
19.17	Файл <code>de2.h</code>	324
19.17.1	Подробное описание	324
19.17.2	Перечисления	325
19.17.3	Функции	326
19.18	Файл <code>drivers.dox</code>	329
19.19	Файл <code>env_vars.h</code>	330
19.19.1	Подробное описание	330
19.19.2	Функции	330
19.20	Файл <code>errno-base.h</code>	331
19.20.1	Подробное описание	331
19.21	Файл <code>errno.h</code>	332
19.21.1	Подробное описание	334
19.21.2	Макросы	334
19.21.3	Переменные	349
19.22	Файл <code>filesyst.h</code>	350
19.22.1	Подробное описание	350
19.22.2	Функции	350
19.23	Файл <code>fnames.h</code>	358
19.23.1	Функции	358
19.24	Файл <code>fonts.h</code>	364
19.24.1	Подробное описание	365
19.24.2	Типы	365
19.24.3	Перечисления	365

19.24.4	Функции	367
19.25	Файл <code>fontsdefines.h</code>	376
19.25.1	Подробное описание	376
19.25.2	Макросы	376
19.25.3	Типы	377
19.25.4	Перечисления	377
19.26	Файл <code>grio.h</code>	379
19.26.1	Подробное описание	380
19.26.2	Макросы	380
19.26.3	Перечисления	381
19.26.4	Функции	381
19.27	Файл <code>i2c.h</code>	387
19.27.1	Подробное описание	387
19.27.2	Макросы	387
19.27.3	Функции	389
19.28	Файл <code>inifiles.h</code>	393
19.28.1	Подробное описание	395
19.28.2	Типы	395
19.28.3	Функции	395
19.29	Файл <code>inputstr.h</code>	410
19.29.1	Перечисления	410
19.29.2	Функции	411
19.30	Файл <code>intl.h</code>	412
19.30.1	Подробное описание	412
19.30.2	Макросы	412
19.30.3	Типы	413
19.30.4	Функции	414
19.31	Файл <code>inttypes.h</code>	416
19.31.1	Подробное описание	418
19.31.2	Макросы	419
19.31.3	Функции	436
19.32	Файл <code>io.h</code>	438
19.32.1	Подробное описание	441
19.32.2	Макросы	441
19.32.3	Типы	445

19.32.4	Функции	447
19.32.5	Переменные	465
19.33	Файл iso646.h	466
19.33.1	Подробное описание	466
19.33.2	Макросы	466
19.34	Файл kernel.dox	468
19.35	Файл limits.h	469
19.35.1	Подробное описание	470
19.35.2	Макросы	470
19.36	Файл list.h	473
19.36.1	Подробное описание	473
19.36.2	Макросы	474
19.36.3	Типы	474
19.36.4	Функции	474
19.37	Файл manual.dox	480
19.38	Файл mapstr.h	481
19.38.1	Подробное описание	482
19.38.2	Макросы	482
19.38.3	Перечисления	482
19.38.4	Функции	482
19.39	Файл math.h	487
19.39.1	Макросы	488
19.39.2	Функции	488
19.40	Файл memlib.h	495
19.40.1	Подробное описание	495
19.40.2	Макросы	495
19.40.3	Функции	496
19.41	Файл mpeg4codec.h	501
19.41.1	Функции	501
19.42	Файл msgqlib.h	504
19.42.1	Подробное описание	504
19.42.2	Макросы	505
19.42.3	Типы	506
19.42.4	Функции	506
19.43	Файл multex.h	510

19.43.1	Подробное описание	511
19.43.2	Макросы	511
19.43.3	Типы	518
19.43.4	Перечисления	518
19.44	Файл multimedia.dox	520
19.45	Файл names.h	521
19.45.1	Функции	521
19.46	Файл net.dox	523
19.47	Файл pipelib.h	524
19.47.1	Функции	524
19.48	Файл pll.h	525
19.48.1	Подробное описание	525
19.48.2	Макросы	526
19.48.3	Функции	527
19.49	Файл project.dox	531
19.50	Файл rwm.h	532
19.50.1	Подробное описание	532
19.50.2	Макросы	532
19.50.3	Функции	533
19.51	Файл ringbuffer.h	536
19.51.1	Подробное описание	536
19.51.2	Функции	536
19.52	Файл semlib.h	540
19.52.1	Подробное описание	541
19.52.2	Макросы	542
19.52.3	Типы	544
19.52.4	Перечисления	544
19.52.5	Функции	545
19.53	Файл setjmp.h	551
19.53.1	Подробное описание	551
19.53.2	Функции	551
19.54	Файл shell.dox	553
19.55	Файл shell.h	554
19.55.1	Функции	554
19.56	Файл signal.h	556

19.56.1	Подробное описание	558
19.56.2	Макросы	558
19.56.3	Типы	563
19.56.4	Функции	564
19.57	Файл sleep.h	569
19.57.1	Макросы	569
19.57.2	Функции	569
19.58	Файл socket.h	573
19.58.1	Подробное описание	574
19.58.2	Макросы	574
19.58.3	Типы	576
19.58.4	Функции	576
19.59	Файл softgraph.dox	581
19.60	Файл softgraph.h	582
19.60.1	Подробное описание	584
19.60.2	Типы	584
19.60.3	Функции	584
19.61	Файл sound.h	601
19.61.1	Подробное описание	601
19.61.2	Функции	602
19.62	Файл spi.h	606
19.62.1	Подробное описание	606
19.62.2	Макросы	607
19.62.3	Функции	608
19.63	Файл stdarg.h	613
19.63.1	Подробное описание	613
19.63.2	Макросы	613
19.63.3	Типы	614
19.64	Файл stdbool.h	615
19.64.1	Подробное описание	615
19.64.2	Макросы	615
19.65	Файл stddef.h	616
19.65.1	Подробное описание	616
19.65.2	Макросы	616
19.65.3	Типы	616

19.65.4	Функции	617
19.66	Файл <code>stdint.h</code>	618
19.66.1	Подробное описание	619
19.66.2	Макросы	620
19.66.3	Типы	626
19.67	Файл <code>stdio.h</code>	630
19.67.1	Подробное описание	632
19.67.2	Макросы	632
19.67.3	Типы	633
19.67.4	Функции	633
19.67.5	Переменные	652
19.68	Файл <code>stdlib.h</code>	653
19.68.1	Подробное описание	654
19.68.2	Макросы	654
19.68.3	Функции	654
19.69	Файл <code>stdnoreturn.h</code>	669
19.69.1	Подробное описание	669
19.69.2	Макросы	669
19.70	Файл <code>string.h</code>	670
19.70.1	Подробное описание	671
19.70.2	Макросы	671
19.70.3	Функции	671
19.71	Файл <code>tasklib.h</code>	693
19.71.1	Подробное описание	694
19.71.2	Макросы	695
19.71.3	Типы	696
19.71.4	Перечисления	696
19.71.5	Функции	697
19.72	Файл <code>terminator.h</code>	707
19.72.1	Функции	707
19.73	Файл <code>time.h</code>	708
19.73.1	Подробное описание	708
19.73.2	Макросы	708
19.73.3	Типы	709
19.73.4	Функции	709

19.74	Файл timer-arm.h	715
19.74.1	Подробное описание	715
19.74.2	Макросы	715
19.74.3	Функции	716
19.75	Файл timer.h	719
19.75.1	Подробное описание	719
19.75.2	Функции	719
19.76	Файл tv-decoder.h	721
19.76.1	Подробное описание	721
19.76.2	Перечисления	722
19.76.3	Функции	724
19.77	Файл tv-receiver.h	729
19.77.1	Подробное описание	729
19.77.2	Функции	730
19.78	Файл uart.h	733
19.78.1	Подробное описание	734
19.78.2	Макросы	734
19.78.3	Перечисления	737
19.78.4	Функции	738
19.79	Файл uchar.h	745
19.79.1	Подробное описание	745
19.79.2	Типы	745
19.80	Файл udr.h	746
19.80.1	Подробное описание	747
19.80.2	Макросы	747
19.80.3	Типы	747
19.80.4	Функции	748
19.81	Файл unicode.h	752
19.81.1	Функции	752
19.82	Файл usb.h	758
19.82.1	Подробное описание	760
19.82.2	Макросы	760
19.82.3	Перечисления	769
19.82.4	Функции	770
19.83	Файл usb_driver.h	774

19.83.1	Подробное описание	774
19.83.2	Функции	774
19.84	Файл <code>usbdescriptors.h</code>	776
19.84.1	Подробное описание	777
19.84.2	Макросы	777
19.85	Файл <code>usbman.dox</code>	785
19.86	Файл <code>vdisk.h</code>	786
19.86.1	Подробное описание	786
19.86.2	Функции	786
19.87	Файл <code>vector.h</code>	787
19.87.1	Подробное описание	787
19.87.2	Функции	787
Предметный указатель		792

1. Общее описание

Данное руководство содержит описания основных концепций, заложенных в основу Операционной Системы Реального Времени *MULTEX-ARM*, а так же полный список вызовов функций.

Версия

5.06

Авторство

© ООО «Сэт Код», 2023

1.1. Операционная система жесткого реального времени MULTEX-ARM

Операционная система жесткого реального времени (ОСРВ) *MULTEX-ARM* предназначена для встраиваемых применений. Основное ее назначение — предоставление пользователю необходимого и достаточного набора функций для проектирования, разработки и функционирования систем реального времени на конкретном аппаратном оборудовании. Особенностью *MULTEX-ARM* является то, что весь пользовательский проект собирается на этапе компиляции на *инструментальной машине* в единый загружаемый образ, который содержит как разрабатываемый пользователем программный код, так и все необходимые для него библиотечные процедуры.

MULTEX-ARM представляет собой набор библиотек, обеспечивающих эффективную многозадачность, а также набор драйверов, обеспечивающих взаимодействие пользовательского программного обеспечения с аппаратурой. Она предназначена для использования на процессорах китайской фирмы *Allwinner*, таких как: **A20**, **A40i**, **H3**, **V3S**. При этом пользовательское программное обеспечение пишется на языке **Си**. Процедуры библиотеки ядра *MULTEX-ARM*, написанные на языках **Си** и **Ассемблер**, обеспечивают эффективную вытесняющую многозадачность с заданием приоритетов для каждой задачи. При этом планировщик задач может работать как в приоритетном режиме, так и в режиме карусельного планирования. Для обеспечения многозадачности и межзадачного взаимодействия библиотека ядра предоставляет пользователю различные семафоры и очереди сообщений. *MULTEX-ARM* использует плоскую модель памяти, причем любой задаче полностью доступно все адресное пространство процессора и все глобальные переменные проекта. Любая Си-процедура может быть запущена, как отдельная задача.

Жесткое реальное время подразумевает гарантированную реакцию на внешние события за фиксированный интервал времени. Для *MULTEX-ARM* это время сравнимо с временем вызова Си-процедуры. Внешними событиями в *MULTEX-ARM* выступают прерывания от системного таймера, от устройств ввода/вывода, от внешних сигналов. При этом возможна настройка приоритетов прерываний и выполнение вложенных прерываний, что позволяет увеличить точность генерации внешних сигналов до десятков наносекунд.

Плоская модель памяти — вся память в *MULTEX-ARM* имеет физические адреса, совпадающие с виртуальными. При этом каждой задаче в многозадачной среде доступно все адресное пространство процессора. Все глобальные переменные и все глобальные имена процедур доступны всем задачам. Это облегчает межзадачное взаимодействие. Кроме того, для обеспечения бесконфликтного взаимодействия задач друг с другом имеются такие системные механизмы, как *семафоры* и *очереди* сообщений. Процедуры организации механизмов *многозадачности*, а также функции создания и управления семафорами и очередями сообщений объединены в системные библиотеки. Пользовательское ПО линкуется совместно с системными библиотеками в монолитный образ, который и исполняется на *целевой платформе* в соответствии с программой пользователя.

MULTEX-ARM предоставляет пользователю широкие возможности по отладке проекта. С помощью командного интерпретатора *Shell* пользователь может вызывать любую глобальную процедуру, набирая ее вызов в синтаксисе языка **Си**. Кроме того, можно просматривать, либо изменять значения любых глобальных переменных по ходу выполнения программы. Возможно также просматривать, либо модифицировать любые области памяти вычислителя. Это можно делать с *инструментальной машины*, подключенного к *целевой платформе* с помощью канала **DEBUG-UART**, либо по каналу **Ethernet**. Кроме этого, пользователь получает возможность просматривать/редактировать любые зоны памяти, а также получать информацию о состоянии задач в многозадачной среде, запускать в ручную новые задачи, менять приоритеты любой запущенной задачи и удалять любые задачи. Пользователь также может с помощью *Shell* получать информацию о состоянии системы ввода/вывода, переназначать стандартный вывод на другие устройства непосредственно во время работы.

С помощью интерпретатора команд *Shell* в **MULTEX-ARM** пользователь может взаимодействовать с дисковой подсистемой. При работе в среде *Shell* пользователь может оперативно просматривать каталоги всех блочных устройств в системе, копировать или удалять файлы, просматривать их содержимое. С помощью встроенного в *Shell* текстового редактора пользователь может создавать или редактировать содержимое имеющихся текстовых файлов на дисках.

Таким образом, перечисленные особенности **MULTEX-ARM** позволяют пользователю обеспечить сравнительно легкие и быстрые пути создания и отладки широкого спектра приложений в таких областях, как, робототехника, медицина, управление сложными станками с ЧПУ, в системах технического зрения, системах дистанционного управления в реальном времени и передачи видео и аудио информации.

См. также

Базовые определения см. в файле *multex.h*.

1.2. Наборы библиотек

Ниже приведены версии и дата выпуска актуальных для текущей версии библиотек с разделением на наборы:

- Базовый набор библиотек:
 - **armkernel** ver.5.06.1847 от 2023-12-01 14:03:55
 - **a64** ver.1.02.0004 от 2023-12-01 14:45:59
 - **sunxi** ver.1.09.1842 от 2023-11-20 21:45:40
 - **enet** ver.1.05.0252 от 2023-12-01 12:27:03
 - **tcp** ver.1.06.0202 от 2023-11-20 21:52:33
- Мультимедиа A20:
 - **a20graph** ver.2.03.0548 от 2023-12-01 12:00:33
 - **asx340** ver.1.03.0038 от 2023-12-01 12:09:51
 - **avi** ver.1.02.0053 от 2023-11-20 20:20:09
 - **cedrus** ver.1.01.0120 от 2023-11-20 20:44:38
 - **csi** ver.1.02.0172 от 2023-12-01 12:20:51
 - **mpeg4decode** ver.1.02.0082 от 2023-12-01 12:32:27
 - **tvd** ver.1.01.0031 от 2023-12-01 12:48:59
- Мультимедиа V3s, H3:
 - **de2** ver.1.04.0948 от 2023-11-20 21:05:23
 - **softgraph** ver.1.06.1187 от 2023-11-20 21:31:55
- Дополнение к мультимедийным библиотекам:
 - **font** ver.1.04.0093 от 2023-11-20 21:19:51
 - **png** ver.1.02.0145 от 2023-12-01 12:39:53
 - **z** ver.1.02.0027 от 2023-12-01 12:59:35

1.3. История версий

- **5.06** — Основой обновления стали новые библиотеки работы с графикой.
 - Изменения в графических библиотеках:
 - * Добавлена библиотека **lib_cedrus** – поддержка аппаратного кодера / декодера **Cedrus**.
 - * Добавлена библиотека **lib_csi** – поддержка аппаратного модуля подключения видео камер по параллельному интерфейсу **BT656**.
 - * Добавлена библиотека работы с аппаратным **TV** декодером **lib_tvd** с поддержкой **PAL** и **NTSC**.
 - * Добавлена библиотека **lib_asx340** – поддержка камер на базе матрицы **ASX340AT**.
 - * Повышена стабильность работы графической библиотеки процессора **A20** – **lib_a20graph**.
 - * Повышена стабильность работы графической библиотеки процессоров **V3s** и **H3** — **lib_softgraph**. Добавлена возможность вывода окружностей с полупрозрачностью.
 - Изменения в **базовом** наборе библиотек:
 - * Повышена стабильность работы контроллера сети **EMAC**.
 - * **gpio.h** - добавлена работа с внешними прерываниями.
 - * **i2c.h** - устранена ошибка при регистровом чтении для **A20**.
 - * Добавлена тестовая функция оценки работы процессора **cpuUsage()**.
 - * Добавлена поддержка **UTF-8** в консоли.
 - * Дополнена документация.
- **5.05** — Основные изменения:
 - Обновление драйверов с поддержкой процессора **Allwinner V3s**.
 - Поддержка вложенных прерываний.
 - Поддержка **True Type** шрифтов во всех библиотеках графики.
- **5.04** — Первая публикация документации.

2. Сборка проекта

Сборку проекта пользователя с использованием библиотек *MULTEX-ARM* рекомендуется производить на *инструментальной машине* под управлением ОС **Debian**, либо **Ubuntu** с использованием *Linaro toolchain*. Для настройки проекта пользователя, включения опций и подключения модулей, используются файлы *config.h* и *Makefile*. Оба файла должны лежать в директории проекта. Подробнее о структуре проекта смотри в *соответствующем* разделе.

Результатом сборки является **бинарный файл**, содержащий скомпилированный проект пользователя собранный вместе с библиотеками ядра *MULTEX-ARM*, пригодный для исполнения в качестве программы на *целевой платформе*. Такой файл вместе с дополнительными файлами проекта должен быть записан на загрузочную *карту памяти целевой платформы*.

Карта памяти — постоянное запоминающее устройство, с которого выполняется загрузка операционной системы на *целевой платформе*. В общем случае в качестве загрузочной используется *карта памяти uSD* вставленная в один из слотов *целевой платформы*. В некоторых случаях это может быть установленная на *целевой платформе* микросхема памяти **NAND** или **eMMC**. Иногда загрузка *целевой платформы* начинается с загрузчика записанного в памяти **NAND** или **eMMC**, который проверяет наличие *карты памяти uSD* и передаёт управление найденному там бинарному файлу. В любом случае под загрузочным будет пониматься постоянное запоминающее устройство, с которого выполняется загрузка и запуск исполняемого бинарного файла, содержащего *MULTEX-ARM* с процедурами пользователя.

Целевая платформа — физическое устройство (плата, вычислитель, контроллер управления и т.п.) на базе одного из поддерживаемых процессоров, для которого выполняется сборка *MULTEX-ARM* вместе с процедурами пользователя.

Инструментальная машина — персональный компьютер с установленным инструментарием для компиляции, сборки и копирования собранных файлов на *целевую платформу* (по сети, либо через интерфейс **UART**).

2.1. Среда сборки

Предполагается, что сборка пользовательского проекта ведётся на *инструментальной машине* под управлением **Debian**. Для сборки проекта под **Windows** рекомендуется использовать виртуальную машину *WSL*.

2.1.1. Подготовка WSL

Этот раздел нужен только пользователям **Windows** для запуска и настройки виртуальной машины **Linux**. Изначально *WSL* не содержит ни одной установленной виртуальной машины. Для сборки проектов рекомендуется установить дистрибутив **Debian**. Для этого из командной строки **Windows** следует выполнить следующую команду:

```
wsl --install -d Debian
```

В процессе установки потребуется создать нового пользователя и задать пароль.

На *инструментальной машине* может быть установлено несколько дистрибутивов **Linux**. Просмотреть список установленных можно с помощью команды:

```
wsl -l -v
```

Удобно использовать дистрибутив **Debian** как дистрибутив по умолчанию. Если это не так, выбрать основной дистрибутив можно с помощью команды:

```
wsl -s Debian
```

Далее предполагается что все действия в ОС **Windows** производятся через консоль **WSL** с запущенной виртуальной машиной **Debian**.

2.1.2. Установка пакетов

Для сборки проектов понадобятся следующие пакеты:

- **make** — утилита работы с файлами;
- **Linaro toolchain** — набор инструментов для компиляции проектов под **ARM**.

Все последующие команды набираются в консоли **Linux** либо в консоли **WSL** при работе в ОС **Windows**. Для установки необходимых пакетов рекомендуется воспользоваться следующей последовательностью команд:

```
sudo apt update
sudo apt install make
sudo dpkg --add-architecture armhf
sudo apt update
sudo apt install g++-arm-linux-gnueabi
sudo apt install build-essential git debootstrap u-boot-tools device-tree-compiler
```

Проверить установку компиляторов можно с помощью следующих команд:

```
arm-linux-gnueabi-gcc --h
arm-linux-gnueabi-gcc --version
```

2.1.3. Настройка переменных окружения

При компиляции проекта используются специальные утилиты, поставляемые вместе с библиотеками **MULTEX-ARM**. При сборке в Linux все пути прописываются в **Makefile** и никаких дополнительных действий предпринимать не нужно. В Windows путь к утилитам удобно прописать в переменную окружения **PATH**. Для сборки на виртуальной машине **WSL**, пути можно прописать в *Переменных среды*. После перезагрузки компьютера виртуальная машина подключит прописанные пути. Проверить переменные окружения **WSL** можно из командной строки с помощью команды:

```
wsl env
```

2.2. Описание структуры проекта пользователя

Частью **Makefile** каждого проекта является файл **multex.mk** поставляемый вместе с библиотеками **MULTEX-ARM**, в котором задана предполагаемая структура проекта пользователя. Данная структура

может быть изменена, но в этом разделе будет описана структура проекта заданная по умолчанию. Подкаталоги проекта создаются автоматически (если ещё не созданы) при первом запуске сборки проекта.

Новые проекты рекомендуется размещать в директориях, создаваемых на том же уровне, на котором расположена папка с библиотеками **multex_arm**. В новую папку проекта следует скопировать файлы *config.h* и *Makefile*. Эти файлы можно скачать на сайте set-code.ru в составе демонстрационных примеров для различных плат, либо создать самостоятельно по описаниям приведённым ниже.

Рекомендованная структура проекта выглядит следующим образом:

- **multex_arm** — Директория с файлами *MULTEX-ARM*.
 - **bin** — Утилиты сборки.
 - **include** — Заголовочные файлы.
 - **lib** — Библиотеки.
- **myProject** — Директория проекта пользователя.
 - **src** — Исходные тексты. В этой папке следует размещать все компилируемые и заголовочные файлы проекта. Все файлы из этой директории будут скомпилированы и присоединены к проекту. Допускается создание вложенных директорий и поддиректорий (уровень вложенности не ограничен).
 - **out** — Директория готовых бинарников и вообще всего, что должно быть скопировано на *карту памяти целевой платформы* (см. *Запуск на целевой платформе*).
 - *config.h* — Файл конфигурации работы ядра.
 - *Makefile* — Файл настройки сборки.

2.3. Файл конфигурации config.h

config.h — это файл, содержащий набор макросов, используемых ядром *MULTEX-ARM* для настройки аппаратных и программных модулей. В данном разделе описаны макросы такого файла.

2.3.1. Выбор платформы

Для начала необходимо указать целевой процессор с помощью макроса **ARCH_PROC**. Значение следует выбирать из соответствующей группы *макросов*. Например, для процессора **V3s** в файле *config.h* следует записать следующую строку:

```
#define ARCH_PROC ARCH_PROC_V3S
```

2.3.2. Сопроцессор

Для того, чтобы иметь возможность использовать в проекте инструкции сопроцессора **NEON**, необходимо указать макрос:

```
#define INCLUDE_NEON
```

2.3.3. Кэш память

Для задействования внутреннего **кэша** процессора задается макрос:

```
#define DCACHE_ENABLE
```

Так как при отключении **кэша** быстродействие процессора существенно снижается, не рекомендуется использовать проекты с отключенным макросом разрешения **кэша**. Так, работа с сетью будет приводить к ошибкам при сильной нагрузке на нее. Поэтому работа в таком режиме желательна только в отладочных целях.

2.3.4. Отладочная консоль

Для подключения отладочной консоли по последовательному интерфейсу **UART** следует указать макрос:

```
#define INCLUDE_SIO_CONSOLE
```

2.3.5. Файловая система ОС MS-DOS

Для подключения в проекте файловой системы ОС **MS-DOS** следует использовать макрос:

```
#define INCLUDE_DOSFS
```

2.3.6. Сеть

Если в проекте предусматривается использование сети **Ethernet**, то нужно указать макрос:

```
#define INCLUDE_NETINET
```

Кроме того, необходимо задать **IP**-адрес *целевой платформы*, например:

```
#define IP_ADDRESS "10.0.3.27"
```

Для активации стека протоколов **TCP/IP** и библиотеки сетевых сокетов следует указать:

```
#define INCLUDE_TCP
```

Если планируется взаимодействие с *целевой платформой* через консоль по протоколу **UDP**, следует указать:

```
#define INCLUDE_NET_CONSOLE
```

Если же взаимодействие необходимо по протоколу **TCP/IP**, то следует указать:

```
#define INCLUDE_TCP_CONSOLE
```

Если в проекте необходимо использование **FTP**-сервера (для копирования файлов, или быстрой замены версии), следует указать:

```
#define FTP_SERVER
```

2.3.7. Процедуры пользователя

В файле *config.h* есть возможность указать две процедуры пользователя, которые будут вызваны на различных этапах загрузки *MULTEX-ARM*.

Первая из них может быть вызвана в середине загрузки — сразу после загрузки ядра. Такая процедура может быть использована, например, для вывода логотипа на дисплей (если такой имеется на *целевой платформе*). После вывода логотипа загрузка будет продолжена, что визуально может сократить время реакции системы на включение питания. Такое поведение актуально для процессоров с низкой производительностью. Для указания имени процедуры исполняемой после загрузки ядра следует записать макрос вида:

```
#define DRAW_LOGO usrDraw
```

Вторая процедура запускается после окончания всех программных модулей. Например, для запуска такой процедуры с именем **mainProc()** следует записать:

```
#define USER_PROC mainProc
```

Если макрос **USER_PROC** не указан, то после запуска системы будет вызван *Интерпретатор команд SHELL*.

2.3.8. Пример написания config.h

Ниже приведён пример файла конфигурации для проекта пользователя на базе процессора **V3s**, который может использоваться как основа для файлов конфигурации пользователя.

```
\#ifndef _CONFIG_H_
\#define _CONFIG_H_

\#define PROJECT_BRIEF "{Multex-ARM Project V3s}"
```

```
\#define PROJECT_VERSION_NUMBER      1
\#define PROJECT_VERSION_SUB_NUMBER  0

\#include <arch/archdef.h>
\#define ARCH_PROC ARCH_PROC_V3S

\#define INCLUDE_NEON
\#define DCACHE_ENABLE
\#define INCLUDE_SIO_CONSOLE
\#define INCLUDE_DOSFS
\#define INCLUDE_NETINET

\#ifdef INCLUDE_NETINET
    \#define IP_ADDRESS "{10.0.3.35}"
    \#define CHECK_PRIMARY_IP_ADDRESS
    \#define INCLUDE_NETLOADER
    \#define INCLUDE_TCP
    \#ifdef DEBUG
        \#define FTP_SERVER
        \#define FTP_ALLOW_TO_CHANGE_WORK_DRIVE
        \#define INCLUDE_NET_CONSOLE
        \#define INCLUDE_TCP_CONSOLE
    \#endif
\#endif

\#define DRAW_LOGO startScreen
\#define USER_PROC startProject

\#endif
```

2.4. Конфигурация аппаратной части

Конфигурация аппаратной части — это текстовое описание *целевой платформы* составленное пользователем в определённом формате и предназначенное для настройки библиотечных модулей при запуске ядра *MULTEX-ARM*. Такое описание содержит в себе название процессора, название платы, описание подключения используемых периферийных устройств. Описание составляется пользователем в текстовом виде и размещается в файле с расширением *arc* на *карте памяти целевой платформы*. В начале загрузки ядра *MULTEX-ARM* преобразует текстовый файл в список параметров.

Список параметров конфигурации — набор пар *ключ – значение* созданный ядром *MULTEX-ARM* на этапе загрузки на базе текстового описания конфигурации аппаратной части. Созданный системой список параметров доступен для проекта пользователя только на чтение и добавление данных и может быть дополнен в исходном коде пользователя с помощью функций описанных в *arch.h*. Для описания аппаратной части используются зарезервированные строки-ключи описанные в файле *archdef.h*.

См. также

Функции работы с описанием аппаратной части, а также зарезервированные строки-ключи в файлах *arch.h* и *archdef.h*.

Описание аппаратной части строится на базе списков программного модуля *mapstr.h*. Структура *списка параметров* не является жёсткой в отличие от структур и перечислений стандарта языка **Си**. В процессе развития операционной системы и добавления новых полей параметров структура описания аппаратной части не будет нарушена и все библиотеки будут иметь доступ к используемым ими полям *списка параметров* без необходимости пересборки.



Каждая запись в описании занимает **512** байт ОЗУ. При необходимости в итоговом проекте занимаемую память можно освободить с помощью `archFree()` после инициализации всех библиотек. Учитывая такую политику использования, при разработке новых библиотек **рекомендуется** забирать нужные для работы значения из *списка параметров* в функции инициализации и не обращаться к *списку параметров* после её завершения.

Пример чтения *списка параметров* конфигурации — блок настройки аппаратных модулей характерных для каждого процессора:

```
bool ok;
const char *cpu = archGetString (ARCH_CPU, \&ok);

if (ok \&\& archCheckString (cpu, ARCH_PROC_V3S)) {
    printf ("{V3s seetup...\n"});
    // Some actions for V3s ...
} else

if (ok \&\& archCheckString (cpu, ARCH_PROC_A40)) {
    printf ("{A40 seetup...\n"});
    // Some actions for A40 ...
}
```

2.4.1. Приоритет записей в списке параметров конфигурации

Записи в созданном ядром *MULTEX-ARM* *списке параметров* могут дублироваться, так как заполняются системой из разных источников. Поиск параметров по ключу в списке выполняется с начала списка до первого совпадения. Следовательно, записи в начале *списка параметров* имеют более высокий приоритет. Загрузка списка выполняется ядром системы с учётом этой особенности – вначале грузится список из файла с расширением **arc** на *карте памяти* (если таковой имеется). Для плат, название которых содержится в файле *archdef.h*, *список параметров* может содержать только название платы, остальные параметры конфигурации для таких плат будут подгружены автоматически из ядра системы. Далее подгружаются данные о типе процессора и затем данные из библиотек, основанные на типе процессора. Итоговый список выводится в консоль в сборке *debug*. Если какие-то из загруженных параметров нужно изменить их следует внести в *файл конфигурации* (файл с расширением **arc** на *карте памяти*).

2.4.2. Файл конфигурации

Файл описания конфигурации аппаратной части может размещаться на одном из дисков, монтируемых при старте системы (**uSD, eMMC, NAND**). Название файла не имеет значения, так как поиск файла производится системой по расширению **arc**. Все найденные файлы с таким расширением будут загружены в общий список до инициализации основных модулей системы. В таком файле рекомендуется указать как минимум название платы, выбрав его из зарезервированных строк-ключей файла *arch.h*. Остальные параметры будут подгружены позже при старте программных модулей, на основании выбранной платы. Также в файле можно указать некоторые параметры конфигурации, используемые программными модулями. Так как файлы конфигурации загружаются до инициализации модулей — такие записи будут иметь более высокий приоритет, чем записи, вносимые библиотеками. Это позволяет изменять параметры конфигурации, заложенные в библиотеках. Для новых (ещё не поддерживаемых) плат возможно составить полное описание аппаратной части с помощью такого файла.

Файл описания является текстовым, где каждая строка является одной записью. Каждая запись состоит из набора полей, разделённых точкой с запятой. Некоторые поля могут содержать набор

значений разделённых запятой. В файле могут содержаться закомментированные строки, начинающиеся со знака решётки. Первая строка файла конфигурации обязательно должна содержать подпись **#MAPSTORE**. Стандартный набор полей одной записи описан ниже.

Набор полей файла описания конфигурации:

- **Параметр** — зарезервированная строка-ключ, по которой будет осуществляться поиск значений. Может содержать любые символы латинского алфавита, кроме пробела (пробелы будут удалены при поиске). Все известные строки-ключи описаны в файле [arch.h](#).
- **Тип данных** — служит для правильной интерпретации значений при поиске. Может содержать следующие зарезервированные строки:
 - **STR** — строковое значение;
 - **INT** — набор десятичных значений (частный случай - одно значение).
- **Значение** — для строковых полей это зарезервированная строка, описанная в [arch.h](#). Для десятичных значений это набор, состоящий как минимум из одного значения.
- **Описание** — необязательный параметр, служащий для лучшего понимания десятичных значений при просмотре файла.

Пример файла конфигурации приведён в листинге ниже. Для примера взята плата **SE8351-00**, описание аппаратной части которой уже содержится в ядре **MULTEX-ARM** (см. [ARCH_BOARD_SE8350_00](#) в файле [archdef.h](#)). Описание такой платы может быть сгенерировано автоматически, а значит достаточно указать её название. Остальные параметры будут подгружены системой в список после загрузки файла. Параметр **backlight-pwm** (выбор канала ШИМ для подсветки дисплея) показан для примера. Но, при желании, можно изменить этот параметр на единицу, чтобы перенаправить управление подсветкой на другой вывод процессора.



Некоторые параметры конфигурации используются не во всех проектах. При этом система производит поиск всех системных параметров и выводит в консоль предупреждения о параметрах, которые не были найдены. Такие системные параметры конфигурации, как линии системных светодиодов (см. [ARCH_LED_SYSTEM](#) и [ARCH_LED_DISK](#)), можно указывать с пустым полем **value**, чтобы убрать предупреждения из лога загрузки, выводимого в консоль.

```
#MAPSTORE
#   parameter; type;      value;  desc.
#-----
board-name; STR; SE8351-00;
backlight-pwm; INT;      0;    PWM0
led-system; STR;
led-disk; STR;
```

2.5. Файл сборки Makefile

Makefile — это файл, содержащий набор инструкций, используемых утилитой **make** в инструментарии автоматизации сборки. В данном разделе описаны параметры такого файла, используемого при сборке **MULTEX-ARM**.

При создании директории нового проекта в неё следует скопировать уже имеющийся **Makefile** из аналогичного проекта, например из одного из примеров на сайте [set-code.ru](#). Либо данный файл можно составить самостоятельно. Пример готового **Makefile** приведён в [конце раздела](#). Основная (универсальная для всех проектов) часть инструкций функции и цели сборки, находится

в файле `include/all/multex.mk`, поставляемом вместе с библиотеками *MULTEX-ARM*. Этот файл следует включить в *Makefile* проекта после определения всех переменных.

```
include $(MULTEX_PATH)/include/all/multex.mk
```

Остальные инструкции, уникальные для каждого проекта, следует записать в *Makefile* самостоятельно, либо изменить уже имеющиеся. Ниже приведено описание используемых в *Makefile* уникальных инструкций.

2.5.1. Имя выходного файла

Результатом сборки проекта является исполняемый бинарный файл, либо библиотека. Имя собираемого файла можно настроить. Указывать имя файла следует без расширения. Расширение будет добавлено в зависимости от выбранной цели сборки. Для проектов пользователя рекомендуемое имя — **multex**, так как именно такое имя указано по умолчанию в используемых загрузчиках. Для определения имени выходного файла следует записать инструкцию:

```
PROJ_NAME = multex
```

2.5.2. Определение флагов компилятора

Компиляция файлов исходных кодов осуществляется с оптимальным набором флагов компилятора. Пользовательские флаги компилятора могут быть добавлены при необходимости с помощью переменной **USR_CFLAGS**:

```
USR_CFLAGS =
```

2.5.3. Распределение памяти

В *Makefile* выполняется управление распределением памяти ОЗУ. При сборке итогового проекта линковщику передаётся значение адреса, по которому будет размещена запуская процедура *MULTEX-ARM*. Данное значение размещается в переменной **MX_TEXT**. Это значение должно совпадать со значением записанным в загрузчике в качестве адреса запуска. Кроме того, в здесь же определяются значения адресов начала и конца ОЗУ. Они записываются в переменные **DRAM_START** и **MEM_POOL_END** соответственно и используются самой операционной системой для корректного выделения памяти. Ниже приведены адреса, записываемые в *Makefile* по умолчанию:

```
DRAM_START    = 0x40000000  
MX_TEXT       = 0x48000000  
MEM_POOL_END  = 0x80000000
```

Переменные **DRAM_START** и **MX_TEXT** одинаковы для большинства поддерживаемых плат и их значения можно не указывать в *Makefile* проекта. Значение адреса **MEM_POOL_END** соответствует 1 Гб используемого ОЗУ и должно быть изменено, если на плате установлена память меньшего объёма. Максимальные значения переменной **MEM_POOL_END** для разных объёмов памяти приведены ниже:

- 1 Гб — 0x80000000
- 512 Мб — 0x60000000
- 256 Мб — 0x50000000
- 64 Мб — 0x44000000

2.5.4. Настройка путей

В разделе *Описание структуры проекта пользователя* описана структура проекта по умолчанию. Такое взаимное расположение директорий учтено в файле `include/all/multex.mk` и если придерживаться структуры директорий по умолчанию, то дополнительная настройка путей не потребуется. Однако в некоторых случаях структуру проекта можно изменить или дополнить. Такие изменения взаимного расположения директорий проекта и *MULTEX-ARM* можно сделать с помощью инструкций описанных в этом разделе.

Путь к **текущей версии** *MULTEX-ARM* можно задать с помощью следующей инструкции:

```
MULTEX_PATH = ../multex_arm
```

Список **исходных файлов** проекта и директорий, содержащих исходные файлы можно изменить или дополнить с помощью инструкции **SRC_PATHS**. В процессе компиляции и сборки проекта утилита **make** пройдёт по указанным директориям и всем вложенным в неё и соберёт все исходные файлы. По умолчанию подключается одна папка **src**, лежащая в корне проекта. Если директории по умолчанию не существует, то она будет создана при первом запуске сборки. Все дополнительные файлы и папки следует добавить в переменную **SRC_PATHS**. Если используется директория по умолчанию — переменную добавлять не нужно. Пример определения пути к исходным файлам проекта:

```
SRC_PATHS = ./src
```

Путь к **собираемому бинарному** файлу или библиотеке следует задать, если он отличается от значения по умолчанию **out**. Указанная папка будет создана при сборке, если ещё не создана. Если используется директория по умолчанию — переменную добавлять не нужно. Пример определения пути к собираемому бинарному файлу:

```
OUT_PATH = ./out
```

В переменную **INCLUDES** по умолчанию помещается путь к **заголовочным файлам** *MULTEX-ARM*. Если в проекте используются дополнительные папки с заголовочными файлами их следует добавить к этой переменной, например:

```
INCLUDES += ../my_includes
```

При сборке **библиотеки** имеет смысл указать **заголовочные файлы**, которые будут скопированы в `multex_arm/include`. При этом сама библиотека копируется в `multex_arm/lib`. Например, для копирования двух заголовочных файлов библиотеки шрифтов можно записать:

```
OUT_HEADERS += src/fonts.h
OUT_HEADERS += src/fontsdefines.h
```

Для размещения заголовочных файлов копируемых библиотек в подкаталоге директории **multex_arm/include** следует указать имя подкаталога в переменной **OUT_HEADERS_PATH**. Иначе заголовочные файлы библиотек будут скопированы непосредственно в **multex_arm/include**. Например для копирования заголовочных файлов библиотеки в папку `multex_arm/include/multimedia` следует записать:

```
OUT_HEADERS_PATH = multimedia
```

2.5.5. Настройка подключаемых библиотек

К каждому проекту при сборке подключается набор библиотек. Часть библиотек (например, библиотеки ядра **MULTEX-ARM**) подключаются неявно, остальные подключаемые библиотеки нужно указывать в *Makefile* с помощью инструкции **LIBRARIES**. Все файлы библиотек **MULTEX-ARM** по умолчанию находятся в папке **multex_arm/lib**. Библиотеки пользователя рекомендуется помещать сюда же. Сборка ядра **MULTEX-ARM** возможна с минимальным набором библиотек. Их набор может варьироваться в зависимости от использования в проекте различных аппаратных модулей. Рекомендации по подключению конкретных файлов библиотек содержатся в описаниях подсистем операционной системы. Например, для подключения библиотеки аппаратной поддержки графики с поддержкой формата **PNG** следует указать:

```
LIBRARIES += -l_a20graph
LIBRARIES += -l_png -l_z
```

2.5.6. Подключение примеров и тестовых функций

В комплект поставки **MULTEX-ARM** входят файлы исходных кодов, содержащих примеры использования различных библиотек операционной системы. Для подключения этих функций к собираемому бинарному файлу следует добавить путь к одной из папок с примерами в переменную **SRC_PATHS**. Например, подключить примеры для процессора **V3s** можно с помощью следующей записи:

```
SRC_PATHS += $(MULTEX_PATH)/include/examples/v3s
```

Все подключенные тестовые функции можно вызывать из консоли *Shell*.



При сборке реального проекта данную строчку следует закомментировать, чтобы тестовые функции не вошли в состав итогового бинарного файла.

2.5.7. Пример написания Makefile

Ниже приведён пример простого *Makefile* для процессора **V3s**, который можно использовать в качестве основы для проектов пользователя:

```
#-----  
# Makefile сборки проектов и библиотек  
# для запуска Multex-ARM на процессоре V3s  
# © 000 «Сэт Код», 2023 (set-code.ru)  
#-----  
PROJ_NAME = multex  
MX_TEXT = 0x41000000  
MEM_POOL_END = 0x44000000  
MULTEX_PATH = ../multex_arm  
LIBRARIES += -l_enet -l_tcp  
include $(MULTEX_PATH)/include/all/multex.mk
```

2.6. Сборка проекта (библиотеки)

Файлы проекта вместе с библиотеками и ядром *MULTEX-ARM* собираются с помощью утилиты **make** из директории проекта. В результате сборки в проекте появится папка (по умолчанию **out**) с бинарным файлом (по умолчанию **multex.bin**), который следует скопировать на запоминающее устройство *целевой платформы*. Это и есть запускаемый файл проекта.

Сборка проекта осуществляется с помощью *Makefile* поставляемого вместе с библиотеками. В файле уже имеются специализированные цели сборки проектов, библиотек и объектных файлов. Ниже описаны основные цели сборки, имеющиеся в предоставляемом *Makefile*.

2.6.1. Вызов справки

Краткую помощь по целям сборки можно получить с помощью цели:

```
make help
```

2.6.2. Отладочная сборка

Для сборки отладочной версии проекта используется цель **debug**. В итоге такой сборки получается версия бинарного файла содержащая таблицу символов. При этом появляется возможность вызывать функции по имени из консоли. Кроме того, при компиляции файлов определяется макрос **DEBUG**, который можно использовать для отладочного вывода. Для отладочной сборки следует использовать цель:

```
make debug
```

2.6.3. Сборка поставочной версии

Для сборки поставочной версии проекта используется цель **release**. В такой версии не собирается таблица символов и определяется макрос **RELEASE**. Для сборки поставочной версии следует использовать цель:

```
make release
```

2.6.4. Сборка библиотек

Для сборки библиотеки и копирования её вместе с указанными заголовочными файлам в директорию *MULTEX-ARM* следует использовать цель:

```
make lib
```

Если пересборка библиотеки не нужна а нужно только скопировать итоговый бинарный файл библиотеки вместе с заголовочными файлами в директорию *MULTEX-ARM*, то можно использовать цель:

```
make copy
```

2.6.5. Прочие цели сборки

Очистка проекта от временных и объектных файлов выполняется с помощью цели:

```
make clean
```

Также реализована возможность компилировать объектные файлы из исходных по имени. Например для файла filename.c команда компиляции будет выглядеть так:

```
make filename.o
```

2.7. Запуск на целевой платформе

Для запуска собранного проекта на *целевой платформе* следует:

- скопировать файлы проекта на загрузочную *карту памяти*;
- запустить *целевую платформу* с использованием этой *карты памяти*.

Для копирования файлов на *карту памяти uSD* через кардридер *инструментальной машины* можно воспользоваться целями сборки *Makefile*. Для копирования всех файлов из папки **out**:

```
make install
```

Для обновления только собранного бинарного файла:

```
make update
```



Для копирования файлов на загрузочную *карту памяти uSD* средствами *Makefile* следует предварительно указать переменную окружения **DEVNAME**, определяющую имя устройства в системе, на которое будет произведено копирование. Копирование файлов производится на первый том *карты памяти*. Монтирование нужного тома будет произведено средствами *Makefile*.

```
export DEVNAME=sda
```

Кроме того, бинарный файл можно заменить по сети на уже работающей *целевой платформе*. Для этого на ней должен быть запущен **FTP-сервер**. За запуск сервера отвечает параметр *FTP_SERVER* файла *конфигурации*. При подключении к серверу будет доступна файловая система *целевой платформы* и бинарный файл можно заменить стандартными командами **FTP**.

Параметры для соединения с запущенным **FTP-сервером**:

- Адрес — параметр, указанный в *IP_ADDRESS* файла *конфигурации*.
- Имя учётной записи — **anonymous**.
- Пароль — **gremlin**.
- Режим обмена — **пассивный**.

3. Интерпретатор команд SHELL

В состав библиотек ОС *MULTEX-ARM* включен интерпретатор команд **Shell** – специальная программная оболочка, которая позволяет осуществлять следующие действия:

- Вызывать по именам любую процедуру из проекта пользователя *MULTEX-ARM*.
- Просматривать или изменять содержимое любой глобальной переменной проекта по ее имени в реальном времени.
- Просматривать дампы памяти в виде байтов, двухбайтовых слов, или четырёх-байтовых двойных слов. Можно также записывать новые значения содержимого ячеек памяти по любому адресу.
- Работать с дисковой подсистемой с помощью следующих команд:
 - Просмотр каталогов.
 - Смена рабочего каталога, либо диска.
 - Создание / удаление каталогов и файлов.
- Просматривать перечень всех запущенных в системе задач.
- Получить список открытых файлов.
- Получить список установленных устройств.
- Просматривать таблицы **ARP** состояния локальной сети.

Shell запускается автоматически, если при *конфигурации* проекта не указана задача, которую необходимо запускать при старте системы, либо в качестве параметра *USER_PROC* указана процедура **shell**. Связь с терминалом инструментальной машины происходит при этом по каналу **RS232C**. Параметры подключения:

- Выходной интерфейс процессора — **DEBUG UART0**.
- Скорость — **115200** bps.
- Проверка чётности — **отсутствует**.
- Количество бит данных — **8**.
- Количество стоп бит — **1**.

Если в файле *конфигурации* системы указан параметр *INCLUDE_NET_CONSOLE*, то отдельный экземпляр интерпретатора команд будет запускаться при подключении к целевой машине терминала по локальной сети по протоколу **UDP**, а также, при включении параметра *INCLUDE_TCP_CONSOLE*, будет создаваться по одному экземпляру на каждую сессию при подключении по протоколу **TCP/IP**. Параметры подключения:

- **IP**-адрес — параметр, указанный в *IP_ADDRESS* файла *конфигурации*.
- Порт — **23**.

В качестве терминала для инструментальной машины удобнее всего использовать такую программу, как **Putty**.

После запуска **Shell** на экране инструментальной **ЦВМ** появится приветственная надпись,

3.2. Справка работы с диском

Чтобы получить справку о командах работы с диском достаточно набрать команду **hf**:

```
C: />hf
----- FILE UTILITES -----
dir <path>           - show directory
dir* <path>         - show directory paged
cd <path>           - change work device/directory
mkd <path>          - create new directory
rmd <path>          - delete directory
copy <from> <to>    - copy files by wildcard
del <files>         - delete files by wildcard
type <file>         - show text file
ed <file>           - edit text file
chkdsk <device>    - testing filesystem on device
```

3.3. Дамп памяти

Чтобы получить дамп памяти, начиная с указанного адреса, достаточно использовать команды **d**, **dw**, или **dd**. После команды через пробел нужно указать адрес в шестнадцатеричной форме:

```
C: />d 120
00000120: 06 4A 08 B5 12 1A 02 F0 5D FF 00 21 DF F8 10 90
00000130: 08 46 00 F0 59 FF 00 BF 00 00 F8 4F 50 01 F8 4F
00000140: D0 49 00 00 08 B5 05 46 04 4B DC 68 00 F0 CD F8
00000150: 2A 46 41 F2 BB 01 00 20 A0 47 00 BF 00 00 F8 4F
00000160: 00 20 70 47 08 B5 07 48 01 F0 0E F8 4C F2 50 30
00000170: 02 F0 BE FF FF F7 F4 FF 00 20 00 F0 34 F9 00 20
00000180: 08 BD 00 BF EF 43 00 00 82 B0 00 23 01 93 01 9B
00000190: 63 2B 03 DC 00 46 01 9B 01 33 F7 E7 02 B0 70 47
000001A0: 10 B5 FF F7 F1 FF 11 EE 10 4F FF F7 ED FF 24 F4
000001B0: 80 54 01 EE 10 4F 10 BD 10 B5 FF F7 E5 FF 11 EE
```

3.4. Модифицировать дамп памяти

Для того, чтобы модифицировать содержимое памяти начиная с указанного адреса, достаточно использовать команды **m**, **mw**, или **md**. В качестве параметра нужно указать адрес, с которого нужно начинать модификацию:

```
C: />m 123
Addr:00000123 Data:B5 -
```

Далее нужно ввести новое значение для ячейки памяти в шестнадцатеричной форме и нажать **Enter**. Если данную ячейку изменять не требуется, следует просто нажать **Enter**. Новое значение запишется и произойдет переход на следующую ячейку:

```
Addr:00000123 Data:B5 B7
Addr:00000124 Data:12 -
```

Чтобы выйти из режима вместо данных нужно ввести символ **Пробел** и нажать **Enter**.

3.5. Просмотр задач

Для просмотра сведений об имеющихся в системе задачах и их состоянии используется команда **i**:

```
C: />i
Name      Priority   Id      Delay Semaphore  State
-----
SHELL     50        7FFDF600  0      00000000    IN WORK
RootTask  255       7FFFFC00  0      00000000    ACTIVE
DPCMan    -1        7FFEE700  -1     7FFEF200    PEND
NetPoll   0         7DA26E00  -1     7DA27500    PEND
tsAcker   0         7D94E200  1      7D94E500    PEND
FTP-S     1         7DA00600  -1     7D93DB00    PEND
netShell  5         7DA11D00  -1     7DA16700    PEND
netConTx  5         7DA15000  71     7DA15E00    PEND
-----
C: />
```

В отображаемой таблице будут указаны задачи с указанием их имен, приоритетов, идентификаторов, задержек, семафоров, у которых задачи ожидают и состояния. Текущая выполняемая задача будет иметь состояние **IN WORK**. Активные, но в данный момент менее приоритетные задачи будут иметь состояние **ACTIVE**. Задержанные на какое-то время, или ожидающие у семафора задачи будут иметь состояние **PEND**. Приостановленные задачи будут иметь состояние **SUSPEND**.

3.6. Открытые файлы, сокеты, устройства

Для просмотра информации об открытых файлах, сокетах и символьных устройствах можно ввести команду **iosFdShow**:

```
C: />iosFdShow
fd name      drv
 3 sioCon     0 in out err
 4 netCon    2
 5 socket/<tcp> 3
Value = 1610613075 (0x60000153)
C: />
```

Устройство, являющееся стандартным устройством ввода / вывода, отмечено как **in**, **out** и **err**.

3.7. Установленные устройства

Для просмотра информации об установленных в системе устройствах можно использовать команду **iosDevShow**:

```
C: />iosDevShow
DeviceName Driver  DCB
socket      3      00000000
netCon      2      00000000
C:          1      7FFCD100
```

```
sioCon      0      7FFCF100
Value = 4 (0x4)
C: />
```

В этом примере видно, что в системе имеется **TCP/IP** сокет, открытый на прослушивание **FTP**-соединений, **netCon** — сетевая консоль для связи по протоколу **UDP**, локальный диск **C:** и последовательная консоль **sioCon**.

3.8. Информация о сети

Для просмотра информации о состоянии сети **Ethernet** можно воспользоваться командой **netShow**:

```
C: />netShow
Table of Address Resolution:
-----
1 #0 00:1B:EB:61:6C:7E 10.0.0.222
2 #0 10:7B:44:45:85:2B 10.0.7.156
3 #0 00:1B:EB:61:6C:7E 10.0.0.90
4 #0 BC:EE:7B:71:FB:57 10.0.7.119
-----
Value = 0 (0x0)
C: />
```

Здесь видно активные **IP**-адреса в сети. При этом в таблице показаны порядковый номер, номер сетевого адаптера, **MAC**-адрес и **IP**-адрес каждой записи из таблицы **ARP**.

3.9. Удалить задачу

Для того, чтобы удалить указанную задачу, используется команда **td**. В качестве параметра указывается имя запущенной задачи, которую требуется удалить:

```
C: />td myTask1
Ok!
C: />
```

3.10. Изменить приоритет задачи

Для изменения приоритета указанной задачи используется команда **tp**. При этом первым параметром указывается имя задачи с учетом регистра, а вторым — новое значение приоритета:

```
C: />tp myTask2 100
Ok!
C: />
```

В указанном примере задача *myTask2* получает приоритет *100*.

3.11. Запустить задачу

Чтобы запустить некоторую процедуру языка **Си**, как отдельную задачу, можно воспользоваться командой **sp**. В качестве параметра необходимо указать имя этой процедуры, но перед именем необходимо указать символ **&**:

```
C: />sp &myProc
Task spawned as usrT1 , TID=7D53A000
```

В этом примере процедура *myProc* запускается, как задача с именем *usrT1* и приоритетом *100*.

3.12. Выполнить процедуру

Любая функция, включенная в пользовательский проект и объявленная как глобальная, может быть вызвана из интерпретатора команд путем ввода ее имени с использованием синтаксиса, принятого в языке **Си**. Так, например, для вывода сообщения достаточно просто набрать:

```
C: />printf("s %i",Hello!,38)
Hello! 38
C: />
```

3.13. Аппаратная перезагрузка

Для вызова перезагрузки (аппаратного рестарта) системы служит команда **hr**.

3.14. Работа с переменными

3.14.1. Просмотр переменной

Для просмотра содержимого переменной типа **int**, входящей в проект, достаточно просто набрать ее имя:

```
C: />myVar
Addr = 0x4904082C Value = 0 (0x0)
C: />
```

3.14.2. Изменение переменной в десятичном виде

Чтобы изменить содержимое переменной типа **int**, достаточно записать выражение присваивания языка **Си**:

```
C: />myVar = 1234
Addr = 0x4904082C New value = 1234 (0x4D2)
C: />
```

3.14.3. Изменение переменной в шестнадцатеричном виде

Если требуется ввести значение в шестнадцатеричном виде, то можно записать его так, как это и принято в языке **Си**. При этом буквенные обозначения цифр можно набирать как на верхнем (A,B,C), так и на нижнем (a,b,c) регистре:

```
C: />myVar = 0x1A2C3D4E
Addr = 0x4904082C New value = 439106894 (0x1A2C3D4E)
C: />
```

3.14.4. Изменение строковой переменной

Для ввода значения строковой переменной, оно записывается в кавычках, как и принято в языке **Си**:

```
C: />string = "Hello world"
Addr = 0x490609DC New value = 2102453504 (0x7D50E500)
C: />printf(string)
Hello world
Value = 1610613075 (0x60000153)
C: />
```

3.15. Работа с диском

3.15.1. Просмотр каталога

Для просмотра текущего каталога используется команда **dir**:

```
C: />dir
Volume in drive C: is A20
Directory of 'C:/'
SVS          <DIR>    1/01/2000    0:16:32
BOOT        SCR      151 30/09/2014   13:36:18
LICENSE     SYS       40  1/01/2017   12:10:38
SETTINGS    INI       23  1/01/2017   12:02:00
TST         INI       29  1/01/2000    0:05:24
T           TTT       53  1/01/2000    0:17:16
PB          MP3 3731445 1/01/2000    0:06:54
POEZD      MP3 3545427 1/01/2017   12:04:52
ZATEH      MP3 4681866 1/01/2017   12:24:00
MULTEX     BIN 221832 20/06/2022   11:50:24
Total 12180866 bytes in 9 files, 1 directories
Free space = 4164944 Kbytes.
C: />
```

Для просмотра содержимого подкаталога нужно указать путь к подкаталогу. Например, для просмотра подкаталога **SVS** нужно дать команду **dir C:/SVS**.

```
C: />dir C:/SVS
```

```
Volume in drive C: is A20
Directory of 'C:/SVS/'
.           <DIR>    1/01/2000   0:16:32
..          <DIR>    1/01/2000   0:16:32
Total 0 bytes in 0 files, 2 directories
Free space = 4164944 Kbytes.
C:/>
```

3.15.2. Смена каталога / диска

Для того, чтобы поменять текущий диск, нужно просто ввести в качестве команды его метку. Так, если подключен **USB Flash-диск**, можно переключиться на него:

```
C:/>F:
F:/>
```

Для того, чтобы выбрать текущим каталог, отличный от корневого, используется команда команду **cd**:

```
C:/>cd c:/svs
C:/SVS/>
```

3.15.3. Удаление файла

Для удаления файла, или группы файлов, используйте команду **del**. В следующем примере с диска удаляются все файлы с расширением **.mp3**:

```
C:/>del c:/*.mp3
Delete file 'C:/ZATEH.MP3'...done.
Delete file 'C:/POEZD.MP3'...done.
Delete file 'C:/PB.MP3'...done.
C:/>
```

3.15.4. Копирование файла

Для копирования файла, или группы файлов с одного места в другое, используется команда **copy**:

```
C:/>copy c:/multex.bin f:/*.*
Copy from 'C:/MULTEX.BIN' to 'F:/MULTEX.BIN'...done.
1 file(s) copied.
C:/>
```

3.15.5. Удаление каталога

Для удаления пустого каталога используется команда **rmdir**:

```
C: />rmdir C:/SVS  
C: />
```



Попытка удаления непустого каталога приведет к ошибке.

3.15.6. Создание каталога

Для создания нового каталога используется команда **mkdir**:

```
C: />mkdir C:/SVS  
C: />
```

3.15.7. Проверка диска

Для проверки файловой системы на диске служит команда **chkdsk**:

```
C: />chkdsk  
Checking device C:...  
File System: FAT32  
Check FAT...done  
Clusters:  
Total: 1044224, Reserved: 0, Free: 1044149  
Sectors per cluster: 8  
Check file system...  
Total Files:17, Data:8, Cat's:2, Labels:1 LongNames:6  
Total Deleted Files:67  
Result of checking: OK!
```

3.16. Встроенный текстовый редактор

В **Shell** имеется встроенный текстовый редактор. Редактор позволяет просматривать и редактировать текстовые файлы из консоли.

3.16.1. Просмотр файла

Для просмотра содержимого текстовых файлов используется команда **type**:

```
C: />type C:/settings.ini  
[SCREEN]  
LIGHT=100  
C: />
```

3.16.2. Редактирование файла

Для редактирования небольших текстовых файлов нужно войти в режим редактирования с помощью команды **ed**:

```
C: />ed c:/settings.ini
Edit file c:/settings.ini
L [begin] [end] - list
A                - add new string at end
I [line]         - insert new string
D [line]         - delete string
E [line]         - edit string
W                - write file
Q                - exit without saving
H                - help
-----
0000 | [SCREEN]
0001 | LIGHT=100
ed>
```

Более полное описание работы во встроенном текстовом редакторе доступно в режиме редактирования по команде **help**.

3.17. Поддерживаемые сочетания клавиш

В интерпретатор команд **Shell** встроена поддержка сочетаний клавиш для редактирования вводимого текста и перемещения по вводимой строке.

3.17.1. История команд

- **Ctrl+P** или **Up** – перейти в истории команд на 1 команду назад (к предыдущей команде).
- **Ctrl+N** или **Down** – перейти в истории команд на 1 команду вперед (к следующей команде).

3.17.2. Навигация

- **Ctrl+B** или **Left** – переместить курсор на 1 символ влево.
- **Ctrl+F** или **Right** – переместить курсор на 1 символ вправо.
- **Ctrl+A** или **Home** – перейти в начало строки.
- **Ctrl+E** или **End** – перейти в конец строки.
- **Alt+F** – переместить курсор вперед к следующему слову.
- **Alt+B** – переместить курсор назад к предыдущему слову.

3.17.3. Редактирование

- **Insert** – переключение режима "вставки" на "замену" и обратно.
- **Ctrl+D** или **Delete** – удалить 1 символ справа от текущей позиции.
- **Ctrl+H** или **Backspace** – удалить 1 символ слева от текущей позиции.
- **Alt+D** – удалить все символы "слова" от текущей позиции до правого конца.
- **Ctrl+W** – удалить все символы "слова" от текущей позиции до левого конца.
- **Ctrl+K** – удалить все символы команды от текущей позиции до правого конца.
- **Ctrl+U** – удалить все символы команды от текущей позиции до левого конца.

4. Ядро операционной системы

4.1. Многозадачность и межзадачное взаимодействие

См. также

Описание методов работы с задачами и **примеры кода** в файле *tasklib.h*.

Особенностью построения систем, реализующих программную обработку в реальном времени, является необходимость одновременного выполнения нескольких процедур, причем каждая из них выполняется по своему алгоритму и синхронизируется тем или иным внешним событием (момент поступления данных от устройства ввода/вывода, срабатывание таймера, аппаратное прерывание от внешнего источника, готовность внешней системы к получению данных и т.д.) При этом часто бывает необходимо обеспечить минимальное время реакции программы на внешнее событие. Ситуация усложняется тем, что внешние события могут возникать независимо друг от друга в произвольные моменты времени.

Ядро *MULTEX-ARM* позволяет достаточно легко создавать такие системы и обеспечивает гибкое взаимодействие параллельно выполняющихся процессов с минимальными накладными расходами. Многозадачная среда *MULTEX-ARM* позволяет представлять прикладной проект в виде набора независимых задач и обеспечить кажущуюся одновременность их выполнения. В зависимости от общих требований прикладной проект может быть построен следующими способами (либо их комбинацией):

- Обработка внешних событий занимает сравнительно небольшое время. Все обработчики событий имеют равные приоритеты, инициируются наступлением события, выполняют требуемую обработку и переводятся в состояние ожидания события. В такой системе одновременное наступление нескольких событий приводит к тому, что их обработчики выстраиваются в очередь на выполнение и время реакции на событие может случайным образом изменяться в достаточно больших пределах.
- Как и в предыдущем варианте, обработчики инициируются наступлением событий, но имеют разные приоритеты. Критичные ко времени обработчики имеют более высокий приоритет, поэтому могут выполняться на фоне не критичных (или менее критичных).
- Все обработчики имеют равные приоритеты, но ядро *MULTEX-ARM* переводится в режим карусельного планирования, при котором все активные задачи чередуются циклически, причем каждой из них отводится квант времени, по истечении которого производится переключение на следующую по порядку задачу.
- Ядро находится в режиме карусельного планирования, задачи разбиваются на группы с одинаковыми приоритетами. В этом случае циклическое переключение задач производится только между задачами из высокоприоритетной группы. После того, как в этой группе не останется ни одной активной задачи, начнется выполнение задач из следующей, более низкоприоритетной группы и т.д.

Недостатком режима карусельного планирования является зависимость эффективного быстродействия (а значит и времени выполнения для отдельной задачи) от комбинации состояний остальных задач системы. Однако, *MULTEX-ARM* позволяет динамически изменять режим работы ядра – включать или выключать карусельный режим, а также менять квант времени исполнения задачи, либо полностью блокировать переключение задач на время выполнения критичной ко времени секции процедуры. Кроме того, имеется возможность динамической смены приоритета любой задачи. Благодаря этому в среде *MULTEX-ARM* возможно эффективное построение самых разнообразных прикладных проектов.

Любая C-подпрограмма в среде *MULTEX-ARM* может быть запущена как отдельная задача со своим собственным контекстом и стеком. Базовые средства *MULTEX-ARM* позволяют **создавать, приостанавливать, возобновлять, задерживать и уничтожать** задачи. Задачи могут инициировать внутренние события, на которые будут реагировать другие задачи. Каждая задача *MULTEX-ARM* может находиться в одном из следующих состояний:

- Выполняемая задача **IN WORK** — та задача, которая выполняется процессором в данный момент времени.
- Активная задача **ACTIVE** — задача, готовая к выполнению, но ожидающая своей очереди, так как в данный момент выполняется другая задача с таким же или более высоким приоритетом.
- Задержанная задача **DELAYED** — задача, выполнение которой отложено на заданное время.
- Приостановленная задача **PEND** — задача, ждущая у семафора.
- Остановленная задача **SUSPEND** — задача, переведенная в пассивное состояние командой `taskSuspend()`.

Ядро **MULTEX-ARM** манипулирует задачами с помощью специальных двусвязных циклических динамических списков – каруселей. При инициализации ядра создаются две таких структуры – карусель активных задач и карусель задержанных задач. Каждая задача системы помещается в одну из них, за исключением выполняемой задачи, которая до очередного переключения не отнесена ни к одной из каруселей. Существует еще одна, скрытая задача, которая выполняет пустой “вечный цикл” и служит только для того, чтобы выполняться тогда, когда все остальные задачи приостановлены. В эту задачу вырывается процедура запуска ядра **MULTEX-ARM** после завершения выполнения функции инициализации `kernelInit()`. Процесс переключения с задачи на задачу в **MULTEX-ARM** сводится к следующим шагам:

- Из карусели активных задач выталкивается очередная задача.
- Выполняемая задача вталкивается в карусель активных или задержанных задач (в зависимости от причины, вызвавшей переключение), причем вталкивание может происходить в соответствии с приоритетом задачи, или в режиме **FIFO**.
- Ядро **MULTEX-ARM** производит смену контекста задачи, обеспечивающую возобновление выполнения новой задачи и сохранение состояния старой. При этом происходит замена контекста процессора и зоны стека. Для задач, использующих сопроцессор, помимо этого производится сохранение / восстановление контекста сопроцессора.

Настройка приоритетов аппаратных *прерываний* в комплексе с назначением приоритетов выполняемых задач позволяет реализовать достаточно гибкую и эффективную многозадачную среду на программно-аппаратном уровне. Так, чистое время переключения задач сравнимо с временем вызова “пустой” процедуры языка **C**. Однако, следует иметь в виду, что большое количество задач в системе может привести к увеличению накладных расходов на манипуляции с каруселями.

4.1.1. Управление прерываниями

См. также

Описание методов управления прерываниями в файле `intlib.h`.

Прерывания играют важнейшую роль в системах реального времени. Аппаратное прерывание, вызванное поступлением электрического импульса на один из специализированных входов контроллера является сигналом к немедленным действиям, которые должна выполнить система в ответ на это событие. Ядро **MULTEX-ARM** позволяет обеспечить незамедлительное выполнение при возникновении прерывания пользовательской процедуры, подключенной к этому прерыванию. Такая процедура называется **обработчиком прерывания**. Все действия по сохранению / восстановлению контекста прерываемой задачи берет на себя операционная система. Ядро **MULTEX-ARM** настраивает контроллер прерываний таким образом, чтобы отличать аппаратные прерывания от исключений процессора и особых случаев, которые могут возникать при выполнении программы.

4.1.2. Приоритеты прерываний и задач

При планировании распределения вычислительных ресурсов в проекте следует учитывать иерархию выполнения аппаратных прерываний и пользовательских задач и настраивать их приоритеты соответствующим образом. В *MULTEX-ARM* существует два вида приоритетов. Их описание дано ниже.

4.1.2.1. Приоритеты прерываний В *ARM* процессорах со встроенным контроллером прерываний *GIC* реализована поддержка групп и приоритетов прерываний. Прерывания объединяются в группы внутри которых распределяются приоритеты обработки прерываний. Прерывания более приоритетной группы могут вытеснять (прерывать) обработку менее приоритетных групп. Приоритет внутри группы позволяет получить преимущество при возникновении нескольких прерываний одновременно.

Такая политика обработки прерываний поддержана в *MULTEX-ARM*. Для использования доступно **4** группы аппаратных прерываний и **8** уровней приоритетов. Макросы, описывающие порядок распределения прерываний по группам и назначения приоритетов, собраны в отдельную *группу*. Основной группой для всех прерываний является *INTERRUPT_GROUP_MAJOR*. Большинство прерываний используемых системой включены именно сюда. В эту же группу рекомендуется добавлять пользовательские обработчики прерываний, подключаемые с помощью *interruptConnect()*. Более приоритетной группой является *INTERRUPT_GROUP_SYSTEM*. В неё обычно входит всего одно прерывание — *Системный таймер*. В группу с наивысшим приоритетом *INTERRUPT_GROUP_TIME_CRITICAL* рекомендуется помещать прерывания, обеспечивающие мгновенную реакцию на событие, либо строго задающие период следования импульсов на внешних линиях. Обработчики таких прерываний должны выполняться максимально быстро. Например, задействовав *TIMER_2* и аппаратный модуль ШИМ в импульсном режиме можно получить сигнал с переменной скважностью и жёстко заданным периодом. Вид такого сигнала с периодом 1,5 мкс приведён на верхнем графике *осциллограммы*. На нижнем графике показан тестовый импульс сгенерированный системным таймером. В данном случае высокоприоритетное прерывание, запускающее каждый импульс ШИМ, 3 раза прерывает обработку системного таймера. Наивысший приоритет в данном случае позволяет точно выдержать заданную частоту

следования импульсов.

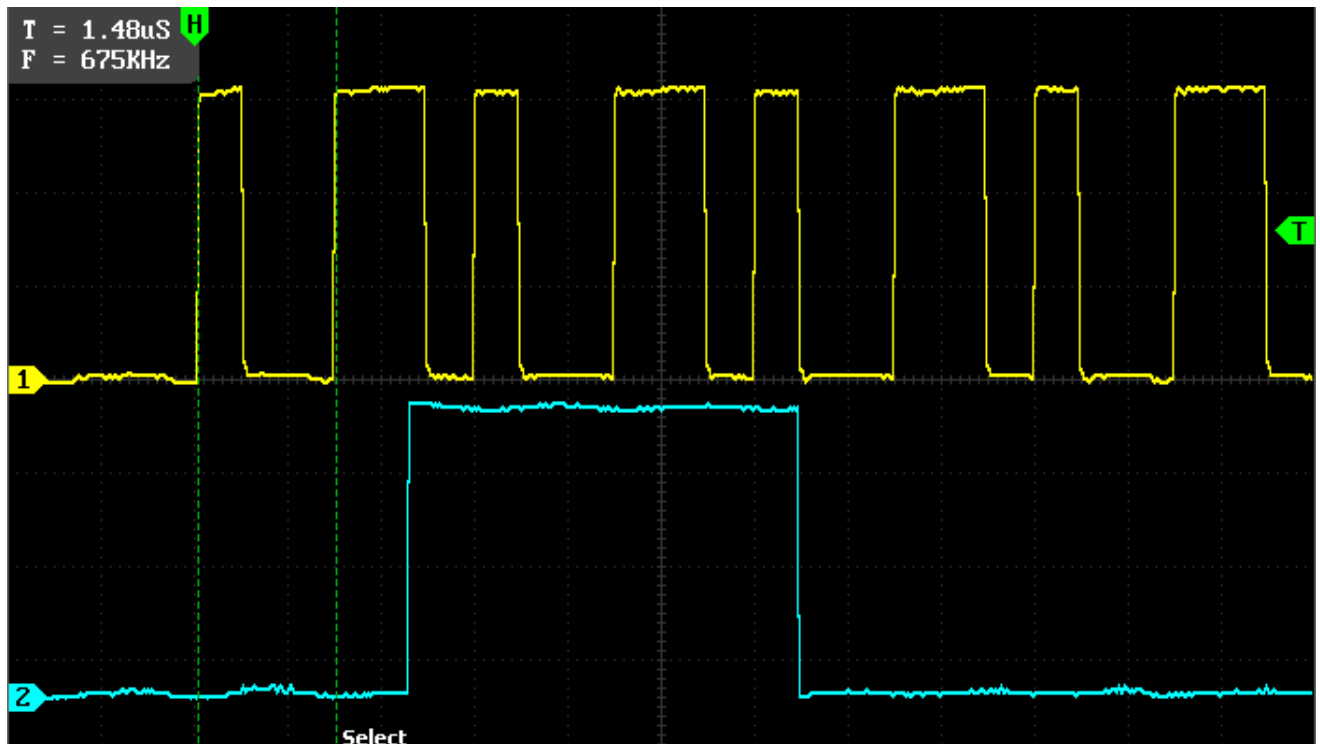


Рисунок 2. Вложенные прерывания

4.1.2.2. Приоритеты задач Выделением процессорного времени для каждой задачи занимается *Системный таймер* реализованный на аппаратном *TIMER_1*. На каждом тике таймера принимается решение какая из задач будет выполняться в настоящий момент. Каждый раз предпочтение отдаётся наиболее приоритетной из активных задач. В то же время не следует путать приоритеты задач и приоритеты прерываний. Во время исполнения даже самой приоритетной задачи может возникнуть аппаратное прерывание, обработчик которого прервёт текущую задачу.

4.1.3. Таймеры

См. также

Описание методов работы с таймерами в файле *timer-arm.h*.

В зависимости от модели процессора в нём может быть реализовано от 3 до 6 аппаратных таймеров. В *MULTEX-ARM* принято распределение таймеров, описанное соответствующей группе *макросов*. Как правило таймеры *TIMER_0* и *TIMER_1* заняты системой. Остальные таймеры могут быть задействованы под конкретный проект.

4.1.3.1. Системный таймер

См. также

Описание методов работы с системным таймером в файле *timer.h*.

Системный таймер (*TIMER_1*) используется ядром *MULTEX-ARM* для обеспечения синхронизации внутренних функций, регистрации таймаутов и организации режима карусельного планирования. Частота системного таймера устанавливается при инициализации ядра равной 1000Гц. Однако она может быть изменена пользовательской задачей на любое целое значение из диапазона

20Гц-1000 Гц. Так как все временные интервалы (таймауты или задержки) задаются в тиках таймера, то при использовании констант они будут изменяться при перестройке частоты таймера. Рекомендуется в значение задержки включать функцию опроса частоты таймера, для получения задержек, независящих от настройки таймера.

4.1.4. Сигналы

См. также

Описание методов работы с сигналами в файле [signal.h](#).

Сигналы — это ограниченная форма *межзадачного* взаимодействия. Сигналом называется асинхронное уведомление, отосланное задаче для того, чтобы сообщить ей о каком-то событии.

В библиотеках ядра *MULTEX-ARM* имеются средства для поддержки сигналов, совместимых с **UNIX**-подобными операционными системами. Стандарт **Си** определяет всего шесть сигналов, которые могут быть обработаны. Все они описаны в группе *Стандартные для Си сигналы* файла [signal.h](#).

Кроме того, есть сигналы, которые не могут быть обработаны, пойманы или игнорированы, такие как *SIGKILL* и *SIGSTOP*.

4.1.5. Семафоры

См. также

Описание методов работы семафорами и **примеры кода** в файле [semlib.h](#).

Семафоры в *MULTEX-ARM* – это основной механизм синхронизации задач в реальном времени и организации взаимоисключающего доступа задач к общим ресурсам.



Семафоры это механизм именно межзадачного взаимодействия. Не следует использовать их для синхронизации задач с аппаратными прерываниями!

Ядро *MULTEX-ARM* поддерживает три типа семафоров:

- Двоичный семафор – служит для синхронизации задач или организации взаимоисключающего доступа.
- Целочисленный семафор – служит для защиты множественных ресурсов.
- Семафор взаимного исключения – служит для наследования приоритета, защиты от удаления и использования рекурсии.

Для семафоров *MULTEX-ARM* реализован единый универсальный интерфейс управления. Функции управления семафорами могут применяться к семафору любого типа, различаются только функции создания семафоров. Функция создания семафора возвращает идентификатор семафора, который используется для ссылок на этот семафор из других функций. При создании семафора указывается тип очереди, в которую будут выстраиваться задачи, ждущие у этого семафора. Очередь может быть двух типов: в порядке приоритета задач (опция *SEM_Q_PRIORITY*) или в порядке поступления (*SEM_Q_FIFO*).

4.1.6. Очереди сообщений

См. также

Описание методов работы с очередями сообщений и **примеры кода** в файле *msgQLib.h*.

Очереди сообщений – удобный механизм межзадачного взаимодействия. С помощью очередей сообщений одна задача может передавать данные другой, не заботясь о синхронизации моментов выдачи и получения данных.



Очереди сообщений это механизм именно межзадачного взаимодействия. Не следует использовать их для обмена данных между задачами и обработчиками аппаратных прерываниями! Для этого служит другой механизм, описанный в *ringbuffer.h*.

Передаваемые через очереди сообщений данные должны быть объединены в общую группу – **сообщение**. При этом структура сообщения может быть произвольной, достаточно только, чтобы одно сообщение занимало непрерывный участок памяти, имело предсказуемый размер и его структура была известна как передающей, так и принимающей задаче. Очередь автоматически буферизует заданное при ее создании количество сообщений, поэтому, если принимающая задача вовремя не опросит очередь, то сообщение не будет потеряно. Если задача пытается получить данные из пустой очереди, то она может быть задержана до появления в очереди сообщения. Впрочем, задача может ждать сообщение заданное время или вовсе не ждать. Таким образом, очередь сообщений выполняет не только функции транспортировки данных, но и функции синхронизации задач в реальном времени. И помещать сообщения в очередь, и извлекать их из нее может одновременно несколько задач. Внутренние механизмы очередей обеспечивают корректный множественный доступ к ее данным. Для ссылок на очередь из разных задач используется идентификатор очереди *MSG_Q_ID*, возвращаемый при ее создании. Помещать сообщения в очередь задача может двумя способами:

- В конец очереди - *MSG_PRI_NORMAL*.
- В начало очереди – *MSG_PRI_URGENT*.

Таким образом, обеспечивается возможность передачи двух типов сообщений – обычных и внеочередных. Задачи, ожидающие получения сообщений из очереди, могут получать данные в порядке их обращений к очереди, или в соответствии с их приоритетами. Режим задается при создании очереди опциями *MSG_Q_FIFO* и *MSG_Q_PRIORITY* соответственно.

См. также

Описание заголовочных файлов в группе *Ядро MULTEX-ARM*.

4.2. Базовая система ввода / вывода

Ядро *MULTEX-ARM* поддерживает драйверы символьных (последовательного доступа) и блочных устройств ввода / вывода. Используя простые соглашения, пользователь может создавать свои драйверы символьных устройств и подключать их в систему самостоятельно. При этом устройство становится доступным как по его дескриптору – небольшому целому числу, так и по символьному имени, задаваемому устройству при регистрации его драйвера в системе. Устройства с дескрипторами 0, 1 и 2 являются стандартными устройствами ввода, вывода и вывода ошибок (*STDIN*, *STDOUT* и *STDERR*). Каждая задача в своем блоке управления *TCB* имеет соответствующие поля – *s_in*, *s_out* и *s_err*. При создании корневой задачи в эти поля заносятся значения *STDIN*, *STDOUT* и *STDERR*. Функции *scanf()*, *printf()* и *printerr()* осуществляют ввод-вывод на эти устройства. В таблице дескрипторов для стандартных устройств указываются ссылки на их адреса, поэтому возможно перенаправление ввода / вывода как глобально, для всех задач, так и отдельно, для конкретной задачи. В базовой системе ввода / вывода *MULTEX-ARM* используются две таблицы:

- **sysDrvTable** – таблица для занесения параметров драйверов устройств ввода / вывода.

- **sysFdTable** – таблица для занесения параметров открытых файлов.

Эти таблицы создаются функцией *kernelInit()*, причем размеры таблиц задаются следующим образом: максимальное число драйверов в системе — 100, а максимальное число открытых файлов — 256.

См. также

Описание методов работы с базовой системой ввода / вывода см. в файле *iolib.h*.

4.2.1. Блочные устройства и файловые системы

В ядре *MULTEX-ARM* поддерживается работа с блочными устройствами, такими как **SD** карты, **USB** флэш накопители, **SATA** устройства. Эти устройства оперируют данными, как блоками фиксированной длины, отсюда и название. Драйверы таких устройств должны иметь как минимум две обязательные функции:

- Функцию **чтения** из устройства заданного количества блоков в буфер в оперативной памяти.
- Функцию записи на устройство из буфера заданного количества блоков.

При создании блочного устройства его драйвер заносит указатели на эти функции в специальную структуру *BLK_DEV*. Так, например, при создании устройства типа **MMC** в ядре *MULTEX-ARM* выполняются следующие действия:

```
Dev = malloc(sizeof(BLK_DEV)); // Выделение памяти под блок управления
memset(Dev, 0, sizeof(BLK_DEV)); // Очистка выделенной области памяти
Dev->signature=BD_STD_SIGNATURE; // Установка правильной сигнатуры
Dev->volume=0x80; // Значение для жесткого диска
Dev->startBlk=0; // Читаем блоки с начала MMC
Dev->blkTotal=2; // Для начала достаточно 2 блоков, потом скорр.
Dev->blkRd = mmcRdBlk; // Процедура чтения
Dev->blkWrt = mmcWrBlk; // Процедура записи
Dev->volConfig = mmc; // Данные для драйвера
res = mmc_init(mmc); // Аппаратная настройка MMC
```

Далее, для того, чтобы получить доступ к файлам, хранящимся на устройстве, требуется создать еще одну специальную структуру — файловую систему. В ядре *MULTEX-ARM* создается такая структура — файловая система **MSDOS FAT12 / FAT16 / FAT32**. Файловые системы создаются вызовом процедуры *iosDrvInstall()* точно так же, как создаются символьные устройства, описанные ранее. Дескриптор файловой системы **FAT12 / FAT16 / FAT32** уже создан ядром при запуске и находится в глобальной переменной *DosDriver*.

Чтобы привязать конкретное блочное устройство к файловой системе в *MULTEX-ARM* достаточно вызвать функцию монтирования *mountVolume()*.

4.3. Диспетчер памяти

См. также

Описание методов управления памятью см. в файле *memlib.h*.

Ядро *MULTEX-ARM* включает комплекс процедур, обеспечивающих работу с динамически выделяемыми блоками памяти (**Heap-memory**). Этот комплекс, называемый диспетчером динамической памяти, позволяет задачам ядра и пользовательским задачам выделять для

собственных нужд блоки памяти требуемого размера и освобождать их по мере надобности. Функции диспетчера памяти используются ядром *MULTEX-ARM* при создании задач, семафоров и очередей.

Особенностью реализации диспетчера памяти в *MULTEX-ARM* является принцип выделения блоков памяти от старших адресов к младшим. При этом наибольший свободный сегмент памяти всегда оказывается первым в цепочке. Это очень важно для повышения быстродействия системы. Диспетчер обеспечивает защиту от сегментации памяти – при высвобождении блоков производится слияние смежных свободных блоков.

Для обеспечения защиты от множественного доступа к диспетчеру памяти принято наиболее простое и эффективное решение:

- Так как диспетчер памяти не может быть защищен семафорами взаимного исключения (сама работа с семафорами нуждается в функциях диспетчера памяти), то на время цикла выделения/высвобождения блока памяти процессор закрывается, что гарантирует непрерываемость операции.

4.4. Работа с встроенной КЭШ-памятью процессора

См. также

Описание методов работы с КЭШ-памятью см. в файле *cache.h*.

Встроенная КЭШ-память процессора обеспечивает существенное повышение быстродействия вычислительной системы. Ядро *MULTEX* при запуске размещает в памяти таблицу виртуализации памяти и активирует *MMU* (модуль управления памятью) в соответствии с этой таблицей. При этом для каждой страницы размером в **1М** память может быть настроена как некешируемая, кешируемая с записью, или кешируемая с отложенной записью. Для этого в составе *BSP* для конкретного модуля назначается специальная структура *SYS_MEM_MAP*, описывающая то, как нужно настраивать память. Так как *MULTEX* работает с использованием кэш, то в некоторых случаях возможно рассогласование между содержимым памяти и данными в кэш. Это бывает, например, когда какой-либо блок заносит данные в память прямым доступом.

4.5. Нелокальные переходы

См. также

Описание функций нелокальных переходов в файлах *setjmp.h* и *stdlib.h*.

В ядре *MULTEX-ARM* содержится библиотека стандартной поддержки нелокальных переходов *setjmp*. Библиотека объявляет тип данных *jmp_buf*, который является массивом и который может использоваться для сохранения и восстановления контекста выполнения программы. Для установки точки сохранения служит функция *setjmp()*. Вернуться в точку сохранения можно с помощью функции *longjmp()*. Для сохранения и восстановления контекста при переключении между задачами служат функции *setexit()* и *exit()*.

4.6. Работа с ini-файлами

См. также

Описание методов работы с ini-файлами в файле *inifiles.h*.

Часто пользователю приходится конфигурировать конкретную систему, целевое **ПО**, работающее в среде *MULTEX-ARM*, производя его настройку с помощью каких-либо переменных. Эти переменные определяют общее поведение системы. Их значение не должно теряться при отключении питания, а напротив — при включении питания и запуске системы

конфигурационные переменные должны принимать те значения, которые были заданы при конфигурировании. С другой стороны, при проведении программных настроек пользователь должен иметь возможность изменять значения каких-либо конфигурационных переменных таким образом, чтобы после повторных включений системы вновь присвоенные значения не терялись. Для этих целей в *MULTEX-ARM* разработан единый механизм хранения (и модификации при необходимости) данных в специальных *ini*-файлах, по аналогии с тем, как это было сделано в ранних версиях **ОС Windows**. *Ini*-файл - это обыкновенный текстовый файл, который может быть подготовлен как средствами *MULTEX-ARM*, так и обыкновенным текстовым редактором, а затем записан на диск, доступный операционной системе. Для работы с такими файлами разработана специальная библиотека, входящая в состав ядра *MULTEX-ARM*.

Эта библиотека позволяет пользователю отыскивать в *ini*-файле значения переменных по их именам. При этом значения могут представляться как в виде текстовых строк, так и в десятичном виде, либо в виде булевых значений (типа да-нет). Кроме этого, ряд функций позволяет записывать новые значения в этот файл по имени переменной. После закрытия файла на диске *MULTEX-ARM* будет создан новый *ini*-файл со всеми внесенными изменениями и дополнениями. Старый *ini*-файл будет при этом удален.

В *ini*-файле переменные сгруппированы в секции. Каждая секция также имеет уникальное имя. Для того, чтобы отыскать, или записать требуемое значение, необходимо задать имя секции и имя переменной. При чтении также задается значение по умолчанию, а при записи — то значение, которое требуется присвоить.

4.7. Работа с межпроцессорными каналами.

4.8. PLL – распределение тактовых частот

См. также

Описание методов работы с **PLL** в файле *pll.h*.

Опорная частота 24 МГц, получаемая с помощью внутренней схемы генерации и стабилизируемая с помощью внешнего кварцевого резонатора, повышается внутри процессора и далее распределяется по внутренним шинам и аппаратным модулям. Как именно происходит распределение частот зависит от конфигурации конкретного процессора. Управление умножением и делением частот и распределение их между шинами и модулями осуществляется с помощью регистров конфигурации **PLL**.

Функции по настройке **PLL** регистров полностью берёт на себя операционная система. Однако при проектировании новых драйверов устройств, а так же для проверки правильной работы уже подключенных аппаратных модулей имеет смысл воспользоваться функциями, описанными в файле *pll.h*. Для вывода в консоль таблицы используемых тактовых частот можно воспользоваться функцией *pll()*. Для получения значения частоты конкретного аппаратного модуля или шины в программе можно воспользоваться функциями *pllGet()*, *pllAhbGet()* и им подобными. Следует учесть, что драйверы устройств могут изменять частоты общих шин, поэтому для получения актуальных значений следует перед использованием **get-функций** вызвать функцию сбора данных *pllUpdate()*.

5. Аппаратные интерфейсы

5.1. GPIO – Порты ввода/вывода

См. также

Описание методов работы с портами ввода/вывода в файле *gpio.h*.

Для работы с портами общего назначения как с простыми линиями ввода/вывода их следует сконфигурировать — настроить мультиплексор используемой линии, подключив её к регистру чтения или записи. Такая настройка выполняется с помощью функций *gpio_direction_input()* и *gpio_direction_output()*. После настройки уровень сигнала на линии может быть считан с помощью *gpio_get_value()* или выставлен с помощью *gpio_set_value()*. Линии, сконфигурированные как входные, могут быть *подтянуты* к нулю или плюсу питания с помощью функций *gpio_pull_up()* и *gpio_pull_down()*. Выбор конфигурируемой линии осуществляется как сумма имени порта из набора *макросов* и номера линии.

В следующем примере **PE1** настраивается как выход с установкой высокого уровня на линии. Далее уровень изменяется на низкий.

```
gpio_direction_output (P_E+1, 1);  
// ...  
gpio_set_value (P_E+1, 0);
```

Пример настройки **PD5** как линии ввода с *подтяжкой* к плюсу питания.

```
gpio_direction_input (P_D+5);  
gpio_pull_up (P_D+5);
```

Для подключения линий портов ввода/вывода к аппаратным модулям процессора используется функция *gpio_set_mux()*. Эта же функция может использоваться для настройки линий как входных и выходных, так как предоставляет возможность управления мультиплексором в общем виде. Для настройки мультиплексора каждой линии порта используется 4 бита, а следовательно линия может иметь до 16 вариантов подключения. Возможные значения мультиплексора каждой линии описаны в документации на используемый процессор.

Пример подключения линии **PB4** к выходу **PWM0** модуля **ШИМ**. В данном примере используется значение мультиплексора **PWM_MUX**, определённое как **2**.

```
gpio_set_mux (P_B+4, PWM_MUX);
```

5.2. PWM (ШИМ)

См. также

Описание методов работы с линиями вывода ШИМ в файле *pwm.h*.

Поддержка аппаратных модулей **ШИМ** в *MULTEX-ARM* реализована в двух режимах — импульсном и непрерывном. Для каждого режима необходимо сначала вызвать функцию инициализации

аппаратного модуля для выбранного канала. Далее в непрерывном режиме следует указать коэффициент заполнения ШИМ. Изменяя коэффициент заполнения в непрерывном режиме можно, например, управлять яркостью дисплея. В импульсном режиме для запуска каждого импульса следует вызывать запускающую функцию. Подробное описание функций можно найти в файле *pwm.h*.

В различных процессорах для выходных линий ШИМ используются разные пины. *Конфигурация* линий конкретного процессора считывается при старте операционной системы. И далее, при вызове функции инициализации для одного из каналов ШИМ, соответствующий каналу вывод процессора будет настроен должным образом.

5.3. SPI

См. также

Описание методов работы с интерфейсом **SPI** в файле *spi.h*.

Шина **SPI** - последовательный синхронный интерфейс передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого высокоскоростного сопряжения микроконтроллеров и периферии. **SPI** также иногда называют четырёхпроводным (four-wire) интерфейсом.

SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая (ведомая) периферия синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства может присоединяться несколько ведомых. Ведущее устройство выбирает ведомое для передачи, активируя сигнал **CS** (Chip Select) на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участия в передаче по **SPI**. В процессорах *Allwinner* может содержаться до четырёх аппаратных интерфейсов **SPI** и для каждого из них предусмотрено до четырёх сигналов **CS**. Число линий **CS** может быть увеличено путём программного использования линий ввода/вывода общего назначения **GPIO**.

Интерфейс **SPI** использует следующие сигналы:

- **SCLK** – тактовый сигнал (от ведущего к ведомому);
- **MOSI** – данные, передаваемые (от ведущего к ведомому);
- **MISO** – принимаемые данные (от ведомого к ведущему);
- **CSn** – выбор абонента (от ведущего к выбираемому ведомому).

В *MULTEX-ARM* реализован драйвер, позволяющий работать с несколькими интерфейсами **SPI** в режиме ведущего устройства (**MASTER**). Для подключения одного из имеющихся интерфейсов **SPI** в *MULTEX-ARM* нужно инициализировать нужный модуль с помощью *spiInit()* и затем вызывать функцию *spiTransfer()* для запуска цикла обмена данными с выбранным устройством. Для подключения аппаратных линий **CS** следует использовать функцию инициализации *spiInitChipSelectLine()* для каждой используемой линии. При подключении выбранного аппаратного модуля **SPI** будут использованы настройки интерфейса по умолчанию. Для изменения настроек следует воспользоваться функциями из соответствующей *группы*. Изменять данные настройки можно перед каждым циклом обмена данными с устройством. Это может потребоваться, если к одному интерфейсу подключены устройства с разными параметрами передачи данных.

Ниже приведён пример инициализации интерфейса **SPI0** и получение данных от подключенного устройства. При инициализации выбирается основной набор линий ввода/вывода и подключается линия выбора ведомого устройства **CS0**. Тактовая частота **CLK** устанавливается равной 4 МГц. Остальные настройки используются по умолчанию. Далее выполняется чтение из устройства по адресу **0x10** (это частный случай определённый протоколом обмена конкретного устройства). Для этого в первый байт массива данных кладётся нужный адрес и задаётся длина значащих

передаваемых байт равная единице. Остальные передаваемые данные будут иметь нулевое значение. Такое поведение задаётся аппаратным модулем **SPI**. Общее количество обменов задаётся равным трём. После получения первого байта ведомое устройство начнёт передачу данных, таким образом прочитанное значение после окончания обмена будет лежать в последних двух байтах массива.

```
spiInit (SPI_0, SPI_GPIO_DEFAULT);
spiSetClkFrequency (SPI_0, 4000000);
spiInitChipSelectLine (SPI_0, SPI_GPIO_DEFAULT, SPI_CS_0);

// read from 0x10
char data[3];
data[0] = 0x10;
spiTransfer (SPI_0, SPI_CS_0, data, 1, 3);
```

5.4. I2C

См. также

Описание методов работы с интерфейсом **I2C** в файле *i2c.h*.

Интерфейс **I2C** — это широко распространенный последовательный синхронный полудуплексный двухпроводный внутрисхемный интерфейс, используемый для управления многими устройствами, например, **CMOS** видеокамерами, или часами реального времени **RTC**.

В составе встроенных аппаратных модулей **ARM** процессоров *Allwinner* может содержаться до пяти контроллеров последовательных шин **I2C**. В ОС *MULTEX-ARM* для подключения драйвера одного из аппаратных интерфейсов следует вызвать функцию инициализации *i2cInit()* с указанием тактовой частоты сигнала **CLK**. Как правило, устройства на шине **I2C** работают на частотах 100 или 400 кГц и в этих случаях для выбора частоты шины можно воспользоваться значениями из группы *макросов*. На самом деле при инициализации можно использовать и другие частоты работы шины. В этом случае частоту следует указать в виде числа в герцах. Далее можно читать и писать в подключенные по данному интерфейсу устройства по адресу с помощью функций *i2cRead()* и *i2cWrite()*. Адрес устройства должен быть указан в документации на само устройство.

Ниже приведён пример инициализации драйвера интерфейса I2C0 и запись массива данных в устройство по адресу 0x10.

```
char data[5];
i2cInit (I2C_0, I2C_GPIO_DEFAULT, I2C_CLK_400_kHz);
// ...
i2cWrite (I2C_0, 0x10, data, sizeof (data));
```

5.5. UART

См. также

Описание методов работы с последовательным портом **UART** в файле *uart.h*.

В состав *MULTEX-ARM* включен драйвер, обеспечивающий работу с последовательными портами **UART** в режиме потоковой записи / чтения. После настройки порт следует открыть с помощью функции *open()*, которая вернет дескриптор выбранного устройства. Используя этот дескриптор, записывать и читать данные можно стандартными средствами ОС – функциями *read()* и *write()*.

Ниже приведён пример использования драйвера для приёма и пересылки обратно принятых данных. В начале инициализируется выбранный для передачи аппаратный модуль **UART2**. При инициализации он получает стандартные настройки, некоторые из которых бывает необходимо изменить. В примере изменяется скорость передачи данных на 38400 бит в секунду. Так же имеет смысл изменить таймаут на приём данных, чтобы задача периодически получала управление при отсутствии данных. Таймаут на добавление данных в очередь отправки можно сделать нулевым (*NO_WAIT*), так как переполнений выходного буфера не предвидится. Размер буферов приёма и отправки можно было бы оставить без изменений, так как при создании каждому из них выделяется по 4096 байт, но на деле достаточно выделить под буферы память просто большую, чем ожидаемый размер пакетов. В примере буферы получают размер вдвое больший, чем нужно для приёма и передачи данных.

После инициализации аппаратного модуля происходит открытие устройства с помощью *open()*. В качестве имени устройства следует использовать функцию получения имени *uartName()*. После успешного открытия порта данные из порта можно читать порциями с помощью *read()* с указанием полученного дескриптора. Если данных достаточно они будут считаны сразу. Если данных не достаточно функция будет ждать накопления данных до истечения указанного при инициализации таймаута. По истечении заданного времени функция вернёт количество прочитанных байт. Если же указан таймаут *WAIT_FOREVER* – функция может не возвращать управление задаче разбора пакетов до накопления нужных данных. Отправку данных следует производить с помощью функции *write()* с указанием полученного при открытии дескриптора.

```
int maxLen = 32;
char data[maxLen];
uartInit (UART_2, UART_GPIO_DEFAULT);
uartSetBaudRate (UART_2, UART_BAUD_38400);
uartSetReadTimeout (UART_2, 100);
uartSetWriteTimeout (UART_2, NO_WAIT);
uartSetWriteBufferSize (UART_2, maxLen*2);
uartSetReadBufferSize (UART_2, maxLen*2);

int uartFd = open (uartName (UART_2), O_RDWR);
while (true) {
    int len = read (uartFd, data, maxLen);
    if (len)
        write (uartFd, data, len);
    else
        printf ("{no data\n"});
}
```

6. Аппаратная поддержка мультимедиа

См. также

Описание заголовочных файлов в группе [Мультимедиа](#).

6.1. Графическая подсистема

Для подключения графической подсистемы к проекту необходимо добавить соответствующую библиотеку в [Makefile](#):

```
LIBRARIES += -l_a20graph
```

При использовании файлов в формате **PNG** понадобятся также дополнительные библиотеки:

```
LIBRARIES += -l_png -l_z
```

См. также

Описание методов работы с графической подсистемой в файле [a20graph.h](#).

6.1.1. Общее описание

В составе библиотек **RTOS MULTEX-ARM** имеется библиотека **A20Graph** — библиотека аппаратной поддержки **2D**-графики для процессора **Allwinner A20**. Эта библиотека позволяет:

- Выводить графику на дисплей с помощью интерфейсов **HDMI** и **LVDS**, а также параллельный **RGB LCD**, **VGA** и **TVOut** в различных форматах вплоть до **FULL HD** (1080P). Это позволяет подключить к вычислителю практически любой монитор или **LCD LVDS** панель.
- Отображать на экране графику в режиме **RGB888** или **RGB565** по 4 или по 2 байта на пиксель.
- Работать с двумя областями памяти – **SCREEN** и **CONSTR**, отображаемой в данный момент на экране и скрытой, конструируемой в тот же момент с возможностью быстрого переключения **SCREEN/CONSTR**.
- Ожидать завершения отрисовки кадра перед переключением для устранения возникновения строба на экране при быстро изменяющихся изображениях.
- Закрашивать произвольные прямоугольные области экрана заданным цветом с заданной степенью прозрачности с помощью функции [fillRect\(\)](#).
- Копировать произвольные прямоугольные области памяти из одного места в другое с помощью функции [bitBlit\(\)](#) с использованием ключевых цветов и альфа-канала.
- Масштабировать произвольные прямоугольные области в памяти, задавая при этом размеры в источнике и в приемнике с помощью функции [stretchBlit\(\)](#).
- Разворачивать выводимое изображение в функциях [bitBlit\(\)](#) и [stretchBlit\(\)](#) на 90/180/270 градусов с помощью задания опций.
- Загружать в память элементы графики в виде прямоугольных областей из графических файлов в стандартных форматах **BMP**, **JPG**, **PNG**, или «сырых» форматах.
- Отображать оверлейные области памяти в виде прямоугольной области на экране с масштабированием, альфа-каналом и использованием ключевых цветов.
- Автоматически преобразовывать цветовое пространство при копировании областей памяти из **RGB565/RGB888/YUV422/YUV411/TILED YUV** в заданное в приемнике.

Все указанные действия производятся с использованием графического **2D** акселератора на аппаратном уровне, поэтому практически не отнимают процессорного времени.

6.1.2. Инициализация графического адаптера

Для инициализации графического адаптера в одном из режимов **HDMI**, **TV-Out** или **LVDS** используются функции `setVideoMode()` и `initLvdsDisplay()`. В процессе инициализации заполняется структура `Display`, вынесенная в глобальную переменную с тем же именем. Эту переменную можно использовать в дальнейшей работе. Так, например, чтобы получить ширину экрана в пикселях в переменную `ScreenWidth`, следует записать:

```
ScreenWidth = Display.Width;
```

6.1.3. Работа с поверхностями

Каждый графический объект, размещаемый в оперативной памяти и представляющий собой заполненную в соответствии с установленным для него цветовым пространством прямоугольную область памяти, имеет одинаковый заголовок, позволяющий правильно осуществлять функции битблиттинга из одного такого объекта в другой **2D** акселератором. Структура заголовка `g2d_image`, описывающая любой графический объект, имеет псевдоним `surface_t` – **поверхность**. А указатель на эту структуру называется `HDC`.

Так как поверхности размещаются в оперативной памяти процессора, то создавать их можно столько, сколько позволит имеющийся объём памяти. Практически, вся работа графической библиотеки в `MULTEX-ARM` заключается в манипуляции поверхностями: создании, загрузке из файлов, копировании прямоугольных областей из одной поверхности в другую, рисования на поверхностях и т.п. Методы работы с поверхностями описаны в файле `a20graph.h`.

6.1.3.1. Примеры работы с поверхностями Пример работы с поверхностями показан в *тестовой* функции `bitBltTest`. Данная функция инициализирует графический адаптер **HDMI** и выводит на экран результат работы аппаратного 2D акселератора. Акселератор производит наложение поверхностей, окрашенных в разные цвета, с заданной прозрачностью. Источник данных о прозрачности (альфа-канал) можно выбрать из перечисления `g2d_blt_flags` и ввести в виде числа в качестве аргумента функции.

См. также

Подключение примеров и тестовых функций.

```
void bitBltTest (int blt_flags) {
    // Инициализация графического адаптера в режиме HDMI.
    if ((int) getLFB () == 0)
        setVideoMode (MODE_1920x1080, 32);
    init_2D_engine ();

    // Начало подготовки нового экрана.

    // Размеры для вывода тестовых поверхностей.
    int dw = CONSTR->w / 5;
    int dh = CONSTR->h / 5;

    // Создание тестовых поверхностей.
    HDC surfR = newSurface (dw * 3, dh * 3, G2D_FMT_ARGB8888);
    HDC surfG = newSurface (dw * 3, dh * 3, G2D_FMT_ARGB8888);
```

```
HDC surfB = newSurface (dw * 3, dh * 3, G2D_FMT_ARGB8888);

// Заполнить всю конструируемую поверхность белым цветом.
fillRect (CONSTR, 0, 0, CONSTR->w, CONSTR->h, 0xFF, 0xFFFFFFFF, G2D_FIL_NONE);

// Заполнение тестовых поверхностей
// (просто копирование как есть).
fillRect (surfR, 0, 0, surfR->w, surfR->h, 0, 0x40FF0000, G2D_BLT_NONE);
fillRect (surfG, 0, 0, surfG->w, surfG->h, 0, 0x4000FF00, G2D_BLT_NONE);
fillRect (surfB, 0, 0, surfB->w, surfB->h, 0, 0x400000FF, G2D_BLT_NONE);

// Ключевой цвет для тестирования Color Key
unsigned int colorKey = 0xFF00FF00;

// Копирование созданных поверхностей
// в конструируемую поверхность.

// Простое копирование поверхностей.
bitBlt (CONSTR, dw * 0, dh * 2, surfR->w, surfR->h, surfR, 0, 0, 0x80,
colorKey, blt_flags);
bitBlt (CONSTR, dw * 2, dh * 0, surfB->w, surfB->h, surfB, 0, 0, 0x80,
colorKey, blt_flags);
bitBlt (CONSTR, dw * 1, dh * 1, surfG->w, surfG->h, surfG, 0, 0, 0x80,
colorKey, blt_flags);

// Смена видимой и конструируемой поверхностей.
FlipScreenAndConstr ();

// Ожидание окончания процесса вывода новой поверхности.
waitVerticalRetrace ();

// Сохранение снимка экрана с номером режима в имени.
char str[10];
sprintf (str, "\\%s", "{bblt}");
makeSS (str, blt_flags);

// Удаление созданных объектов из памяти
deleteSurface (surfR);
deleteSurface (surfG);
deleteSurface (surfB);
}
```

При вызове функции с параметром `G2D_BLT_PIXEL_ALPHA`:

```
C:/>bitBltTest 1
```

копирование поверхности будет произведено с учётом значения **Alpha** цвета каждого пикселя исходной поверхности. В рассматриваемом примере это значение для каждого пикселя равно **0x40**, то есть новые цвета будут добавлены с прозрачностью $\alpha = 0,25$. Итоговое *изображение*

приведено ниже.

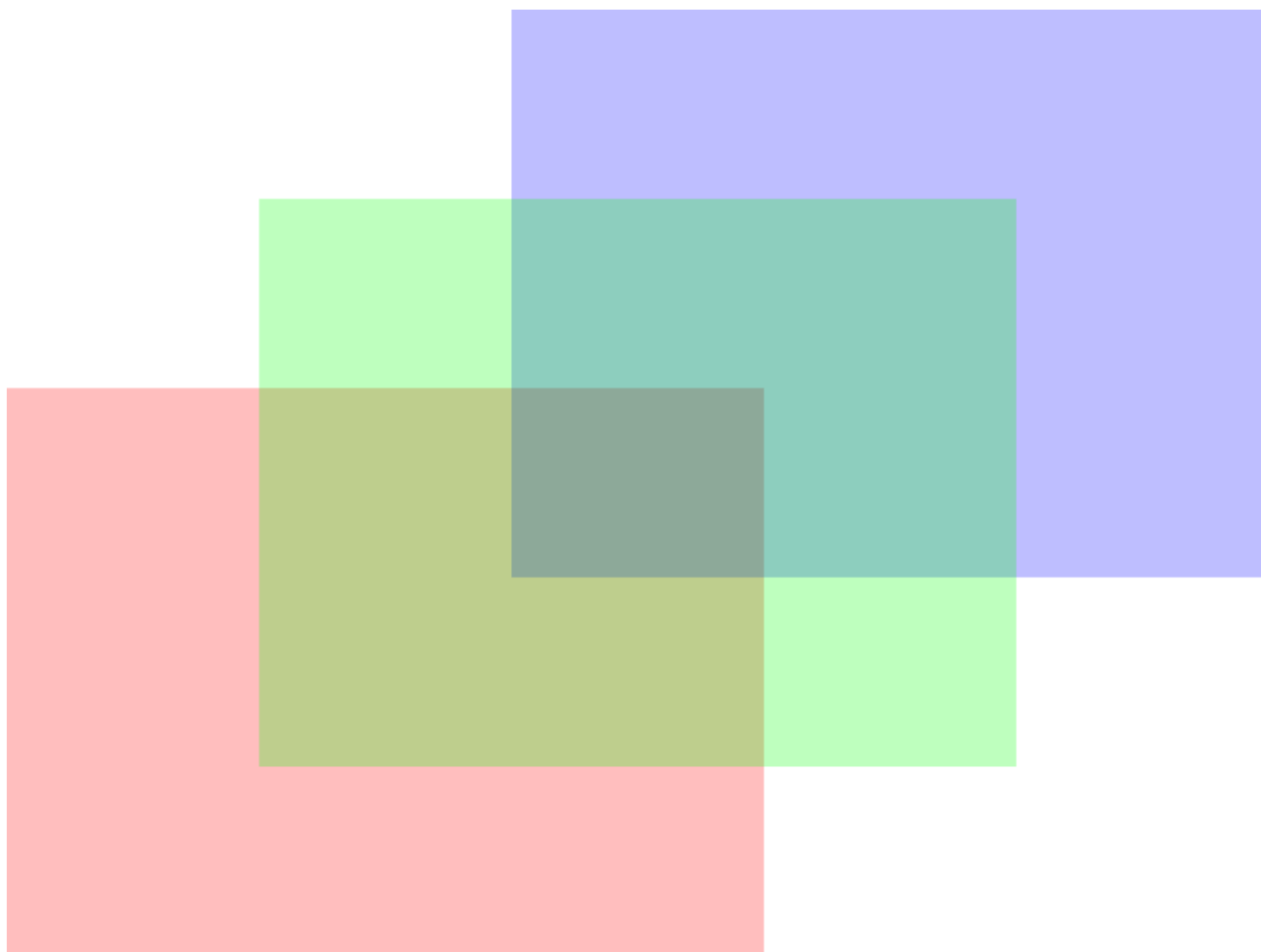


Рисунок 3. Копирование поверхностей с параметром `G2D_BLT_PIXEL_ALPHA`.

При вызове функции с параметром `G2D_BLT_PLANE_ALPHA`:

```
C: />bitBltTest 2
```

копирование поверхности будет произведено с учётом параметра **alpha** функции `bitBlt()`. В рассматриваемом примере это значение для всех поверхностей равно **0x80**, то есть новые цвета

будут добавлены с прозрачностью $\alpha = 0,5$. Итоговое *изображение* приведено ниже.

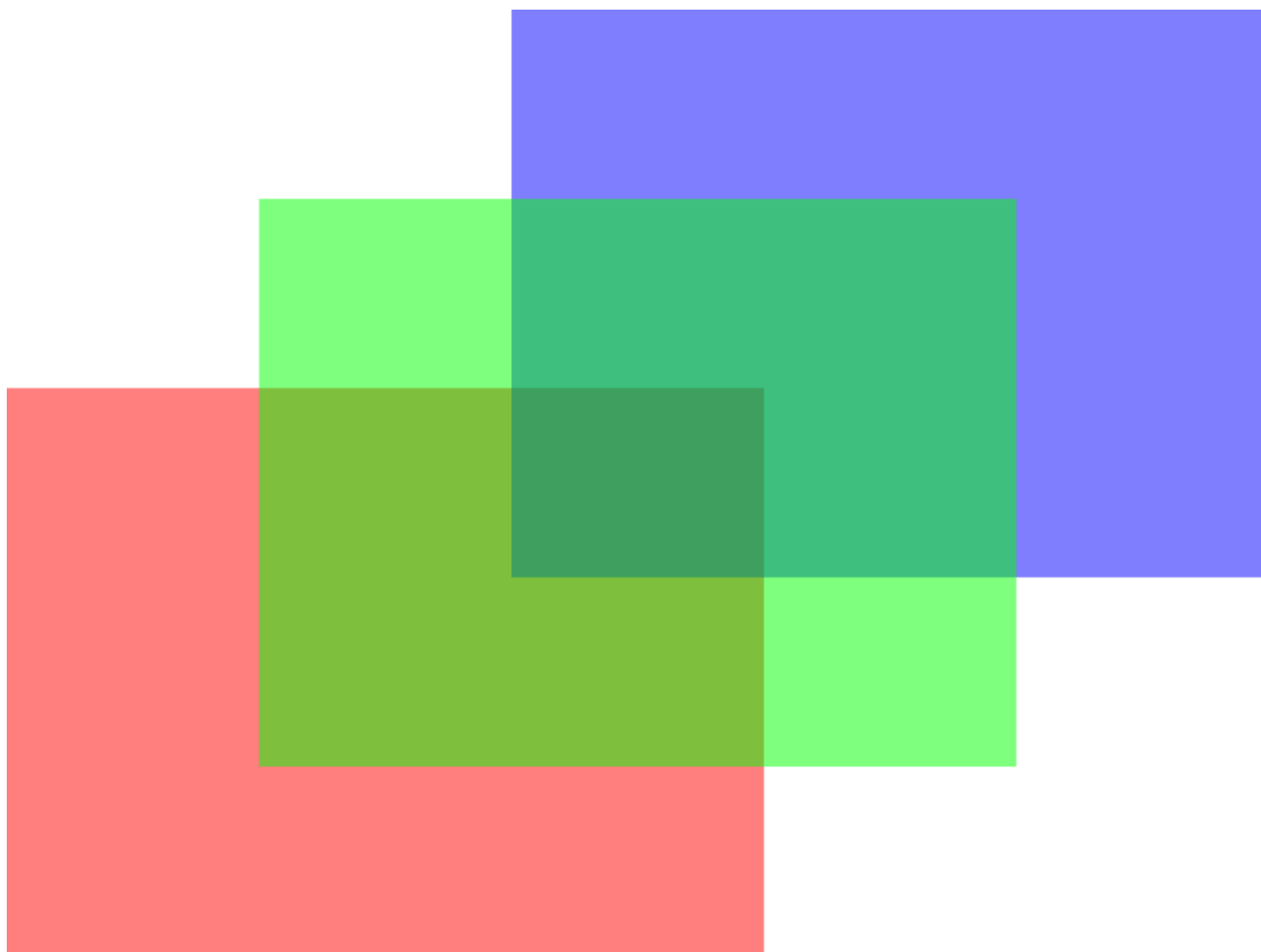


Рисунок 4. Копирование поверхностей с параметром `G2D_BLT_PLANE_ALPHA`.

При вызове функции с параметром `G2D_BLT_MULTI_ALPHA`:

```
C: />bitBltTest 4
```

копирование каждого пикселя поверхности будет произведено с учётом его прозрачности домноженной на параметр **alpha** функции `bitBlt()`. В рассматриваемом примере это значение для

всех пикселей $\alpha = 0,25 \cdot 0,5 = 0,125$. Итоговое *изображение* приведено ниже.

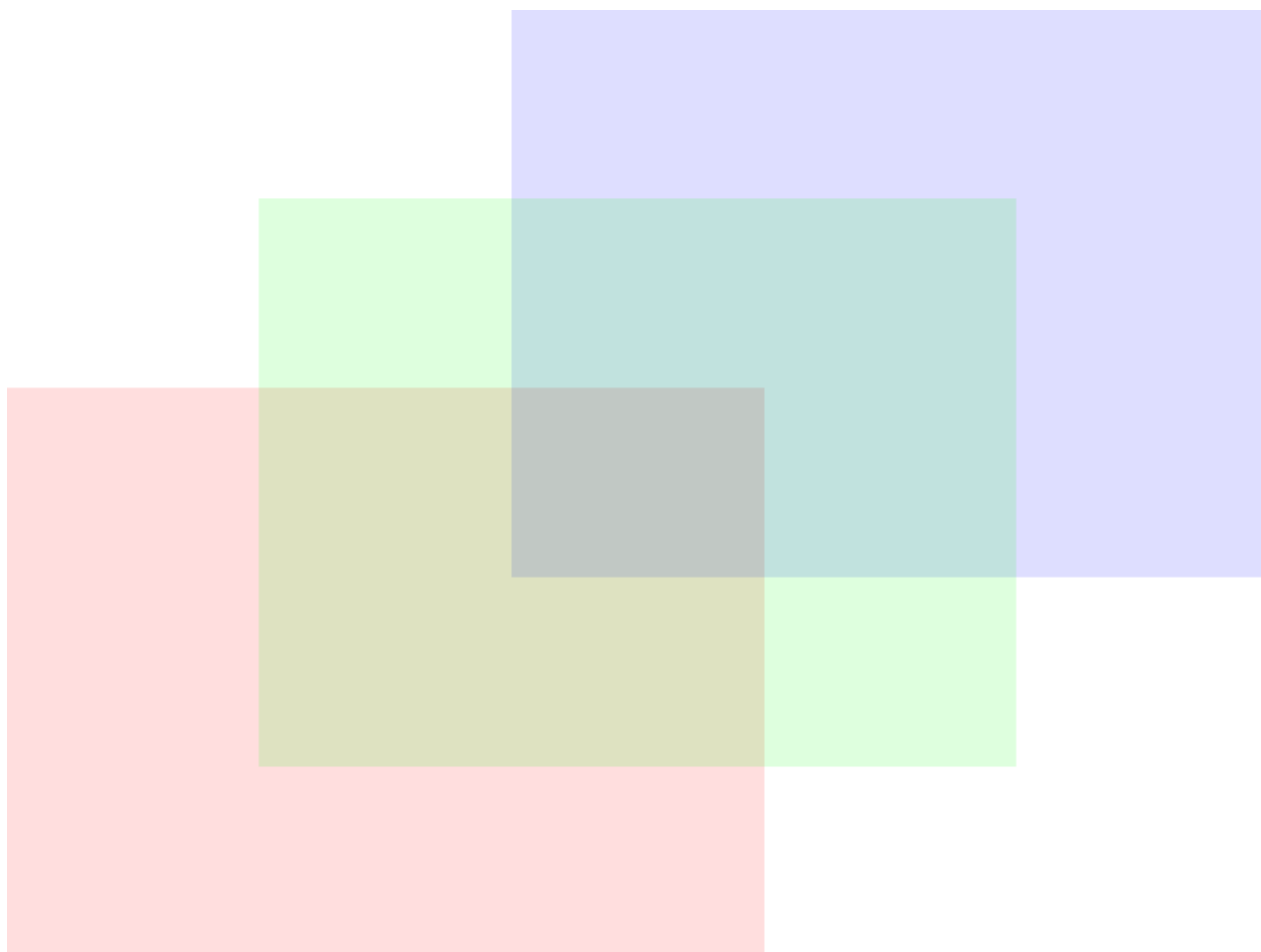


Рисунок 5. Копирование поверхностей с параметром G2D_BLT_MULTI_ALPHA.

Если добавить в опции копирования флаг *G2D_BLT_SRC_COLORKEY*, то при копировании будут отбрасываться пиксели копируемой поверхности, имеющие цвет, указанный в параметре **color** функции *bitBlit()*. В тесте в качестве ключевого цвета указан зелёный, поэтому при вызове теста с добавленным флагом *G2D_BLT_SRC_COLORKEY*

```
C: />bitBlitTest 9
```

зелёная поверхность будет отброшена целиком. Итоговое *изображение* приведено ниже.

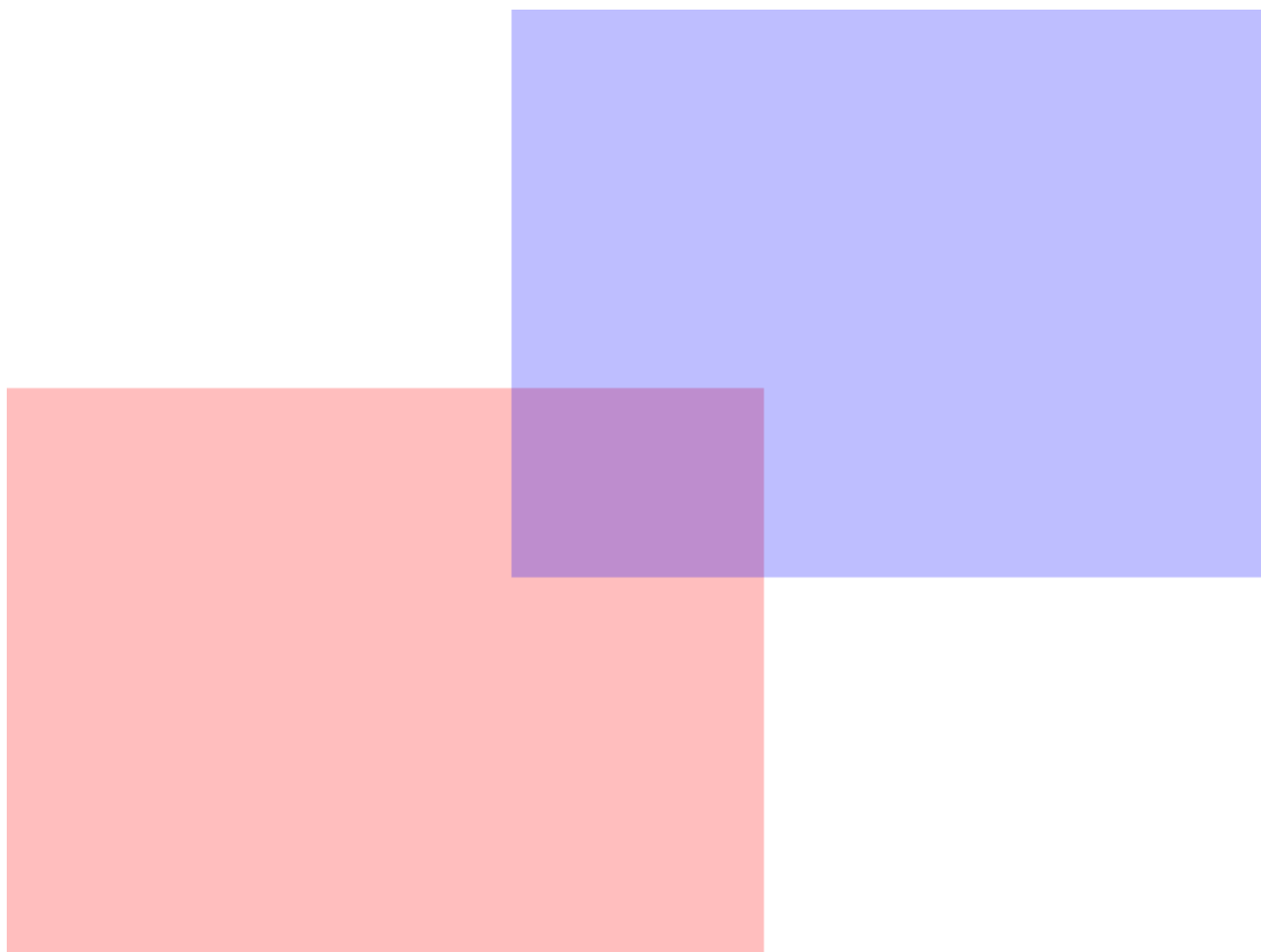


Рисунок 6. Копирование поверхностей с параметрами `G2D_BLT_PIXEL_ALPHA` | `G2D_BLT_SRC_COLORKEY`.

Обратите внимание, что при использовании опций `G2D_FIL_NONE` и `G2D_BLT_NONE` при заливке и копировании поверхностей изначальный цвет копируется вместе с исходной прозрачностью, что может привести к неоднозначному результату при отображении итоговой картины на различных устройствах. Например, в функции `bitBlitTest` все цвета имеют исходную прозрачность `0x40`. Итоговая поверхность получает эту прозрачность в местах наложения тестовых поверхностей и накладывается на пустую область памяти. В результате на дисплее будут отображены тёмные цвета вместо чистых красного, зелёного и синего. Снимок экрана в формате **BMP** (см. *рисунок*) имеет информацию о прозрачности, которая отбрасывается при выводе на других устройствах, а значит цвета такого снимка будут отображены как чистые **RGB**.



Рекомендуется точно указывать источник альфа-канала в функциях *fillRect()* и *bitBlit()*, либо использовать изначальные цвета с прозрачностью **0xFF**.

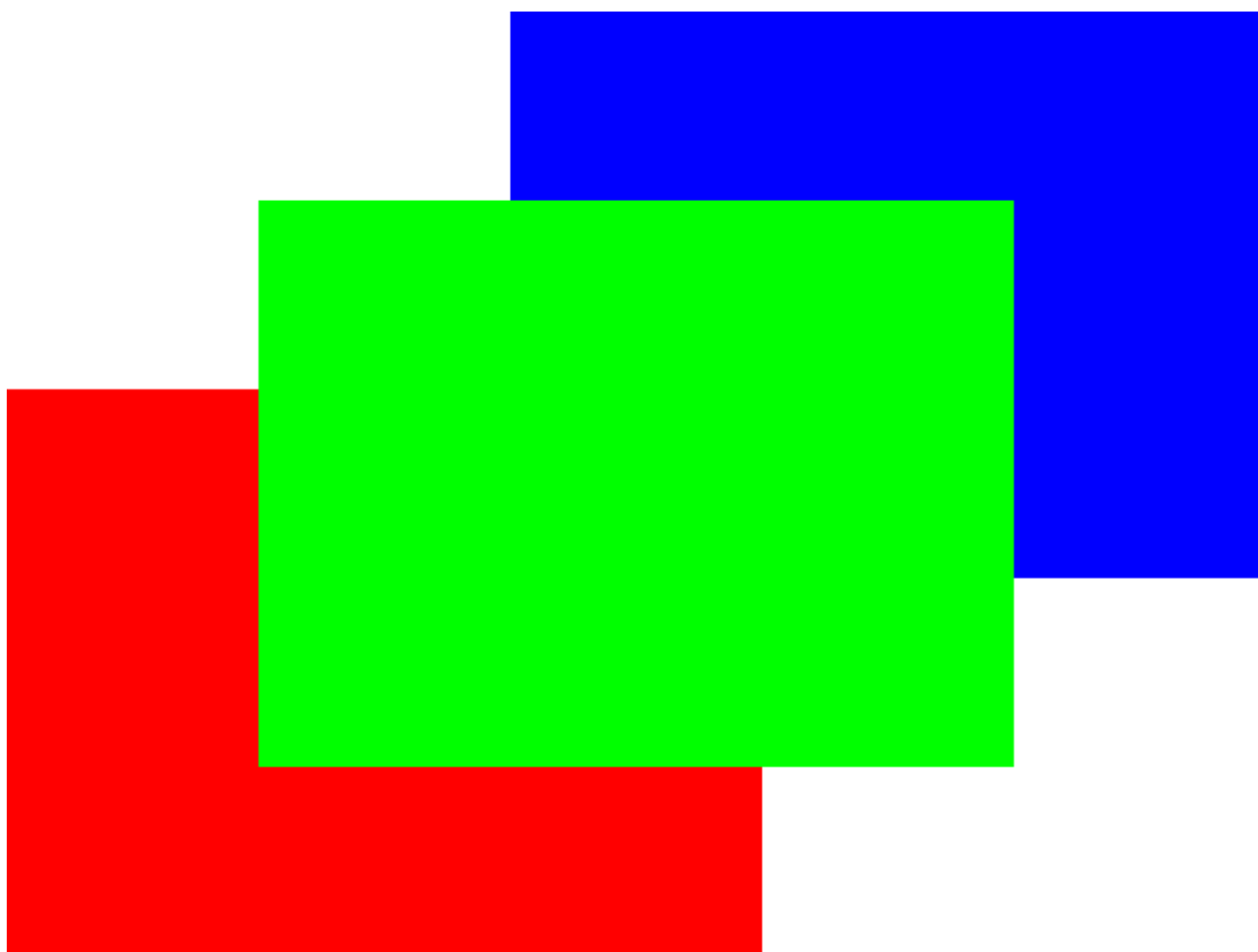


Рисунок 7. Простое копирование поверхностей.

Продемонстрировать работу опций поворота и отражения поверхностей с помощью аппаратного 2D акселератора можно с помощью другой функции из предоставляемых *примеров stretchBlitTest()*. Она отличается только способом создания поверхностей (они загружаются из графических файлов разных форматов) и методом копирования в конструируемую поверхность. Для корректной работы теста загружаемые файлы должны находиться на диске C: целевой платы в паке **surf**. Для копирования поверхностей с изменением масштаба используется функция *stretchBlit()*. В листинге ниже приведена только часть тестовой функции, отличающаяся от предыдущего теста — создание

поверхностей из файлов и копирование поверхностей с изменением масштаба.

```
// Загрузка поверхностей из файлов.

HDC surfBG = loadPNGSurface ("surf/bkgr1080.png");
HDC surfImg = loadPNGSurface ("surf/mario.png");

// Размеры накладываемой поверхности (1/2 высоты экрана).
int dstA = CONSTR->h / 2;
// Координаты, куда положить накладываемую поверхность.
int dstX = (CONSTR->w - dstA) / 2;
int dstY = (CONSTR->h - dstA) / 2;

// Копирование созданных поверхностей
// в конструируемую поверхность с масштабированием.
// Подложка
stretchBlt (CONSTR, 0, 0, CONSTR->w, CONSTR->h, surfBG, 0, 0, surfBG->w,
surfBG->h, 0, 0, G2D_BLT_NONE);
// Изображение
stretchBlt (CONSTR, dstX, dstY, dstA, dstA, surfImg, 0, 0, surfImg->w,
surfImg->h, 0, 0, blt_flags);
```

Для простого вывода загруженного контента можно использовать вызов функции с параметром `G2D_BLT_PIXEL_ALPHA` — будет выполнено копирование созданных поверхностей с масштабированием и учётом их прозрачности.

```
C:/>stretchBltTest 1
```

Первым в тесте копируется изображение, загруженное из файла **BMP**, не имеющее альфа-канала. Поверх него будет наложено изображение, загруженное из файла **PNG**, имеющее прозрачный

фон. Результирующее *изображение* показано ниже.



Рисунок 8. Копирование поверхностей с масштабированием.

Акселератор позволяет аппаратно отражать поверхности по горизонтали, вертикали и диагоналям, а так же, поворачивать поверхности на 90, 180 и 270 градусов. Например, для отражения вдоль диагонали под 45°, следует вызвать тест с флагами `G2D_BLT_PIXEL_ALPHA` и `G2D_BLT_MIRROR45`:

```
C: />stretchBlitTest 1025
```

Результат наложения отмасштабированных поверхностей с отражением по диагонали показан на

рисунке ниже.



Рисунок 9. Копирование поверхностей с отражением.

6.1.4. Известные ошибки работы с поверхностями

6.1.4.1. Ошибка при повороте поверхности на 45°

Ошибка Повороты и отражения поверхностей на 90° средствами графического 2D-акселератора в любую сторону работает корректно только для квадратных объектов. Поворот прямоугольных объектов приводит ко появлению артефактов.

6.1.4.2. Ошибка ColorKey для результирующей поверхности

Ошибка Использование *ColorKey* для игнорирования пикселей результирующей поверхности в графическом 2D-акселераторе приводит к смешиванию полупрозрачных пикселей исходной поверхности с полностью прозрачными пикселями результирующей поверхности. Смешивание двух полупрозрачных пикселей реализовано в аппаратном миксере с ошибкой. Не рекомендуется использовать данную *опцию*.

6.2. Работа с оверлеями

Графический **2D** акселератор, встроенный в процессор **Allwinner A20**, позволяет выводить на экран аппаратный **оверлей** — прямоугольную область экрана, отображающую содержимое произвольной зоны оперативной памяти. При этом цветовое пространство для этой зоны может отличаться от цветового пространства отображаемого в данный момент фрейм-буфера. Так, например, фрейм-буфер может содержать данные в формате **RGB888** а оверлейная зона — в формате **YUV422**. Методы работы с оверлеями описаны в файле *a20graph.h*.

6.2.1. Пример работы с оверлеем

В качестве примера можно привести вывод **AVI** файла в экранную область, используемый в функции и **aviTest** из поставляемых *местов*.

Для начала работы необходимо вызвать процедуру инициализации *OverlayInit()* и открыть **overlay** с помощью *OverlayOpen()*. Для вывода **overlay** поверх основной экранной области следует изменить приоритет с помощью *setOverlayPriority()*. Используя прозрачность в поверхностях основной экранной области можно выводить графические объекты поверх изображения области **overlay**. Пример запуска **overlay** показан ниже. Адреса зоны **overlay** (videoBuff) в данном примере располагаются в выходном буфере декодера **MP4** с учётом размеров кадра воспроизводимого видео.

```
// Настройка области оверлей, запуск вывода на экран.
int lSizeM = ALIGN (videoWidth, 32) * ALIGN (videoHeight, 32);
videoBuff[0] = (int) pOutFrame;
videoBuff[1] = videoBuff[0] + lSizeM;
videoBuff[2] = videoBuff[1] + lSizeM / 4;

OverlayInit ();
// Поднять приоритет вывода - выводить поверх основного экрана.
setOverlayPriority (2);
OverlayOpen ((int*)videoBuff, 0, 0,
            videoWidth, videoHeight,
            screenWidth, screenHeight,
            OT_MB_UV_COMBINED);
```

В процессе декодирования видео файла готовые кадры сразу попадают в зону **overlay** и отображаются на экране без дополнительного копирования. После завершения воспроизведения следует завершить работу с **overlay** и освободить используемую память с помощью *OverlayClose()*.

6.3. Аппаратный TV-декодер

В **MULTEX-ARM** реализована поддержка аппаратного декодера видео сигнала **CVBS**. Для работы с аппаратной частью следует использовать устройство захвата телевизионного сигнала. Устройство захвата позволяет:

- принимать видео сигнал стандартов **PAL / NTSC**;
- получать сигнал с разрешениями **480i, 576i, 480p, 576p**;

- захватывать 4 канала видео одновременно.

Для подключения библиотеки работы с аппаратным TV-декодером к проекту необходимо добавить соответствующие библиотеки в *Makefile*:

```
LIBRARIES += -l_tvd
```

См. также

Описание работы устройства захвата телевизионного сигнала и примеры кода в файле *tv-receiver.h*.

6.4. Поддержка шрифтов FreeType

В ядро ОСРВ *MULTEX-ARM* включены библиотеки поддержки шрифтов **TrueType**. Функции *графической подсистемы* работают с **ТТf** шрифтами на уровне поверхностей. Отображение шрифтов с использованием альфа-канала производится при этом на аппаратном уровне.

Для подключения библиотек работы со шрифтами к проекту необходимо добавить соответствующие библиотеки в *Makefile*:

```
LIBRARIES += -l_font -l_freetype
```

См. также

Описание методов работы со шрифтами в файле *fonts.h*.

6.4.1. Пример работы со шрифтами

Надписи, сделанные заданным шрифтом, выводятся на заранее созданные *поверхности* обрабатываемые **2D** графическим акселератором. Соответственно перед использованием шрифтов следует запустить графический адаптер и акселератор, как показано в разделе *Примеры работы с поверхностями*. Ниже приведён пример вывода текста на поверхность *CONSTR*. Текст выводится поверх ограничивающего прямоугольника. Размеры прямоугольника рассчитаны средствами, предоставляемыми этой же библиотекой.

```
// Загрузка шрифта
pTtfFont font = ttf_LoadFont ("{fnt/nsans-md.ttf"}}, 64, 0x3A3A3A,
ttf_NoGlyphSaving);

// Расчёт ограничивающего прямоугольника
sTextRect rect;
ttf_GetTextRect (&rect.w, &rect.h, font, testString);
rect.x = (CONSTR->w - rect.w) / 2;
rect.y = (CONSTR->h - rect.h) / 2;

// Показать ограничивающий прямоугольник
fillRect (CONSTR, rect.x, rect.y, rect.w, rect.h, 0xFF, 0xFFFFFFFF,
G2D_FIL_NONE);

// Вывести надпись по центру
```

```
ttf_PrintRect (CONSTR, (pTextRect)\&rect, alignHCenter | alignVMiddle, font,
testString);

// Удалить шрифт после использования
ttf_FreeFont (font);
```

После окончания подготовки поверхности следует поменять местами конструируемую и подготавливаемую поверхности, как показано в *примере*. Результат работы приведённого кода показан на *рисунке* ниже.



Рисунок 10. Вывод надписи на поверхность.

6.5. Работа с AVI-файлами

Для подключения работы с файлами **AVI** к проекту необходимо добавить соответствующую библиотеку в *Makefile*:

```
LIBRARIES += -l_a20graph -l_avi -l_mpeg4decode
```

В состав *MULTEX-ARM* входит библиотека для работы (чтения и записи) с файлами-контейнерами стандарта **AVI** (*Audio Video Interleave*), содержащими видео и звуковую информацию. После записи

таких файлов и переносе их в персональный компьютер, они могут быть проиграны на нем стандартными средствами. Файлы **AVI**, записанные на стороннем компьютере, могут быть воспроизведены в среде *MULTEX-ARM*, только если использовался кодек видео **h.264** или **MP4** и наличии ряда ограничений. В реализации **MP4** не поддерживаются бэкфреймы. В **h.264** реализовано ограниченное количество форматов. Перед использованием видеофайлов в проектах *MULTEX-ARM* рекомендуется конвертировать их с помощью `ffmpeg`. Ниже приведён пример скрипта конвертации некоторого видео файла `video.mp4` в поддерживаемый формат. Разрешение итогового видео можно изменить в параметре `scale`.

```
ffmpeg -y -i "video.mp4" -hide_banner -vf scale=1920:1080 -c:v libxvid -b:v 8000k -bf 0 -force_key_frames expr:eq(n,0) -an "video.avi"
```

См. также

Описание методов работы с **AVI**-файлами — *avilib.h*.

6.5.1. Пример воспроизведения AVI файла через overlay

Пример воспроизведения **AVI** файла в формате **MP4** показан в *местовой* функции `aviTest`, приведённой в файле примеров `avilib-example.c`. Данная функция воспроизводит в *Оверлей* и выводит на экран видео файл, записанный на жёстком диске целевой платы. По умолчанию это файл `avi/countdown.avi`, поставляемый вместе с примерами.

Для начала воспроизведения файл открывается с помощью `openAVIFile()`. Также следует выбрать видео режим `setVideoMode()`. Для воспроизведения видео в **overlay** его следует запустить, как описано в разделе *Работа с оверлеями*. Далее выполняется настройка декодера **MP4**, как показано в листинге:

```
mpegDecoder = mpeg4InitDecoder (videoWidth, videoHeight);  
pInptBuf = getMpeg4InptFrameBuff (mpegDecoder);  
pOutFrame = (unsigned char*) getMpeg4OutFrame (mpegDecoder);
```

Покадровое чтение файла производится с помощью `readAVIFrame()` с контролем возвращаемого результата. Закольцевать воспроизведение одного файла можно с помощью `seekToFirstVideoFrame()` после достижения конца файла.

```
int readLen;  
while ((readLen = readAVIFrame (aviFile, (char*)pInptBuf)) == -1) {  
    seekToFirstVideoFrame (aviFile);  
}
```

Каждый считанный кадр декодируется с помощью `mpeg4DecodeBlock()`. При этом итоговый кадр сразу же попадает в **overlay** и выводится на экран.

```
if (mpeg4DecodeBlock (mpegDecoder, NULL, readLen) != OK) {  
    printf ("{decode error\n"});  
    return ERROR;  
}
```

После выхода из задачи воспроизведения использованные ресурсы освобождаются.

```
closeAVIFile (aviFile);
freeMpeg4FrameMem (mpegDecoder);
OverlayClose ();
```

6.5.2. Пример воспроизведения AVI файла через 2D акселератор

Воспроизводить видео поток можно также на подготовленную *поверхность*, микшируемую в экранную область с помощью аппаратного миксера.

Пример одновременного воспроизведения 4-х видео потоков в экранной области приведён *местовой* функции **aviTest4**, из файла примеров **avi-4-streams.c**. В отличие от предыдущего примера декодирование будет производиться в заранее отведённые поверхности. Инициализация одной из 4-х областей показана ниже.

```
HDC pSrf0 = newSurface (width1, height1, G2D_FMT_PYUV420UVC);
```

Покадровое чтение из открытого файла **ah1** производится как и в предыдущем примере во входной буфер декодера **stream1**, полученный с помощью *getMpeg4InptFrameBuff()*.

```
while ((len1 = readAVIframe (ah1, (char*)stream1)) == -1) {
    seekToFirstVideoFrame (ah1);
}
```

А вот декодирование выполняется в ранее подготовленную поверхность **pSrf0**.

```
if (mpeg4DecodeBlock (dd1, (unsigned char *)pSrf0->addr[0], len1) != OK) {
    printf ("{decode error in line %i\n"}{, __LINE__);
    return ERROR;
}
```

Далее выполняется смешивание декодированной поверхности с конструируемой.

```
stretchBlt ((HDC)pConstr, ddRect1.posX, ddRect1.posY, vidOutWidth,
vidOutHeight, (HDC)pSrf0, 0, 0, width1, height1, 0, 0, G2D_BLT_NONE);
```

После завершения копирования всех поверхностей на конструируемую последняя выводится на экран.

```
FlipScreenAndConstr ();
waitVerticalRetrace ();
```

Далее запускается чтение и декодирование нового кадра.

После завершения задачи воспроизведения все созданные поверхности следует освободить с помощью `deleteSurface()`, используемые декодеры следует освободить с помощью `freeMpeg4FrameMem()`, а открытые файлы закрыть с помощью `closeAVIFile()`.

6.6. Кодер/декодер видео h.264 CEDRUS

См. также

Описание методов работы с **CEDRUS** в файле `cedrus.h`.

В состав библиотек **RTOS MULTEX-ARM** входит библиотека аппаратного кодирования/декодирования видео потоков в стандарте **h.264** — **CEDRUS**.

Библиотека позволяет осуществлять в реальном времени как кодирование видеоинформации, получаемой от камеры, так и декодирование с последующим выводом на экран монитора ранее закодированной информации. Это позволяет в частности передавать видеоинформацию по сравнительно низкоскоростным каналам передачи от системы-сервера, снабженного видеокамерой, системе-клиенту с целью вывода полученной информации на экран монитора, или записи на диск в виде файла. Такие системы находят широкое применение в системах видео наблюдения и системах управления беспилотными аппаратами.

6.7. Звуковая подсистема

См. также

Описание методов работы со звуковой подсистемой в файле `sound.h`.

В составе библиотек ОС **MULTEX-ARM** имеется поддержка записи/воспроизведения звука. Для подключения звуковой подсистемы следует вызвать функцию инициализации `soundInit()`. Данная функция подключат аппаратный кодек используемого процессора. Ниже приведён пример запуска звуковой подсистемы и запуска воспроизведения файла.

```
\#include <sound.h>
void play () {
    soundInit ();
    setMasterVol (70);
    playWaveFile ("{sound.wav}");
}
```

7. Программный вывод графики

7.1. Графика на простых процессорах

В простых **ARM** процессорах, таких как **V3s**, отсутствует возможность аппаратной обработки графических объектов. В таких процессорах, как правило, есть только небольшой графический модуль для вывода участков памяти на экран, называемый **Display-Engine 2**. Работа с поверхностями возможна в этом случае только программно. Для программной обработки графических объектов и вывода итоговых изображений на экран в **MULTEX-ARM** предназначены библиотеки **softgraph** и **de2**.

Для подключения библиотек к проекту необходимо добавить их в *Makefile*:

```
LIBRARIES += -l_de2 -l_softgraph
```

При использовании файлов в формате **PNG** и **JPG** понадобятся также дополнительные библиотеки:

```
LIBRARIES += -l_png -l_z -l_jpeg
```

См. также

Описание методов программного вывода графики в файлах *softgraph.h* и *de2.h*.

7.1.1. Общее описание

Display-Engine 2 — простой аппаратный модуль **ARM** процессоров компании *Allwinner*, позволяющий выводить указанные области памяти на дисплей. В настоящее время поддержан вывод только через **RGB** параллельный интерфейс, что идеально подходит для бюджетных проектов с небольшими дисплеями. За настройку аппаратной части модуля и работу с ним отвечает библиотека **de2**. Функции библиотеки описаны в файле *de2.h*.

Soft Graph — библиотека программного построения сцен на основе поверхностей с поддержкой прозрачности. С её помощью происходит подготовка выводимых изображений в неактивной области экранной памяти. После завершения отрисовки сцены экранные области меняются местами средствами библиотеки **de2**. В результате подготовленная область памяти начинает выводиться на экран, в то время как ранее активная область памяти становится доступна для отрисовки новой сцены. Ознакомиться с методами библиотеки можно в файле *softgraph.h*.

7.1.2. Работа с поверхностями

Каждый графический объект, размещаемый в оперативной памяти и представляющий собой заполненную в соответствии с установленным для него цветовым пространством прямоугольную область памяти, имеет одинаковый заголовок, позволяющий правильно осуществлять функции программного битблиттинга из одного такого объекта в другой. Структура заголовка, описывающая графический объект, определена в *SURFACE*. А указатель на эту структуру называется *HDCp*.

Так как поверхности размещаются в оперативной памяти процессора, то создавать их можно столько, сколько позволит имеющийся объём памяти. Практически, вся работа программной графической библиотеки в **MULTEX-ARM** заключается в манипуляции поверхностями: создании, загрузке из файлов, копировании прямоугольных областей из одной поверхности в другую, рисования на поверхностях и т.п. Методы работы с поверхностями описаны в файле *softgraph.h*.

7.1.2.1. Примеры программной работы с поверхностями Пример работы с поверхностями показан в *тестовой* функции **startScreen**. Данная функция инициализирует библиотеку аппаратного вывода графики **de2** и выводит на экран тестовое изображение. В конце работы происходит высвобождение занятых ресурсов.

См. также

Подключение примеров и тестовых функций.

```
void startScreen () {
    tScreenDeviceMode mode;
    sDisplayInfo display;

    // Заполнение описания экрана стандартными значениями
    de2GetDefaultScreenMode (&mode,
                             screenType_TM043NBH02);

    // Инициализация программной и аппаратной части
    display.width = mode.xres;
    display.height = mode.yres;
    display.bpp = mode.bpp;
    softGraphInit (&display,
                   de2Init (&mode, ovDataFormat_ARGB_8888)
                   );

    // Загрузка изображения
    HDCr bkgr;
    bkgr = loadFromJPG ("surf/wall_272.jpg");

    // Вывод изображения на экран
    sfDraw (softGraphConstr (), 0, 0, bkgr);

    // Ожидание окончания текущего буфера на экран
    de2WaitVerticalRetrace ();

    // Смена экранной и конструируемой поверхностей
    softGraphConstrUpdate (
        de2FlipScreenAndConstr ()
    );

    // Включение подсветки - яркость 80%
    de2SetBacklight (80);

    // Удаление временных объектов
    freeSurface (bkgr);
}
```


8. Сетевая подсистема

В *MULTEX-ARM* имеются средства обмена информацией с другими устройствами по сети **Ethernet**. При этом используется стандартный стек IP-протоколов (**UDP** и **TCP/IP**). Благодаря этому обмен данными может производиться с любыми устройствами, поддерживающими эти протоколы. В частности, возможна работа *MULTEX-ARM* с удаленными устройствами, находящимися в глобальной сети **Internet**. Скорость соединения по сети **Ethernet** определяется используемым сетевым контроллером. Для процессора **A20** фирмы **AllWinner** она составляет 10, 100, или 1000 Мбит/сек. Драйвер сетевого контроллера, входящего в состав **A20**, имеется в ядре *MULTEX-ARM*.

Для работы с сетью по протоколу **TCP/IP** *MULTEX-ARM* использует библиотеку сокетов. Эта библиотека предоставляет пользователям **API**, практически полностью совместимый с **API** сокетов Беркли (**BSD**).

Для работы по протоколу **UDP** в *MULTEX-ARM* использован **API** собственной разработки, который на наш взгляд более удобен, чем сокет типа **DATAGRAM**.

8.1. Подключение к проекту

Для подключения сетевой подсистемы к проекту необходимо:

1. Указать соответствующий макрос в файле *config.h*:

```
#define INCLUDE_NETINET
```

2. Указать сетевой адрес устройства и прочие настройки сети в файле *config.h*.
Рекомендованный блок настроек приведен ниже:

```
#ifdef INCLUDE_NETINET
#define IP_ADDRESS "10.0.3.36"
#define CHECK_PRIMARY_IP_ADDRESS
#define INCLUDE_NETLOADER
#define INCLUDE_TCP
#ifdef DEBUG
#define FTP_SERVER
#define FTP_ALLOW_TO_CHANGE_WORK_DRIVE
#define INCLUDE_NET_CONSOLE
#define INCLUDE_TCP_CONSOLE
#endif
#endif
```

Указанный сетевой адрес можно изменить в процессе работы устройства с помощью функции *setPrimaryIpAddress()*.

3. Подключить сетевые библиотеки в *Makefile*:

```
LIBRARIES += -l_enet -l_tcp
```

8.2. Протокол UDP

UDP расшифровывается как **User Datagram Protocol** — протокол дейтаграмм пользователя. Сообщение **UDP** называется дейтаграммой по аналогии с телеграммой — короткое сообщение,

которое быстро доставляется. **UDP** не предоставляет дополнительного уровня надежности по сравнению с протоколом **TCP/IP**.

См. также

Описание методов работы с протоколом **UDP** и **примеры кода** в файле [udp.h](#).

8.3. Протокол TCP/IP, сокет TCP

В отличие от **UDP**, **TCP** обеспечивает надежную доставку данных. При этом **TCP** обеспечивает как гарантию доставки данных, так и гарантию сохранения порядка следования сообщений. Для работы по протоколу **TCP** используются сокет.

Для передачи данных через сокет можно пользоваться стандартными функциями ввода / вывода [read\(\)](#) и [write\(\)](#).

См. также

Описание методов работы с протоколом **TCP/IP** в файле [socket.h](#).

9. Подсистема USB

См. также

Группа файлов *USB*.

9.1. Подключение к проекту

Для подключения библиотеки **USB** к проекту необходимо:

1. Указать соответствующий макрос в файле *config.h*:

```
#define INCLUDE_USB
```

2. Подключить сетевые библиотеки в *Makefile*:

```
LIBRARIES += -l_usb_sunxi
```

9.2. Общее описание

Текущая версия библиотеки **lib_usb_sunxi** поддерживает **control**, **bulk** и **interrupt** обмены. Такие обмены, как *isochronous* в данной версии не поддерживаются.

Для того, чтобы устройство, подключаемое к шине USB, было распознано и правильно настроено, в системе должен присутствовать драйвер этого устройства. Этот драйвер должен быть зарегистрирован в системе. При этом новый драйвер включается в цепочку USB-драйверов и при подключении нового устройства все драйверы цепочки будут последовательно «пробовать» к этому устройству до тех пор, пока функция примерки у очередного драйвера не вернет подтверждения пригодности. Таким образом, драйвер любого USB-устройства должен иметь функцию **probe()**, которая будет вызываться при подключении нового устройства. Для корректного отключения устройства его драйвер должен содержать еще одну функцию — **disconnect()**. Для регистрации драйвера в системе служит функция *usb_register_driver()*, которая подключает данный драйвер в цепочку. При этом необходимо указать имя, под которым устройство будет видно в системе, а также указатели на функции **probe()** и **disconnect()**. Регистрировать драйвер устройства можно до инициализации подсистемы USB. Тогда уже подключенное устройство будет правильно распознано. Если же регистрацию драйвера производить позже инициализации USB, то драйвер будет правильно работать только с вновь подключаемым устройством.

При подключении нового устройства к шине USB подсистема считывает для этого устройства все дескрипторы и устанавливает для него новый уникальный адрес. Все данные нового устройства заносятся в специальную структуру — *usb_device*. Функция **probe()** драйвера получает в качестве параметра указатель на эту структуру. Для того, чтобы определить, подходит ли это устройство для драйвера, функция **probe()** запрашивает данные устройства и сравнивает их с заданными. Если устройство уникально, то самое простое — запросить из структуры *usb_device_descriptor* поля *idVendor* и *idProduct*. Соответствие обоих этих полей заданным значениям однозначно подтверждает, что драйвер разработан именно для этого устройства. Если же драйвер написан для целого класса устройств, например это клавиатура или **massStorage**, то приходится проверять такие поля, как *bDeviceClass*, *bDeviceSubClass*, или *bDeviceProtocol*.

Если какие-то параметры в дескрипторах не соответствуют заданным, то функция **probe()** должна вернуть значение **-1**, в противном случае функция продолжает настройку устройства: посылает сообщения *control* или *bulk*, устанавливает обработчики прерываний от конечных точек. Если все проходит успешно, функция возвращает значение **0**.

9.3. Получение дескрипторов из структуры `usb_device`

При подключении нового устройства USB *MULTEX-ARM* загружает все дескрипторы для этого устройства в структуру `usb_device`, устанавливает адрес для нового устройства, после чего вызывает функцию `probe()` для каждого драйвера USB, зарегистрированного в системе.

При этом структура `usb_device` содержит поле `descriptor`, в которое помещается структура `usb_device_descriptor`. Аналогично, поле `config` содержит структуру `usb_configuration_descriptor_desc` и массив структур `usb_interface_if_desc[USB_MAXINTERFACES]`.

9.4. Использование интегрального параметра `pipe`

Параметр `pipe` является по сути 32-битным контейнером, в который помещаются в упакованном виде следующие параметры USB-канала обмена:

Биты	Значения
0-1	Максимальный размер пакета в байтах. Возможные значения: <ul style="list-style-type: none"> • 00 = 8 • 01 = 16 • 10 = 32 • 11 = 64
2-6	Зарезервировано.
7	Направление. Возможные значения: <ul style="list-style-type: none"> • 0 = Host-to-Device [Out] • 1 = Device-to-Host [In]
8-14	Номер устройства.
15-18	Номер конечной точки.
19	Текущее состояние переключателя. Возможные значения: <ul style="list-style-type: none"> • 0 = Data0 • 1 = Data1
20-25	Зарезервировано.
26-27	Скорость. Возможные значения: <ul style="list-style-type: none"> • 0 = Полная • 1 = Низкая • 2 = Высокая
28-29	Зарезервировано.
30-31	Тип канала. Возможные значения: <ul style="list-style-type: none"> • 00 = isochronous • 01 = interrupt • 10 = control • 11 = bulk

Такая упаковка бит максимально соответствует спецификации **UHCI**, поэтому упрощается разработка драйвера аппаратуры USB. Для создания нужного значения **pipe** в **MULTEX-ARM** имеются макросы описанные в отдельной *группе*. Кроме того, имеются *макросы*, облегчающие программистам многие стандартные действия с **pipe**.

10. Поддержка CSI (Camera Sensor Interface)

Для подключения методов работы с **CSI** к проекту необходимо добавить соответствующую библиотеку в *Makefile*:

```
LIBRARIES += -l_csi
```

См. также

Описание методов работы с аппаратным интерфейсом подключения цифровых камер в файле *csi.h*.

10.1. Работа с цифровыми видеокameraми

В *MULTEX-ARM* включена поддержка драйвера **CSI** (Camera Sensor Interface), обеспечивающего высокоскоростной параллельный обмен с цифровыми источниками видеосигнала, работающими по стандартам:

- CCIR656
- BT1120
- 8 bit raw
- 16 bit YUV422
- time-multiplexed ITU-R BT656 с поддержкой двойной буферизации.

Процессор **Allwinner A20** имеет два независимых порта **CSI**. Драйвер может работать одновременно с двумя входами. При этом **CSIO** поддерживает разрешение до **1080p@30FPS**, а **CSI1** разрешение **720p@30FPS**.

В библиотеке поддержки **CSI** (*csi.h*) содержатся все необходимые функции для работы с видеоканалом. Пример запуска канала *CSI_0* приведён в примере ниже:

```
// Создание устройства
void *dev = csiDevCreate (CSI_0, 1920, 1080);
// Настройка устройства
csiSetMode (dev, csiInFmt_YUV422, csiOutFmt_YCbCr_422_frp_UVc);
csiSetPhase (dev, csiActiveHigh, csiActiveHigh, csiRisingEdge);
// Запуск захвата видео
csiRun (dev);
while (anyCondition) {
    // Ожидание захвата кадра
    csiWaitFrame (dev);
    // Получение указателей на захваченные данные
    void *lBuf = csiGetLuma (dev);
    void *chBuf = csiGetChroma (dev);
    ...
}
// Остановка захвата, освобождение ресурсов
csiStop (dev);
```

11. Ошибки

Страница *Аппаратная поддержка мультимедиа* Повороты и отражения поверхностей на 90° средствами графического 2D-акселератора в любую сторону работает корректно только для квадратных объектов. Поворот прямоугольных объектов приводит ко появлению артефактов. Использование *ColorKey* для игнорирования пикселей результирующей поверхности в графическом 2D-акселераторе приводит к смешиванию полупрозрачных пикселей исходной поверхности с полностью прозрачными пикселями результирующей поверхности. Смешивание двух полупрозрачных пикселей реализовано в аппаратном миксере с ошибкой. Не рекомендуется использовать данную *опцию*.

12. Список устаревших определений и описаний

Глобальный ***intConnect*** (***int vector, usr_int_proc routine, int parameter***) Функция устарела, поскольку не поддерживает приоритеты прерываний. Подключаемый прерывания будут иметь значения приоритета по умолчанию. В новых проектах следует использовать функцию ***interruptConnect()***.

Глобальный ***G2D_FMT_ABGR_AVUY8888*** То же, что ***G2D_FMT_ABGR8888*** (не рекомендуется использовать в новых проектах).

Глобальный ***G2D_FMT_ARGB_AYUV8888*** То же, что ***G2D_FMT_ARGB8888*** (не рекомендуется использовать в новых проектах).

Глобальный ***G2D_FMT_BGRA_VUYA8888*** То же, что ***G2D_FMT_BGRA8888*** (не рекомендуется использовать в новых проектах).

Глобальный ***G2D_FMT_RGBA_YUVA8888*** То же, что ***G2D_FMT_RGBA8888*** (не рекомендуется использовать в новых проектах).

13. Список экспериментальных опций

Глобальный ***pllSet*** (**unsigned int n, unsigned int out, unsigned int frequency**) Функция реализована с ограниченным функционалом и находится в стадии тестирования.

14. Алфавитный указатель групп

14.1. Группы

Полный список групп.

SCI (Camera Sensor Interface)	99
USB	100
Драйвера интерфейсов	101
Мультимедиа	102
Стандартные типы	103
Ядро MULTEX-ARM	104

15. Алфавитный указатель структур данных

15.1. Структуры данных

Структуры данных с их кратким описанием.

<i>blk_cache</i>		106
<i>blk_dev</i>		108
<i>complex</i>		110
<i>date_time</i>		111
<i>device_header</i>	Общий заголовок устройства	113
<i>Display</i>	Структура инициализации графического адаптера	114
<i>div_t</i>	Результат деления с остатком	116
<i>dtcompact</i>	Структура формата Дата / Время (DOS-совместимая)	117
<i>env_var</i>		119
<i>exit_st</i>		120
<i>ffblk</i>	Блок информации для поиска файлов в каталоге	121
<i>FILE</i>	Структура файла на блочном устройстве	123
<i>file_fcb</i>	Блок управления файла	124
<i>g2d_blt</i>		126
<i>g2d_fillrect</i>		128
<i>g2d_image</i>		129
<i>g2d_rect</i>		130
<i>g2d_stretchblt</i>		131
<i>imaxdiv_t</i>	Возврат функции <code>imaxdiv</code> , результат деления	132
<i>in_addr</i>		133
<i>iniBinaryArray</i>	Структура для сохранения массива бинарных данных в ini-файле	134
<i>iniCoords</i>		135
<i>iniIntArray</i>	Структура для сохранения массива целых чисел в ini-файле	136

<i>iniRect</i>		137
<i>ip_packet</i>	Заголовок пакета IP	138
<i>jmp_buf</i>		139
<i>ldiv_t</i>	Результат деления чисел типа long long с остатком	140
<i>listNode</i>		141
<i>msgQID</i>	Структура блока управления очереди	142
<i>REG_SET</i>		144
<i>sDisplayInfo</i>	Описание параметров дисплея	146
<i>seekblk</i>	Блок информации для функции seek	147
<i>Sem_Id</i>	Структура семафора	148
<i>sigaction</i>	Структура обработчика сигнала	150
<i>siginfo</i>	Структура данных сигнала	151
<i>sigval</i>		152
<i>sList</i>		153
<i>sockaddr</i>		154
<i>sockaddr_in</i>	Структура адреса сокета для связи через сеть	155
<i>sTtfFont</i>		156
<i>sVector</i>		157
<i>tagSURFACE</i>	Описание параметров поверхности	158
<i>TCB</i>	Структура блока управления задачей	159
<i>tDrvBit</i>	Описание одного бита регистра	164
<i>tDrvBitGroup</i>	Описание группы бит регистра	165
<i>tDrvGpio</i>	Описание аппаратных линий вывода	166

<i>textRect</i>	Структура прямоугольника	167
<i>timespec</i>	Тики процессора	168
<i>tm</i>		169
<i>tMapIterators</i>	Набор указателей для работы со списками	171
<i>tRingBuffer</i>	Структура кольцевого буфера	172
<i>tScreenDeviceMode</i>	Структура настройки подключения дисплея LCD	173
<i>udp_hdr</i>	Заголовок пакета UDP	175
<i>udp_service</i>		176
<i>usb_class_abstract_control_descriptor</i>		177
<i>usb_class_atm_networking_descriptor</i>		178
<i>usb_class_call_management_descriptor</i>		180
<i>usb_class_capi_control_descriptor</i>		181
<i>usb_class_country_selection_descriptor</i>		182
<i>usb_class_descriptor</i>		183
<i>usb_class_direct_line_descriptor</i>		186
<i>usb_class_ethernet_networking_descriptor</i>		187
<i>usb_class_extension_unit_descriptor</i>		189
<i>usb_class_function_descriptor</i>		190
<i>usb_class_function_descriptor_generic</i>		191
<i>usb_class_header_function_descriptor</i>		192
<i>usb_class_hid_descriptor</i>		193
<i>usb_class_mdIm_descriptor</i>		194
<i>usb_class_mdImd_descriptor</i>		195
<i>usb_class_multi_channel_descriptor</i>		196
<i>usb_class_network_channel_descriptor</i>		197
<i>usb_class_protocol_unit_function_descriptor</i>		198
<i>usb_class_report_descriptor</i>		199
<i>usb_class_telephone_call_descriptor</i>		200

<i>usb_class_telephone_operational_descriptor</i>		201
<i>usb_class_telephone_ringer_descriptor</i>		202
<i>usb_class_union_function_descriptor</i>		203
<i>usb_class_usb_terminal_descriptor</i>		204
<i>usb_config</i>	Структура дескриптора конфигурации	206
<i>usb_configuration_descriptor</i>	Структура дескриптора конфигурации	207
<i>usb_descriptor</i>		209
<i>usb_device</i>	Структура данных о подключении USB-устройства	211
<i>usb_device_descriptor</i>	Структура дескриптора устройства USB	215
<i>usb_endpoint_descriptor</i>	Структура дескриптора конечной точки	218
<i>usb_generic_descriptor</i>		221
<i>usb_interface</i>	Структура дескриптора интерфейса	222
<i>usb_interface_descriptor</i>	Структура дескриптора интерфейса	223
<i>usb_string_descriptor</i>	Структура дескриптора строки	225

16. Список файлов

16.1. Файлы

Полный список файлов.

<i>a20graph.h</i>	Работа с графической подсистемой	226
<i>arch.h</i>	Описание аппаратной части текущего проекта	253
<i>archdef.h</i>	Зарезервированные строки описания аппаратной части	262
<i>assert.h</i>	Механизмы диагностики и проверки	265
<i>avilib.h</i>	Работа с AVI файлами	268
<i>blkcache.h</i>	Кэширование записи / чтения блоков данных	274
<i>cache.h</i>	Методы работы с КЭШ-памятью	278
<i>cedrus.h</i>	Работа с кодером/декодером видео h.264 CEDRUS	281
<i>console.h</i>	Дополнительные функции для работы с текстовым вводом-выводом	288
<i>crc32.h</i>	Имплементация crc32 из GCC	291
<i>crc8.h</i>	Чек-сумма crc8	292
<i>crt.h</i>	Функции для работы с терминалом и клавиатурой, а также заглушки для обратной совместимости	294
<i>csi.h</i>	Работа с интерфейсом CSI (Camera Sensor Interface)	303
<i>ctype.h</i>	Классификация и преобразование отдельных символов	313
<i>datetime.h</i>	Дополнительные функции для работы с датой/временем	321
<i>de2.h</i>	Allwinner DE2	324
<i>env_vars.h</i>	Дополнительные функции для работы с переменными окружения	330
<i>errno-base.h</i>	Заголовочный файл для обратной совместимости	331

<i>errno.h</i>	Обработка причин ошибок в библиотечных функциях	332
<i>filesystem.h</i>	Дополнительные функции для работы с файловой системой	350
<i>fnames.h</i>	Функционал для работы с именами файлов	358
<i>fonts.h</i>	Высокоуровневые интерфейсы для работы с TTF-шрифтами	364
<i>fontdefines.h</i>	Различные общие структуры, перечисления и дефайны для шрифтов	376
<i>gpio.h</i>	Порты ввода/вывода (GPIO)	379
<i>i2c.h</i>	Интерфейс I2C	387
<i>inifiles.h</i>	Методы работы с ini-файлами	393
<i>inputstr.h</i>	Функции для обработки введенных строк и работы с управляющими клавишами	410
<i>intlib.h</i>	Методы управления прерываниями	412
<i>inttypes.h</i>	Расширения для работы с типами заданного размера	416
<i>iolib.h</i>	Работа с базовой системой ввода / вывода	438
<i>iso646.h</i>	Текстовые макросы для символьных операторов Си	466
<i>limits.h</i>	Характеристики общих типов	469
<i>list.h</i>	Не-потокобезопасный двухсвязный список	473
<i>mapstr.h</i>	Список пар типа строка-значение	481
<i>math.h</i>	Стандартные математические функции	487
<i>memlib.h</i>	Управление диспетчером памяти	495
<i>mpeg4codec.h</i>	Программно-аппаратный декодер видео файлов формата MP4. Использует аппаратный видео-енкодер процессора и препроцессор NEON	501

<i>msgqlib.h</i>	Создание очередей сообщений	504
<i>multex.h</i>	Основной подключаемый файл RTOS <i>MULTEX-ARM</i>	510
<i>names.h</i>	Функции для работы с таблицами символов	521
<i>pipelib.h</i>	Межпроцессорные каналы	524
<i>pll.h</i>	Работа с PLL	525
<i>pwm.h</i>	Управление линиями ШИМ (PWM)	532
<i>ringbuffer.h</i>	Кольцевой буфер	536
<i>semlib.h</i>	Управление семафорами	540
<i>setjmp.h</i>	Нелокальные переходы	551
<i>shell.h</i>	Терминал	554
<i>signal.h</i>	Обработка сигналов (C11 + частично POSIX)	556
<i>sleep.h</i>	Различные обертки над функционалом приостановки задачи	569
<i>socket.h</i>	Протокол TCP	573
<i>softgraph.h</i>	Аппаратная реализация работы с поверхностями	582
<i>sound.h</i>	Работа со звуковой подсистемой	601
<i>spi.h</i>	Интерфейс SPI	606
<i>stdarg.h</i>	Макросы для поддержки функций с неопределенным числом аргументов неопределенного типа	613
<i>stdbool.h</i>	Стандартные логические типы данных	615
<i>stddef.h</i>	Стандартные определения	616
<i>stdint.h</i>	Целочисленные типы заданного размера	618

<i>stdio.h</i>	Стандартные функции ввода-вывода	630
<i>stdlib.h</i>	Стандартная библиотека	653
<i>stdnoreturn.h</i>	Определение макроса <code>noreturn</code>	669
<i>string.h</i>	Работа с массивами символов	670
<i>tasklib.h</i>	Управление задачами	693
<i>terminator.h</i>	Обработка корректного закрытия драйверов для горячей перезагрузки ОС	707
<i>time.h</i>	Стандартные манипуляции с датой и временем	708
<i>timer-arm.h</i>	Управление аппаратными таймерами ARM процессоров	715
<i>timer.h</i>	Управление системным таймером	719
<i>tv-decoder.h</i>	Драйвер аппаратного TV-декодера	721
<i>tv-receiver.h</i>	Устройство захвата телевизионного сигнала	729
<i>uart.h</i>	Драйвер UART. Поточковая передача данных	733
<i>uchar.h</i>	Unicode utilities from C11	745
<i>udp.h</i>	Сетевой протокол UDP	746
<i>unicode.h</i>	Преобразование строк и символов между различными кодировками (UTF-16, UTF-8, Cp1251, Cp866, Ascii)	752
<i>usb.h</i>	Методы работы с USB	758
<i>usb_driver.h</i>	Регистрация USB драйвера в <i>MULTEX-ARM</i>	774
<i>usbdescriptors.h</i>	Структуры дескрипторов USB	776
<i>vdisk.h</i>	Механизмы монтирования виртуального тома	786

vector.h

Работа с массивами данных типа Вектор

787

17. Группы

17.1. SCI (Camera Sensor Interface)

Файлы

- файл [csi.h](#)
Работа с интерфейсом CSI (Camera Sensor Interface)

17.1.1. Подробное описание

17.2. USB

Файлы

- файл *usb.h*
Методы работы с USB.
- файл *usb_driver.h*
Регистрация USB драйвера в MULTEX-ARM.
- файл *usbdescriptors.h*
Структуры дескрипторов USB.

17.2.1. Подробное описание

См. также

Подсистема USB.

17.3. Драйвера интерфейсов

Файлы

- файл *gpio.h*
Порты ввода/вывода (GPIO).
- файл *i2c.h*
Интерфейс I2C.
- файл *pwm.h*
Управление линиями ШИМ (PWM).
- файл *spi.h*
Интерфейс SPI.
- файл *uart.h*
Драйвер UART. Поточковая передача данных.
- файл *udp.h*
Сетевой протокол UDP.

17.3.1. Подробное описание

17.4. Мультимедиа

Файлы

- файл *a20graph.h*
Работа с графической подсистемой.
- файл *avilib.h*
Работа с AVI файлами.
- файл *cedrus.h*
Работа с кодером/декодером видео h.264 CEDRUS.
- файл *de2.h*
Allwinner DE2.
- файл *fonts.h*
Высокоуровневые интерфейсы для работы с TTF-шрифтами.
- файл *fontdefines.h*
Различные общие структуры, перечисления и дефайны для шрифтов.
- файл *mpeg4codec.h*
*Программно-аппаратный декодер видео файлов формата MP4. Использует аппаратный видео-енкодер процессора и препроцессор **NEON**.*
- файл *softgraph.h*
Аппаратная реализация работы с поверхностями.
- файл *sound.h*
Работа со звуковой подсистемой.
- файл *tv-decoder.h*
Драйвер аппаратного TV-декодера.
- файл *tv-receiver.h*
Устройство захвата телевизионного сигнала.

17.4.1. Подробное описание

Мультимедиа (англ. multimedia) — данные, или содержание, которые представляются одновременно в разных формах: звук, анимированная компьютерная графика, видеоряд.

17.5. Стандартные типы

Файлы

- файл *assert.h*
Механизмы диагностики и проверки.
- файл *ctype.h*
Классификация и преобразование отдельных символов.
- файл *errno-base.h*
Заголовочный файл для обратной совместимости.
- файл *errno.h*
Обработка причин ошибок в библиотечных функциях.
- файл *inttypes.h*
Расширения для работы с типами заданного размера.
- файл *iso646.h*
Текстовые макросы для символьных операторов Си.
- файл *limits.h*
Характеристики общих типов.
- файл *list.h*
Не-потокбезопасный двухсвязный список.
- файл *math.h*
Стандартные математические функции
- файл *stdarg.h*
Макросы для поддержки функций с неопределенным числом аргументов неопределенного типа.
- файл *stddef.h*
Стандартные определения.
- файл *stdint.h*
Целочисленные типы заданного размера.
- файл *stdio.h*
Стандартные функции ввода-вывода.
- файл *stdlib.h*
Стандартная библиотека.
- файл *stdnoreturn.h*
Определение макроса noreturn.
- файл *string.h*
Работа с массивами символов.
- файл *time.h*
Стандартные манипуляции с датой и временем.
- файл *vector.h*
*Работа с массивами данных типа **Вектор**.*

17.5.1. Подробное описание

В файлах группы содержатся описания типов стандарта C11 7.4, реализованные в *MULTEX-ARM*.

17.6. Ядро MULTEX-ARM

Файлы

- файл *arch.h*
Описание аппаратной части текущего проекта.
- файл *archdef.h*
Зарезервированные строки описания аппаратной части.
- файл *cache.h*
Методы работы с КЭШ-памятью.
- файл *console.h*
Дополнительные функции для работы с текстовым вводом-выводом.
- файл *crc32.h*
Имплементация crc32 из GCC.
- файл *crc8.h*
Чек-сумма crc8.
- файл *crt.h*
Функции для работы с терминалом и клавиатурой, а также заглушки для обратной совместимости.
- файл *datetime.h*
Дополнительные функции для работы с датой/временем.
- файл *env_vars.h*
Дополнительные функции для работы с переменными окружения.
- файл *filesyst.h*
Дополнительные функции для работы с файловой системой.
- файл *fnames.h*
Функционал для работы с именами файлов.
- файл *inifiles.h*
Методы работы с ini-файлами.
- файл *inputstr.h*
Функции для обработки введенных строк и работы с управляющими клавишами.
- файл *intlib.h*
Методы управления прерываниями.
- файл *iolib.h*
Работа с базовой системой ввода / вывода.
- файл *mapstr.h*
*Список пар типа **строка-значение**.*
- файл *memlib.h*
Управление диспетчером памяти.
- файл *msgqlib.h*
Создание очередей сообщений.
- файл *multex.h*
*Основной подключаемый файл **RTOS MULTEX-ARM**.*
- файл *names.h*
Функции для работы с таблицами символов.
- файл *pipelib.h*
Межпроцессорные каналы.
- файл *pll.h*
Работа с PLL.
- файл *ringbuffer.h*
Кольцевой буфер.
- файл *semlib.h*
Управление семафорами.
- файл *setjmp.h*
Нелокальные переходы.
- файл *shell.h*
Терминал.
- файл *signal.h*
*Обработка сигналов (**C11** + частично **POSIX**).*

- файл *sleep.h*
Различные обертки над функционалом приостановки задачи.
- файл *tasklib.h*
Управление задачами.
- файл *terminator.h*
Обработка корректного закрытия драйверов для горячей перезагрузки ОС.
- файл *timer-arm.h*
Управление аппаратными таймерами ARM процессоров.
- файл *timer.h*
Управление системным таймером.
- файл *unicode.h*
Преобразование строк и символов между различными кодировками (UTF-16, UTF-8, Cp1251, Cp866, Ascii).
- файл *vdisk.h*
Механизмы монтирования виртуального тома.

17.6.1. Подробное описание

18. Структуры данных

18.1. Структура `blk_cache`

Поля данных

- `int BlkSize`
- `char * BlkStat`
- `char * Cache`
- `char * Cache_na`
- `int CacheIdx`
- `int cacheOff`
- `int CacheSize`
- `void * hDrv`
- `blk_cache_proc read`
- `blk_cache_proc write`

18.1.1. Подробное описание

Структура данных организации КЭШ-памяти.

18.1.2. Поля

18.1.2.1. `BlkSize`

`int BlkSize`

Размер блока памяти.

18.1.2.2. `BlkStat`

`char* BlkStat`

Карта состояния КЭШ.

18.1.2.3. `Cache`

`char* Cache`

КЭШ-память.

18.1.2.4. `Cache_na`

`char* Cache_na`

Невыровненная на границу страницы (для free).

18.1.2.5. `CacheIdx`

`int CacheIdx`

Номер первого блока в КЭШ.

18.1.2.6. cacheOff

int cacheOff

18.1.2.7. CacheSize

int CacheSize

Размер КЭШ-памяти в блоках.

18.1.2.8. hDrv

void* hDrv

Указатель на драйвер.

18.1.2.9. read

blk_cache_proc read

Процедура чтения драйвера.

18.1.2.10. write

blk_cache_proc write

Процедура записи драйвера.

Объявления и описания членов структуры находятся в файле:

- *blkcache.h*

18.2. Структура `blk_dev`

Поля данных

- struct `blk_dev` int int `arg`
- int `bd_blkTotal`
- int `bd_bytesPerBlk`
- BOOL `bd_catSaved`
- void * `bd_rootCat`
- UINT `bd_signature`
- int `bd_startBlk`
- int `bd_volume`
- void * `fsData`
- struct `blk_dev` int `funcCode`
- struct `blk_dev` int int `numBlks`
- struct `blk_dev` int int char * `pBuf`
- struct `blk_dev` * `pDev`
- struct `blk_dev` int `startBlk`
- `DEVICE` * `volConfig`

18.2.1. Поля

18.2.1.1. `arg`

struct `blk_dev` int int `arg`

18.2.1.2. `bd_blkTotal`

int `bd_blkTotal`

18.2.1.3. `bd_bytesPerBlk`

int `bd_bytesPerBlk`

18.2.1.4. `bd_catSaved`

BOOL `bd_catSaved`

18.2.1.5. `bd_rootCat`

void* `bd_rootCat`

18.2.1.6. `bd_signature`

UINT `bd_signature`

18.2.1.7. bd_startBlk

int bd_startBlk

18.2.1.8. bd_volume

int bd_volume

18.2.1.9. fsData

void* fsData

18.2.1.10. funcCode

struct *blk_dev* int funcCode

18.2.1.11. numBlks

struct *blk_dev* int int numBlks

18.2.1.12. pBuf

struct *blk_dev* int int char * pBuf

18.2.1.13. pDev

struct *blk_dev* * pDev

18.2.1.14. startBlk

struct *blk_dev* int startBlk

18.2.1.15. volConfig

*DEVICE** volConfig

Объявления и описания членов структуры находятся в файле:

- *iolib.h*

18.3. Структура `complex`

Поля данных

- `double x`
- `double y`

18.3.1. Поля

18.3.1.1. `x`

`double x`

18.3.1.2. `y`

`double y`

Объявления и описания членов структуры находятся в файле:

- `math.h`

18.4. Структура date_time

Поля данных

- int *Day*
Дни. Первый день = 1.
- int *Hour*
Часы 0-23.
- int *Minute*
Минуты 0-59.
- int *Month*
Месяцы. Январь = 1.
- int *Second*
Секунды 0-59.
- int *Year*
Годы.

18.4.1. Подробное описание

Структура формата Дата / Время.

18.4.2. Поля

18.4.2.1. Day

int Day

18.4.2.2. Hour

int Hour

18.4.2.3. Minute

int Minute

18.4.2.4. Month

int Month

18.4.2.5. Second

int Second

18.4.2.6. Year

int Year

Объявления и описания членов структуры находятся в файле:

- *datetime.h*

18.5. Структура `device_header`

Общий заголовок устройства.

Поля данных

- `DCB * devDCB`
- `char * devName`
- `int drvNumber`
- `struct device_header * next`

18.5.1. Подробное описание

Структура общего заголовка устройства.

18.5.2. Поля

18.5.2.1. `devDCB`

`DCB* devDCB`

Данные, зависящие от у-ва.

18.5.2.2. `devName`

`char* devName`

Символьное имя устройства.

18.5.2.3. `drvNumber`

`int drvNumber`

Номер драйвера в таблице.

18.5.2.4. `next`

`struct device_header* next`

Следующее у-во в цепочке.

Объявления и описания членов структуры находятся в файле:

- `iolib.h`

18.6. Структура Display

Структура инициализации графического адаптера.

Поля данных

- int *BPP*
Bit Per Pixel.
- void * *Constr*
Указатель на конструируемую область.
- int *DSize*
Размер видимой области в байтах.
- int *Height*
Высота экрана в пикселях.
- int *interface*
Интерфейс: HDMI, LCD, LVDS.
- int *LFB*
Адрес начала видеопамяти.
- struct fb_videomode *mode*
Настройки растра.
- char * *modeline*
Строка-описатель (Xfree86 style modeline).
- void * *Screen*
Указатель на отображаемую область.
- int *VideoMode*
Видео режим из набора lvds_param_t.
- int *Width*
Ширина экрана в пикселях.

18.6.1. Поля

18.6.1.1. BPP

int BPP

18.6.1.2. Constr

void* Constr

18.6.1.3. DSize

int DSize

18.6.1.4. Height

int Height

18.6.1.5. interface

int interface

18.6.1.6. LFB

int LFB

18.6.1.7. mode

struct fb_videomode mode

18.6.1.8. modeline

char* modeline

18.6.1.9. Screen

void* Screen

18.6.1.10. VideoMode

int VideoMode

18.6.1.11. Width

int Width

Объявления и описания членов структуры находятся в файле:

- *a20graph.h*

18.7. Структура `div_t`

Результат деления с остатком.

Поля данных

- `int quot`
- `int rem`

18.7.1. Подробное описание

Результат деления с остатком.

18.7.2. Поля

18.7.2.1. `quot`

`int quot`

Частное от деления.

18.7.2.2. `rem`

`int rem`

Остаток от деления.

Объявления и описания членов структуры находятся в файле:

- `stdlib.h`

18.8. Структура dtcompact

Структура формата Дата / Время (DOS-совместимая).

Поля данных

- UINT *Day*:5
- UINT *Hour*:5
- UINT *Minute*:6
- UINT *Month*:4
- UINT *Sec2*:5
- UINT *Year*:7

18.8.1. Подробное описание

Компактное представление даты / времени совместимое с DOS.

18.8.2. Поля

18.8.2.1. Day

UINT Day

Дни.

18.8.2.2. Hour

UINT Hour

Часы.

18.8.2.3. Minute

UINT Minute

Минуты.

18.8.2.4. Month

UINT Month

Месяцы.

18.8.2.5. Sec2

UINT Sec2

Пары секунд (значение 1 обозначает 2 секунды).

18.8.2.6. Year

UINT Year

Годы.

Объявления и описания членов структуры находятся в файле:

- *datetime.h*

18.9. Структура `env_var`

Поля данных

- `char * name`
- `struct env_var * next`
- `char * val`

18.9.1. Поля

18.9.1.1. `name`

`char* name`

18.9.1.2. `next`

`struct env_var* next`

18.9.1.3. `val`

`char* val`

Объявления и описания членов структуры находятся в файле:

- `env_vars.h`

18.10. Структура `exit_st`

Поля данных

- `void(* exit_fun)()`
- `struct exit_st * next`

18.10.1. Подробное описание

Структура элемента списка функций-обработчиков, вызываемых при завершении задачи при помощи `exit()` и `abort()`.

18.10.2. Поля

18.10.2.1. `exit_fun`

```
void(* exit_fun )()
```

18.10.2.2. `next`

```
struct exit_st* next
```

Объявления и описания членов структуры находятся в файле:

- `tasklib.h`

18.11. Структура `ffblk`

Блок информации для поиска файлов в каталоге.

Поля данных

- `int ff_attrib`
Атрибуты файла.
- `DT_COMPACT ff_date_time`
Время создания файла.
- `void * ff_directory`
Указатель на каталог.
- `char ff_dname [13]`
Имя устройства.
- `int ff_fsize`
Размер файла.
- `int ff_idx`
Индекс файла в каталоге.
- `char ff_lname [128]`
Длинное имя файла.
- `char ff_mask [13]`
Маска имени файла.
- `char ff_name [13]`
Имя файла.
- `char * ff_path`
Полный путь в каталог.

18.11.1. Подробное описание

Структура данных блока поиска файлов в каталоге.

18.11.2. Поля

18.11.2.1. `ff_attrib`

`int ff_attrib`

18.11.2.2. `ff_date_time`

`DT_COMPACT ff_date_time`

18.11.2.3. `ff_directory`

`void* ff_directory`

18.11.2.4. `ff_dname`

`char ff_dname[13]`

18.11.2.5. ff_size

int ff_size

18.11.2.6. ff_idx

int ff_idx

18.11.2.7. ff_lname

char ff_lname[128]

18.11.2.8. ff_mask

char ff_mask[13]

18.11.2.9. ff_name

char ff_name[13]

18.11.2.10. ff_path

char* ff_path

Объявления и описания членов структуры находятся в файле:

- *iolib.h*

18.12. Структура FILE

Структура файла на блочном устройстве.

Поля данных

- int *fd*
Дескриптор файла.
- int *signa*
Сигнатура файла.
- int *ugf*
Флаг ungetc.
- char *unget*
Буфер ungetc.

18.12.1. Поля

18.12.1.1. fd

int fd

18.12.1.2. signa

int signa

18.12.1.3. ugf

int ugf

18.12.1.4. unget

char unget

Объявления и описания членов структуры находятся в файле:

- *stdio.h*

18.13. Структура file_fcb

Блок управления файла.

Поля данных

- char * *blkBuf*
- int *blkNum*
- int *catIndex*
- *DEV_HDR* * *devHdr*
- int *eof*
- char * *fileName*
- unsigned int *fileSize*
- int *flushed*
- int *mode*
- void * *paramBlk*
- unsigned int *position*
- int *startBlk*

18.13.1. Подробное описание

Структура блока управления файла.

18.13.2. Поля

18.13.2.1. blkBuf

char* blkBuf

Для блочных устройств - буфер блока.

18.13.2.2. blkNum

int blkNum

Номер блока, содержащегося в буфере.

18.13.2.3. catIndex

int catIndex

Порядковый номер файла в каталоге.

18.13.2.4. devHdr

*DEV_HDR** devHdr

Указатель на заголовок устройства.

18.13.2.5. eof

int eof

Признак конца файла.

18.13.2.6. fileName

char* fileName

18.13.2.7. fileSize

unsigned int fileSize

18.13.2.8. flushed

int flushed

Блок сохранен на диске.

18.13.2.9. mode

int mode

18.13.2.10. paramBlk

void* paramBlk

Указатель на блок параметров для файла.

18.13.2.11. position

unsigned int position

18.13.2.12. startBlk

int startBlk

Стартовый блок файла.

Объявления и описания членов структуры находятся в файле:

- [*iolib.h*](#)

18.14. Структура `g2d_blt`

Поля данных

- `__u32 alpha`
- `__u32 color`
- `g2d_image dst_image`
- `__s32 dst_x`
- `__s32 dst_y`
- `g2d_blt_flags flag`
- `g2d_image src_image`
- `g2d_rect src_rect`

18.14.1. Поля

18.14.1.1. `alpha`

`__u32 alpha`

Plane alpha value.

18.14.1.2. `color`

`__u32 color`

Colorkey color.

18.14.1.3. `dst_image`

`g2d_image dst_image`

18.14.1.4. `dst_x`

`__s32 dst_x`

Left top point coordinate x of dst rect.

18.14.1.5. `dst_y`

`__s32 dst_y`

Left top point coordinate y of dst rect.

18.14.1.6. `flag`

`g2d_blt_flags flag`

18.14.1.7. `src_image`

`g2d_image src_image`

18.14.1.8. `src_rect`

`g2d_rect src_rect`

Объявления и описания членов структуры находятся в файле:

- `a20graph.h`

18.15. Структура `g2d_fillrect`

Поля данных

- `__u32 alpha`
- `__u32 color`
- `g2d_image dst_image`
- `g2d_rect dst_rect`
- `g2d_fillrect_flags flag`

18.15.1. Поля

18.15.1.1. `alpha`

`__u32 alpha`

Plane alpha value.

18.15.1.2. `color`

`__u32 color`

Fill color.

18.15.1.3. `dst_image`

`g2d_image dst_image`

18.15.1.4. `dst_rect`

`g2d_rect dst_rect`

18.15.1.5. `flag`

`g2d_fillrect_flags flag`

Объявления и описания членов структуры находятся в файле:

- `a20graph.h`

18.16. Структура `g2d_image`

Поля данных

- `__u32 addr` [3]
- `g2d_data_fmt format`
- `__u32 h`
- `g2d_pixel_seq pixel_seq`
- `__u32 w`

18.16.1. Подробное описание

`image struct`

18.16.2. Поля

18.16.2.1. `addr`

`__u32 addr`[3]

Base `addr` of image frame buffer in byte.

18.16.2.2. `format`

`g2d_data_fmt format`

Pixel format of image frame buffer.

18.16.2.3. `h`

`__u32 h`

Height of image frame buffer in pixel.

18.16.2.4. `pixel_seq`

`g2d_pixel_seq pixel_seq`

Pixel sequence of image frame buffer.

18.16.2.5. `w`

`__u32 w`

Width of image frame buffer in pixel.

Объявления и описания членов структуры находятся в файле:

- `a20graph.h`

18.17. Структура `g2d_rect`

Поля данных

- `__u32 h`
- `__u32 w`
- `__s32 x`
- `__s32 y`

18.17.1. Подробное описание

Flip rectangle struct.

18.17.2. Поля

18.17.2.1. `h`

`__u32 h`

Rectangle height.

18.17.2.2. `w`

`__u32 w`

Rectangle width.

18.17.2.3. `x`

`__s32 x`

Left top point coordinate `x`.

18.17.2.4. `y`

`__s32 y`

Left top point coordinate `y`.

Объявления и описания членов структуры находятся в файле:

- `a20graph.h`

18.18. Структура `g2d_stretchblt`

Поля данных

- `__u32 alpha`
- `__u32 color`
- `g2d_image dst_image`
- `g2d_rect dst_rect`
- `g2d_blt_flags flag`
- `g2d_image src_image`
- `g2d_rect src_rect`

18.18.1. Поля

18.18.1.1. `alpha`

`__u32 alpha`

Plane alpha value.

18.18.1.2. `color`

`__u32 color`

Colorkey color.

18.18.1.3. `dst_image`

`g2d_image dst_image`

18.18.1.4. `dst_rect`

`g2d_rect dst_rect`

18.18.1.5. `flag`

`g2d_blt_flags flag`

18.18.1.6. `src_image`

`g2d_image src_image`

18.18.1.7. `src_rect`

`g2d_rect src_rect`

Объявления и описания членов структуры находятся в файле:

- `a20graph.h`

18.19. Структура `imaxdiv_t`

Возврат функции `imaxdiv`, результат деления.

Поля данных

- `intmax_t quot`
Частное.
- `intmax_t rem`
Остаток.

18.19.1. Поля

18.19.1.1. `quot`

`intmax_t quot`

18.19.1.2. `rem`

`intmax_t rem`

Объявления и описания членов структуры находятся в файле:

- `inttypes.h`

18.20. Структура `in_addr`

Поля данных

- `UINT s_addr`

18.20.1. Подробное описание

Структура адреса **Internet**.

18.20.2. Поля

18.20.2.1. `s_addr`

`UINT s_addr`

Объявления и описания членов структуры находятся в файле:

- `socket.h`

18.21. Структура `iniBinaryArray`

Структура для сохранения массива бинарных данных в `ini`-файле.

Поля данных

- `uint8_t * Array`
- `size_t ArrayLength`

18.21.1. Подробное описание

сохраняются с указанием длины массива.



Данная структура должна освобождаться через два `free()` - для поля **Array** и для самой структуры.

18.21.2. Поля

18.21.2.1. Array

`uint8_t* Array`

18.21.2.2. ArrayLength

`size_t ArrayLength`

Объявления и описания членов структуры находятся в файле:

- `inifiles.h`

18.22. Структура iniCoords

Поля данных

- int *x*
- int *y*

18.22.1. Подробное описание

Структура данных для сохранения пары координат.

18.22.2. Поля

18.22.2.1. *x*

int *x*

X-координата

18.22.2.2. *y*

int *y*

Y-координата

Объявления и описания членов структуры находятся в файле:

- *inifiles.h*

18.23. Структура `iniIntArray`

Структура для сохранения массива целых чисел в `ini`-файле.

Поля данных

- `int * Array`
- `int ArrayLength`

18.23.1. Подробное описание

Массивы целых чисел сохраняются с указанием длины массива.



Данная структура должна освобождаться через два `free()` - для поля **Array** и для самой структуры. *

18.23.2. Поля

18.23.2.1. Array

`int* Array`

Указатель на данные.

18.23.2.2. ArrayLength

`int ArrayLength`

Длина массива.

Объявления и описания членов структуры находятся в файле:

- `inifiles.h`

18.24. Структура iniRect

Поля данных

- int *h*
- int *w*
- int *x*
- int *y*

18.24.1. Подробное описание

Структура хранения размеров прямоугольной

18.24.2. Поля

18.24.2.1. h

int h

Высота прямоугольника.

18.24.2.2. w

int w

Ширина прямоугольника.

18.24.2.3. x

int x

X-координата.

18.24.2.4. y

int y

Y-координата.

Объявления и описания членов структуры находятся в файле:

- *inifiles.h*

18.25. Структура ip_packet

Заголовок пакета IP.

Поля данных

- unsigned int *d_addr*
Адрес отправителя.
- char * *data*
Указатель на заголовок полученных данных UDP_HDR.
- unsigned int *length*
Длина пакета IP.
- unsigned char *protocol*
- unsigned int *s_addr*
Адрес отправителя.

18.25.1. Подробное описание

Заголовок пакета IP. Порядок следования байт – младшим байтом вперёд.

18.25.2. Поля

18.25.2.1. d_addr

unsigned int d_addr

18.25.2.2. data

char* data

18.25.2.3. length

unsigned int length

18.25.2.4. protocol

unsigned char protocol

18.25.2.5. s_addr

unsigned int s_addr

Объявления и описания членов структуры находятся в файле:

- *udp.h*

18.26. Структура jmp_buf

Поля данных

- *REG_SET REGS*

18.26.1. Подробное описание

Тип, описывающий информацию необходимую для нелокального перехода и позволяющий сохранить контекст.

18.26.2. Поля

18.26.2.1. REGS

REG_SET REGS

Объявления и описания членов структуры находятся в файле:

- *setjmp.h*

18.27. Структура `ldiv_t`

Результат деления чисел типа `long long` с остатком.

Поля данных

- `long quot`
- `long rem`

18.27.1. Подробное описание

Результат деления чисел типа `long long` с остатком.

18.27.2. Поля

18.27.2.1. `quot`

`long quot`

Частное от деления.

18.27.2.2. `rem`

`long rem`

Остаток от деления.

Объявления и описания членов структуры находятся в файле:

- `stdlib.h`

18.28. Структура `listNode`

Поля данных

- `void * pData`
Указатель на данные в узле.
- `struct listNode * pNext`
Указатель на следующий узел.
- `struct listNode * pPrev`
Указатель на предыдущий узел.

18.28.1. Подробное описание

Узел двухсвязного списка.

18.28.2. Поля

18.28.2.1. `pData`

`void* pData`

18.28.2.2. `pNext`

`struct listNode* pNext`

18.28.2.3. `pPrev`

`struct listNode* pPrev`

Объявления и описания членов структуры находятся в файле:

- `list.h`

18.29. Структура msgQID

Структура блока управления очереди.

Поля данных

- int *count*
- int *F*
- char * *first*
- char * *last*
- char * *p_rd*
- char * *p_wr*
- int *PW_Id*
- *SEM_ID Sem_R*
- *SEM_ID Sem_W*
- int *size*

18.29.1. Поля

18.29.1.1. count

int count

Максимальное число сообщений.

18.29.1.2. F

int F

Опции очереди.

18.29.1.3. first

char* first

Указатель на первый элемент.

18.29.1.4. last

char* last

Указатель на последний эл-т.

18.29.1.5. p_rd

char* p_rd

Указатель чтения.

18.29.1.6. p_wr

char* p_wr

Указатель записи.

18.29.1.7. PW_Id

int PW_Id

Сигнатура очереди.

18.29.1.8. Sem_R

SEM_ID Sem_R

Семафор чтения.

18.29.1.9. Sem_W

SEM_ID Sem_W

Семафор записи.

18.29.1.10. size

int size

Размер сообщений.

Объявления и описания членов структуры находятся в файле:

- *msgqlib.h*

18.30. Структура REG_SET

Поля данных

- int *fp*
- int *lr*
- int *pc*
- int *r10*
- int *r2*
- int *r3*
- int *r4*
- int *r5*
- int *r6*
- int *r7*
- int *r8*
- int *r9*
- int *sp*

18.30.1. Подробное описание

Тип, описывающий набор регистров ARMv7.

18.30.2. Поля

18.30.2.1. fp

int fp

18.30.2.2. lr

int lr

18.30.2.3. pc

int pc

18.30.2.4. r10

int r10

18.30.2.5. r2

int r2

18.30.2.6. r3

int r3

18.30.2.7. r4

int r4

18.30.2.8. r5

int r5

18.30.2.9. r6

int r6

18.30.2.10. r7

int r7

18.30.2.11. r8

int r8

18.30.2.12. r9

int r9

18.30.2.13. sp

int sp

Объявления и описания членов структуры находятся в файле:

- *setjmp.h*

18.31. Структура sDisplayInfo

Описание параметров дисплея.

Поля данных

- int *bpp*
Количество бит на пиксель.
- int *height*
Высота экрана в пикселях.
- int *width*
Ширина экрана в пикселях.

18.31.1. Поля

18.31.1.1. bpp

int bpp

18.31.1.2. height

int height

18.31.1.3. width

int width

Объявления и описания членов структуры находятся в файле:

- *softgraph.h*

18.32. Структура `seekblk`

Блок информации для функции `seek`.

Поля данных

- `int seektype`
- `int shift`

18.32.1. Подробное описание

Структура данных блока информации функции `seek`.

18.32.2. Поля

18.32.2.1. `seektype`

`int seektype`

18.32.2.2. `shift`

`int shift`

Объявления и описания членов структуры находятся в файле:

- `iolib.h`

18.33. Структура Sem_Id

Структура семафора.

Поля данных

- int *count*
- *SEM_FLUSH_STATE* *flushed*
- int *marker*
- int *options*
- *taskId* * *owner*
- int *ownpri*
- *SEM_CLASS* *semclass*
- *SEM_B_STATE* *state*

18.33.1. Подробное описание

Структура семафора.

18.33.2. Поля

18.33.2.1. count

int count

Счетчик рекурсий (для мьютексов) или захватов (для счетчика).

18.33.2.2. flushed

SEM_FLUSH_STATE flushed

Устаревшее поле, на данный момент не используется.

18.33.2.3. marker

int marker

Маркер-сигнатура семафора.

18.33.2.4. options

int options

Опции семафора.

18.33.2.5. owner

*taskId** owner

Захватившая задача.

18.33.2.6. ownpri

int ownpri

Приоритет захватившей задачи.

18.33.2.7. semclass

SEM_CLASS semclass

Класс семафора.

18.33.2.8. state

SEM_B_STATE state

Положение семафора.

Объявления и описания членов структуры находятся в файле:

- *semlib.h*

18.34. Структура sigaction

Структура обработчика сигнала.

Поля данных

- `int sa_flags`
- `void(* sa_handler)(int)`
- `sigset_t sa_mask`
- `void(* sa_sigaction)(int, siginfo_t *, void *)`

18.34.1. Подробное описание

Структура обработчика сигнала.

18.34.2. Поля

18.34.2.1. sa_flags

`int sa_flags`

Флаги, влияющие на поведение сигнала.

18.34.2.2. sa_handler

`void(* sa_handler)(int)`

Функция обработчика.

18.34.2.3. sa_mask

`sigset_t sa_mask`

Сигналы, блокир-ся во время обработки сигнала.

См. также

Функции создания наборов сигналов для поля sa_mask.

18.34.2.4. sa_sigaction

`void(* sa_sigaction)(int, siginfo_t *, void *)`

Указатель на обработчик сигнала.

Объявления и описания членов структуры находятся в файле:

- `signal.h`

18.35. Структура `siginfo`

Структура данных сигнала.

Поля данных

- `int si_code`
- `int si_signo`
- `union sigval si_value`

18.35.1. Подробное описание

Структура данных сигнала.

18.35.2. Поля

18.35.2.1. `si_code`

`int si_code`

Код сигнала.

18.35.2.2. `si_signo`

`int si_signo`

Номер передаваемого сигнала.

18.35.2.3. `si_value`

`union sigval si_value`

Значение сигнала.

Объявления и описания членов структуры находятся в файле:

- `signal.h`

18.36. Объединение sigval

Поля данных

- int *sival_int*
- void * *sival_ptr*

18.36.1. Поля

18.36.1.1. sival_int

int sival_int

18.36.1.2. sival_ptr

void* sival_ptr

Объявления и описания членов объединения находятся в файле:

- *signal.h*

18.37. Структура sList

Поля данных

- `size_t NumOfNodes`
Кол-во узлов в списке.
- `sListNode * pFirst`
Указатель на первый узел в списке.
- `sListNode * pLast`
Указатель на последний узел в списке.

18.37.1. Подробное описание

Двухсвязный список.

18.37.2. Поля

18.37.2.1. NumOfNodes

`size_t NumOfNodes`

18.37.2.2. pFirst

`sListNode* pFirst`

18.37.2.3. pLast

`sListNode* pLast`

Объявления и описания членов структуры находятся в файле:

- `list.h`

18.38. Структура `sockaddr`

Поля данных

- `UCHAR sa_data [14]`
- `sa_family_t sa_family`

18.38.1. Подробное описание

Обобщенная структура адреса сокета.

18.38.2. Поля

18.38.2.1. `sa_data`

`UCHAR sa_data[14]`

Адрес сокета.

18.38.2.2. `sa_family`

`sa_family_t sa_family`

Семейство адресов.

Объявления и описания членов структуры находятся в файле:

- `socket.h`

18.39. Структура `sockaddr_in`

Структура адреса сокета для связи через сеть.

Поля данных

- struct `in_addr sin_addr`
IP - адрес.
- `sa_family_t sin_family`
Семейство адресов.
- `in_port_t sin_port`
Номер порта.
- unsigned char `sin_zero` [8]
Поле выравнивания.

18.39.1. Подробное описание

Структура адреса сокета для связи через сеть.



В отличие от стандарта параметр `sin_port` в структуре `sockaddr_in` требуется задавать в формате **LITTLE ENDIAN**, т.е. не требуется использовать макрос `htons` при его задании.

18.39.2. Поля

18.39.2.1. `sin_addr`

```
struct in_addr sin_addr
```

18.39.2.2. `sin_family`

```
sa_family_t sin_family
```

18.39.2.3. `sin_port`

```
in_port_t sin_port
```

18.39.2.4. `sin_zero`

```
unsigned char sin_zero[8]
```

Объявления и описания членов структуры находятся в файле:

- `socket.h`

18.40. Структура sTtfFont

Поля данных

- unsigned int *FontColor*
- *eTtfFontOptions* *FontOptions*
- unsigned int *FontPixelSize*
- unsigned int *FontSize*
- *pTtfPrivateFontStruct* *PrivateStructPointer*

18.40.1. Подробное описание

Структура шрифта.

18.40.2. Поля

18.40.2.1. FontColor

unsigned int *FontColor*

Цвет шрифта в RGB-формате.

18.40.2.2. FontOptions

eTtfFontOptions *FontOptions*

Опции шрифта, являются комбинацией перечислений *eTtfFontOptions*.

18.40.2.3. FontPixelSize

unsigned int *FontPixelSize*

Размер шрифта в пикселях (фактически высота самой высокого символа от базовой линии).

18.40.2.4. FontSize

unsigned int *FontSize*

Размер шрифта в пунктах.

18.40.2.5. PrivateStructPointer

pTtfPrivateFontStruct *PrivateStructPointer*

Инкапсулированные параметры шрифта.

Объявления и описания членов структуры находятся в файле:

- *fonts.h*

18.41. Структура sVector

Поля данных

- `size_t` *CurrentElementsAmount*
- `void *` *Data*
- `size_t` *ElementSize*
- `size_t` *MaxElementsAmount*

18.41.1. Подробное описание

Структура вектора - динамически расширяющегося массива.

18.41.2. Поля

18.41.2.1. CurrentElementsAmount

`size_t` *CurrentElementsAmount*

Текущее кол-во элементов в векторе.

18.41.2.2. Data

`void*` *Data*

Указатель на блок памяти с данными.

18.41.2.3. ElementSize

`size_t` *ElementSize*

Размер элемента.

18.41.2.4. MaxElementsAmount

`size_t` *MaxElementsAmount*

Максимальное кол-во элементов в векторе (может быть автоматически увеличено при добавлении элементов).

Объявления и описания членов структуры находятся в файле:

- *vector.h*

18.42. Структура tagSURFACE

Описание параметров поверхности.

Поля данных

- unsigned int *sfBPP*
- void * *sfData*
- unsigned int *sfHeight*
- unsigned int *sfWidth*
- unsigned int *sfX*
- unsigned int *sfY*

18.42.1. Поля

18.42.1.1. sfBPP

unsigned int sfBPP

18.42.1.2. sfData

void* sfData

18.42.1.3. sfHeight

unsigned int sfHeight

18.42.1.4. sfWidth

unsigned int sfWidth

18.42.1.5. sfX

unsigned int sfX

18.42.1.6. sfY

unsigned int sfY

Объявления и описания членов структуры находятся в файле:

- *softgraph.h*

18.43. Структура TCB

Структура блока управления задачей.

Поля данных

- `struct TCB * child`
Дочерняя задача.
- `int childstatus`
Статус завершения дочерней задачи.
- `int cursp`
Адрес вершины стека.
- `int delay`
Значение ожидания в тиках.
- `int exit_code`
Код выхода.
- `exit_proc * exit_list`
Список функций-обработчиков, вызываемых по завершении задачи.
- `jmp_buf exitbuf`
Точка выхода задачи.
- `int flags`
Служебные флаги задачи.
- `unsigned intCounter`
Счетчик вложенных прерываний, исполняющихся в контексте задачи.
- `int marker`
Маркер-сигнатура задачи.
- `char * name`
Имя задачи.
- `struct TCB * Next`
Указатель для обслуживания списка задач.
- `struct TCB * parent`
Родительский процесс.
- `struct TCB * Prev`
Указатель для обслуживания списка задач.
- `int priority`
Приоритет задачи (0 - 255), чем меньше, тем приоритетнее.
- `int ps_sp`
Счетчик служебного стека.
- `int ps_stack [TASK_PS_STACK_SIZE]`
Служебный стек.
- `int s_err`
Стандартные потоки: stdin, stdout, stderr.
- `int s_in`
- `int s_out`
- `sigset_t sa_mask`
Маскированные сигналы.
- `int safe`
Флаг защиты от удаления/приостановки.
- `void * sem`
Указатель на использующийся семафор.
- `sig_handle sh [32]`
Обработчики сигналов.
- `int signal`
Полученный задачей сигнал.
- `void * stack`
Указатель на область памяти, выделенную под стек.
- `int stackSize`
Размер стека.

- unsigned long *startSecond*
Секунда с момента включения, в которую была запущена данная задача (при запуске через 3 месяца после включения может возникнуть путаница).
- *TASK_SEM_EXIT taskSemExit*
Признак снятия задачи с семафора.
- void * *vfparea*
Указатель на данные сопроцессора.
- unsigned long *workTime*
Микросекунды, в течении которых задача была активна (переполняется за 71 минуту!).
- unsigned long *workTimeOverflowCount*
Кол-во переполнений поля workTime.

18.43.1. Поля

18.43.1.1. child

struct *TCB** child

18.43.1.2. childstatus

int childstatus

18.43.1.3. cursp

int cursp

18.43.1.4. delay

int delay

18.43.1.5. exit_code

int exit_code

18.43.1.6. exit_list

*exit_proc** exit_list

18.43.1.7. exitbuf

jmp_buf exitbuf

18.43.1.8. flags

int flags

18.43.1.9. intCounter

unsigned intCounter

18.43.1.10. marker

int marker

18.43.1.11. name

char* name

18.43.1.12. Next

struct *TCB** Next

18.43.1.13. parent

struct *TCB** parent

18.43.1.14. Prev

struct *TCB** Prev

18.43.1.15. priority

int priority

18.43.1.16. ps_sp

int ps_sp

18.43.1.17. ps_stack

```
int ps_stack[TASK_PS_STACK_SIZE]
```

18.43.1.18. s_err

```
int s_err
```

18.43.1.19. s_in

```
int s_in
```

18.43.1.20. s_out

```
int s_out
```

18.43.1.21. sa_mask

```
sigset_t sa_mask
```

18.43.1.22. safe

```
int safe
```

18.43.1.23. sem

```
void* sem
```

18.43.1.24. sh

```
sig_handle sh[32]
```

18.43.1.25. signal

```
int signal
```

18.43.1.26. stack

void* stack

18.43.1.27. stackSize

int stackSize

18.43.1.28. startSecond

unsigned long startSecond

18.43.1.29. taskSemExit

TASK_SEM_EXIT taskSemExit

18.43.1.30. vfparea

void* vfparea

18.43.1.31. workTime

unsigned long workTime

18.43.1.32. workTimeOverflowCount

unsigned long workTimeOverflowCount

Объявления и описания членов структуры находятся в файле:

- *tasklib.h*

18.44. Структура tDrvBit

Описание одного бита регистра.

Поля данных

- unsigned int *addr*
- unsigned int *n*

18.44.1. Подробное описание

Описание одного бита регистра, отвечающего за настройку работы модуля. В зависимости от процессора одни и те же настройки могут быть включены в состав различных регистров.

18.44.2. Поля

18.44.2.1. addr

unsigned int addr

Адрес регистра (абсолютный). 0 – настройка отсутствует.

18.44.2.2. n

unsigned int n

Номер бита в регистре.

Объявления и описания членов структуры находятся в файле:

- *arch.h*

18.45. Структура tDrvBitGroup

Описание группы бит регистра.

Поля данных

- unsigned int *addr*
- unsigned int *mask*
- unsigned int *n*

18.45.1. Подробное описание

Описание группы бит регистра, отвечающих за настройку работы модуля. В зависимости от процессора одни и те же настройки могут быть включены в состав различных регистров.

18.45.2. Поля

18.45.2.1. addr

unsigned int addr

Адрес регистра (абсолютный). 0 – настройка отсутствует.

18.45.2.2. mask

unsigned int mask

Маска (не сдвинутая – содержится в младших битах).

18.45.2.3. n

unsigned int n

Номер младшего бита в регистре.

Объявления и описания членов структуры находятся в файле:

- *arch.h*

18.46. Структура tDrvGpio

Описание аппаратных линий вывода.

Поля данных

- *bool available*
Аппаратное наличие линии.
- *bool enabled*
Использование линии драйвером.
- unsigned int *mux*
Значение мультиплексора.
- unsigned int *pin*
Используемая линия.

18.46.1. Подробное описание

Описание аппаратных линий вывода.

18.46.2. Поля

18.46.2.1. available

bool available

Не все наборы содержат полный комплект линий.

18.46.2.2. enabled

bool enabled

Могут использоваться не все пины из доступного набора.

18.46.2.3. mux

unsigned int mux

Значение мультиплексора для подключения к аппаратному модулю.

18.46.2.4. pin

unsigned int pin

Номер порта + номер бита.

Объявления и описания членов структуры находятся в файле:

- *arch.h*

18.47. Структура `textRect`

Структура прямоугольника.

Поля данных

- `int h`
- `int w`
- `int x`
- `int y`

18.47.1. Поля

18.47.1.1. `h`

`int h`

Высота прямоугольника.

18.47.1.2. `w`

`int w`

Ширина прямоугольника.

18.47.1.3. `x`

`int x`

X-координата.

18.47.1.4. `y`

`int y`

Y-координата.

Объявления и описания членов структуры находятся в файле:

- `fontsdefines.h`

18.48. Структура `timespec`

Тики процессора.

Поля данных

- `long tv_nsec`
Наносекунды, [0, 999999999].
- `time_t tv_sec`
Количество целых секунд.

18.48.1. Подробное описание

Временной интервал.

18.48.2. Поля

18.48.2.1. `tv_nsec`

`long tv_nsec`

18.48.2.2. `tv_sec`

`time_t tv_sec`

Объявления и описания членов структуры находятся в файле:

- `time.h`

18.49. Структура tm

Поля данных

- int *tm_hour*
- int *tm_isdst*
- int *tm_mday*
- int *tm_min*
- int *tm_mon*
- int *tm_sec*
- int *tm_wday*
- int *tm_yday*
- int *tm_year*

18.49.1. Подробное описание

Время в календарном представлении.

18.49.2. Поля

18.49.2.1. tm_hour

int tm_hour

Количество часов, [0, 23].

18.49.2.2. tm_isdst

int tm_isdst

Флаг летнего времени.

18.49.2.3. tm_mday

int tm_mday

День в месяце, [1, 31].

18.49.2.4. tm_min

int tm_min

Количество минут, [0, 59].

18.49.2.5. tm_mon

int tm_mon

Количество месяцев с Января, [0, 11].

18.49.2.6. tm_sec

int tm_sec

Количество секунд, [0, 60].

18.49.2.7. tm_wday

int tm_wday

Количество дней с Воскресенья, [0, 6].

18.49.2.8. tm_yday

int tm_yday

Количество дней с 1го января, [0, 365].

18.49.2.9. tm_year

int tm_year

Количество лет с 1900.

Объявления и описания членов структуры находятся в файле:

- *time.h*

18.50. Структура tMapIterators

Набор указателей для работы со списками.

Поля данных

- void * *end*
Итератор конца списка.
- void * *start*
Итератор начала списка.

18.50.1. Подробное описание

Набор указателей нужен для добавления записей и поиска по списку.

18.50.2. Поля

18.50.2.1. end

void* end

18.50.2.2. start

void* start

Объявления и описания членов структуры находятся в файле:

- *mapstr.h*

18.51. Структура tRingBuffer

Структура кольцевого буфера.

Поля данных

- char * *data*
- int *delta*
- unsigned int *inCnt*
- unsigned int *maxCnt*
- unsigned int *nSize*
- unsigned int *outCnt*

18.51.1. Поля

18.51.1.1. data

char* data

Указатель на данные.

18.51.1.2. delta

int delta

Дельта между входящими и исходящими элементами (количество данных).

18.51.1.3. inCnt

unsigned int inCnt

Счётчик входящих элементов буфера.

18.51.1.4. maxCnt

unsigned int maxCnt

Количество элементов буфера.

18.51.1.5. nSize

unsigned int nSize

Размер одного элемента данных буфера.

18.51.1.6. outCnt

unsigned int outCnt

Счётчик исходящих элементов буфера.

Объявления и описания членов структуры находятся в файле:

- *ringbuffer.h*

18.52. Структура tScreenDeviceMode

Структура настройки подключения дисплея **LCD**.

Поля данных

- int *bpp*
- int *hsync_len*
- int *left_margin*
Horizontal back porch.
- int *lower_margin*
Vertical front porch.
- int *pixclock_khz*
- int *right_margin*
Horizontal front porch.
- int *sync*
- int *upper_margin*
Vertical back porch.
- int *vmode*
- int *vsync_len*
- int *xres*
- int *yres*

18.52.1. Подробное описание

Параметры для конкретного дисплея следует брать из документации на этот дисплей.

18.52.2. Поля

18.52.2.1. bpp

int bpp

Количество бит на пиксель (обычно - 32).

18.52.2.2. hsync_len

int hsync_len

Длительность сигнала строчной синхронизации.

18.52.2.3. left_margin

int left_margin

Отступ слева - интервал между сигналом строчной синхронизации и началом видимого сигнала (Horizontal back porch, задаётся в пикселях).

18.52.2.4. lower_margin

int lower_margin

Отступ снизу - интервал после окончания вывода кадра до сигнала вертикальной синхронизации (Vertical front porch, задаётся в количестве строк).

18.52.2.5. pixclock_khz

int pixclock_khz

Частота pixel clock в КГц (Влияет на fps, некоторые мониторы можно разгонять выше заявленной).

18.52.2.6. right_margin

int right_margin

Отступ справа - интервал после окончания видимого сигнала до следующего сигнала строчной синхронизации (Horizontal front porch, задаётся в пикселях).

18.52.2.7. sync

int sync

Флаги синхронизации: 1 - строчная синхронизация; 2 - вертикальная синхронизация. Обычно оба флага должны быть установлены (значение переменной равно 3).

18.52.2.8. upper_margin

int upper_margin

Отступ сверху - интервал от сигнала вертикальной синхронизации до начала вывода нового кадра (Vertical back porch, задаётся в количестве строк).

18.52.2.9. vmode

int vmode

Флаги видео режимов: 1 - чересстрочная развёртка. Для LCD дисплеев - значение 0.

18.52.2.10. vsync_len

int vsync_len

Длительность сигнала вертикальной синхронизации.

18.52.2.11. xres

int xres

Количество отображаемых точек по вертикали.

18.52.2.12. yres

int yres

Количество отображаемых точек по горизонтали.

Объявления и описания членов структуры находятся в файле:

- [de2.h](#)

18.53. Структура `udp_hdr`

Заголовок пакета **UDP**.

Поля данных

- unsigned short `udp_d_port`
Порт получателя.
- unsigned short `udp_len`
*Длина пакета **UDP**.*
- unsigned short `udp_s_port`
Порт отправителя.
- unsigned short `udp_sum`
Контрольная сумма.

18.53.1. Подробное описание

Заголовок пакета **UDP**. Порядок следования байт – старшим байтом вперёд. Для чтения значений байты следует менять местами.

18.53.2. Поля

18.53.2.1. `udp_d_port`

unsigned short `udp_d_port`

18.53.2.2. `udp_len`

unsigned short `udp_len`

18.53.2.3. `udp_s_port`

unsigned short `udp_s_port`

18.53.2.4. `udp_sum`

unsigned short `udp_sum`

Объявления и описания членов структуры находятся в файле:

- `udp.h`

18.54. Структура `udp_service`

Поля данных

- struct `udp_service` * *next*
- unsigned short *port*
- `udpServRout` *service*

18.54.1. Поля

18.54.1.1. `next`

struct `udp_service`* *next*

18.54.1.2. `port`

unsigned short *port*

18.54.1.3. `service`

`udpServRout` *service*

Объявления и описания членов структуры находятся в файле:

- `udp.h`

18.55. Структура `usb_class_abstract_control_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.55.1. Поля

18.55.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.55.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.55.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.55.1.4. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.56. Структура `usb_class_atm_networking_descriptor`

Поля данных

- u8 *bDescriptorSubtype*
- u8 *bDescriptorType*
- u8 *bFunctionLength*
- u8 *bmATMDeviceStatistics*
- u8 *bmDataCapabilities*
- u8 *iEndSystemIdentifier*
- u16 *wMaxVC*
- u16 *wType2MaxSegmentSize*
- u16 *wType3MaxSegmentSize*

18.56.1. Поля

18.56.1.1. `bDescriptorSubtype`

u8 `bDescriptorSubtype`

18.56.1.2. `bDescriptorType`

u8 `bDescriptorType`

18.56.1.3. `bFunctionLength`

u8 `bFunctionLength`

18.56.1.4. `bmATMDeviceStatistics`

u8 `bmATMDeviceStatistics`

18.56.1.5. `bmDataCapabilities`

u8 `bmDataCapabilities`

18.56.1.6. `iEndSystemIdentifier`

u8 `iEndSystemIdentifier`

18.56.1.7. `wMaxVC`

u16 `wMaxVC`

18.56.1.8. wType2MaxSegmentSize

u16 wType2MaxSegmentSize

18.56.1.9. wType3MaxSegmentSize

u16 wType3MaxSegmentSize

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.57. Структура `usb_class_call_management_descriptor`

Поля данных

- `u8 bDataInterface`
- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.57.1. Поля

18.57.1.1. `bDataInterface`

`u8 bDataInterface`

18.57.1.2. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.57.1.3. `bDescriptorType`

`u8 bDescriptorType`

18.57.1.4. `bFunctionLength`

`u8 bFunctionLength`

18.57.1.5. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.58. Структура `usb_class_capi_control_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.58.1. Поля

18.58.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.58.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.58.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.58.1.4. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.59. Структура `usb_class_country_selection_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 iCountryCodeRelDate`
- `u16 wCountryCode0 [0]`

18.59.1. Поля

18.59.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.59.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.59.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.59.1.4. `iCountryCodeRelDate`

`u8 iCountryCodeRelDate`

18.59.1.5. `wCountryCode0`

`u16 wCountryCode0[0]`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.60. Структура `usb_class_descriptor`

Поля данных

- union {
 - struct `usb_class_abstract_control_descriptor` `abstract_control`
 - struct `usb_class_atm_networking_descriptor` `atm_networking`
 - struct `usb_class_call_management_descriptor` `call_management`
 - struct `usb_class_capi_control_descriptor` `capi_control`
 - struct `usb_class_country_selection_descriptor` `country_selection`
 - struct `usb_class_direct_line_descriptor` `direct_line`
 - struct `usb_class_ethernet_networking_descriptor` `ethernet_networking`
 - struct `usb_class_extension_unit_descriptor` `extension_unit`
 - struct `usb_class_function_descriptor` `function`
 - struct `usb_class_function_descriptor_generic` `generic`
 - struct `usb_class_header_function_descriptor` `header_function`
 - struct `usb_class_hid_descriptor` `hid`
 - struct `usb_class_mdln_descriptor` `mobile_direct`
 - struct `usb_class_mdln_detail_descriptor` `mobile_direct_detail`
 - struct `usb_class_multi_channel_descriptor` `multi_channel`
 - struct `usb_class_network_channel_descriptor` `network_channel`
 - struct `usb_class_telephone_call_descriptor` `telephone_call`
 - struct `usb_class_telephone_operational_descriptor` `telephone_operational`
 - struct `usb_class_telephone_ringer_descriptor` `telephone_ringer`
 - struct `usb_class_union_function_descriptor` `union_function`
 - struct `usb_class_usb_terminal_descriptor` `usb_terminal`

18.60.1. Поля

18.60.1.1. `abstract_control`

```
struct usb_class_abstract_control_descriptor abstract_control
```

18.60.1.2. `atm_networking`

```
struct usb_class_atm_networking_descriptor atm_networking
```

18.60.1.3. `call_management`

```
struct usb_class_call_management_descriptor call_management
```

18.60.1.4. `capi_control`

```
struct usb_class_capi_control_descriptor capi_control
```


18.60.1.5. country_selection

struct *usb_class_country_selection_descriptor* country_selection

18.60.1.6.

union { ... } descriptor

18.60.1.7. direct_line

struct *usb_class_direct_line_descriptor* direct_line

18.60.1.8. ethernet_networking

struct *usb_class_ethernet_networking_descriptor* ethernet_networking

18.60.1.9. extension_unit

struct *usb_class_extension_unit_descriptor* extension_unit

18.60.1.10. function

struct *usb_class_function_descriptor* function

18.60.1.11. generic

struct *usb_class_function_descriptor_generic* generic

18.60.1.12. header_function

struct *usb_class_header_function_descriptor* header_function

18.60.1.13. hid

struct *usb_class_hid_descriptor* hid

18.60.1.14. mobile_direct

struct *usb_class_mdlm_descriptor* mobile_direct

18.60.1.15. mobile_direct_detail

struct *usb_class_mdlmd_descriptor* mobile_direct_detail

18.60.1.16. multi_channel

struct *usb_class_multi_channel_descriptor* multi_channel

18.60.1.17. network_channel

struct *usb_class_network_channel_descriptor* network_channel

18.60.1.18. telephone_call

struct *usb_class_telephone_call_descriptor* telephone_call

18.60.1.19. telephone_operational

struct *usb_class_telephone_operational_descriptor* telephone_operational

18.60.1.20. telephone_ringer

struct *usb_class_telephone_ringer_descriptor* telephone_ringer

18.60.1.21. union_function

struct *usb_class_union_function_descriptor* union_function

18.60.1.22. usb_terminal

struct *usb_class_usb_terminal_descriptor* usb_terminal

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.61. Структура `usb_class_direct_line_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`

18.61.1. Поля

18.61.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.61.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.61.1.3. `bFunctionLength`

`u8 bFunctionLength`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.62. Структура `usb_class_ethernet_networking_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u32 bmEthernetStatistics`
- `u8 bNumberPowerFilters`
- `u8 iMACAddress`
- `u16 wMaxSegmentSize`
- `u16 wNumberMCFilters`

18.62.1. Поля

18.62.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.62.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.62.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.62.1.4. `bmEthernetStatistics`

`u32 bmEthernetStatistics`

18.62.1.5. `bNumberPowerFilters`

`u8 bNumberPowerFilters`

18.62.1.6. `iMACAddress`

`u8 iMACAddress`

18.62.1.7. `wMaxSegmentSize`

`u16 wMaxSegmentSize`

18.62.1.8. wNumberMCFilters

u16 wNumberMCFilters

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.63. Структура `usb_class_extension_unit_descriptor`

Поля данных

- `u8 bChild0 [0]`
- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bEntityId`
- `u8 bExtensionCode`
- `u8 bFunctionLength`
- `u8 iName`

18.63.1. Поля

18.63.1.1. `bChild0`

`u8 bChild0[0]`

18.63.1.2. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.63.1.3. `bDescriptorType`

`u8 bDescriptorType`

18.63.1.4. `bEntityId`

`u8 bEntityId`

18.63.1.5. `bExtensionCode`

`u8 bExtensionCode`

18.63.1.6. `bFunctionLength`

`u8 bFunctionLength`

18.63.1.7. `iName`

`u8 iName`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.64. Структура `usb_class_function_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`

18.64.1. Поля

18.64.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.64.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.64.1.3. `bFunctionLength`

`u8 bFunctionLength`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.65. Структура `usb_class_function_descriptor_generic`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.65.1. Поля

18.65.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.65.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.65.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.65.1.4. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.66. Структура `usb_class_header_function_descriptor`

Поля данных

- u16 *bcdCDC*
- u8 *bDescriptorSubtype*
- u8 *bDescriptorType*
- u8 *bFunctionLength*

18.66.1. Поля

18.66.1.1. `bcdCDC`

u16 `bcdCDC`

18.66.1.2. `bDescriptorSubtype`

u8 `bDescriptorSubtype`

18.66.1.3. `bDescriptorType`

u8 `bDescriptorType`

18.66.1.4. `bFunctionLength`

u8 `bFunctionLength`

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.67. Структура `usb_class_hid_descriptor`

Поля данных

- u16 *bcdCDC*
- u8 *bCountryCode*
- u8 *bDescriptorType*
- u8 *bDescriptorType0*
- u8 *bLength*
- u8 *bNumDescriptors*
- u16 *wDescriptorLength0*

18.67.1. Поля

18.67.1.1. `bcdCDC`

u16 `bcdCDC`

18.67.1.2. `bCountryCode`

u8 `bCountryCode`

18.67.1.3. `bDescriptorType`

u8 `bDescriptorType`

18.67.1.4. `bDescriptorType0`

u8 `bDescriptorType0`

18.67.1.5. `bLength`

u8 `bLength`

18.67.1.6. `bNumDescriptors`

u8 `bNumDescriptors`

18.67.1.7. `wDescriptorLength0`

u16 `wDescriptorLength0`

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.68. Структура `usb_class_mdIm_descriptor`

Поля данных

- u16 *bcdVersion*
- u8 *bDescriptorSubtype*
- u8 *bDescriptorType*
- u8 *bFunctionLength*
- u8 *bGUID* [16]

18.68.1. Поля

18.68.1.1. `bcdVersion`

u16 `bcdVersion`

18.68.1.2. `bDescriptorSubtype`

u8 `bDescriptorSubtype`

18.68.1.3. `bDescriptorType`

u8 `bDescriptorType`

18.68.1.4. `bFunctionLength`

u8 `bFunctionLength`

18.68.1.5. `bGUID`

u8 `bGUID`[16]

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.69. Структура `usb_class_md_lmd_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bDetailData [0]`
- `u8 bFunctionLength`
- `u8 bGuidDescriptorType`

18.69.1. Поля

18.69.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.69.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.69.1.3. `bDetailData`

`u8 bDetailData[0]`

18.69.1.4. `bFunctionLength`

`u8 bFunctionLength`

18.69.1.5. `bGuidDescriptorType`

`u8 bGuidDescriptorType`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.70. Структура `usb_class_multi_channel_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.70.1. Поля

18.70.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.70.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.70.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.70.1.4. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.71. Структура `usb_class_network_channel_descriptor`

Поля данных

- `u8 bChannelIndex`
- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bEntityId`
- `u8 bFunctionLength`
- `u8 bPhysicalInterface`
- `u8 iName`

18.71.1. Поля

18.71.1.1. `bChannelIndex`

`u8 bChannelIndex`

18.71.1.2. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.71.1.3. `bDescriptorType`

`u8 bDescriptorType`

18.71.1.4. `bEntityId`

`u8 bEntityId`

18.71.1.5. `bFunctionLength`

`u8 bFunctionLength`

18.71.1.6. `bPhysicalInterface`

`u8 bPhysicalInterface`

18.71.1.7. `iName`

`u8 iName`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.72. Структура `usb_class_protocol_unit_function_descriptor`

Поля данных

- `u8 bChild0 [0]`
- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bEntityId`
- `u8 bFunctionLength`
- `u8 bProtocol`

18.72.1. Поля

18.72.1.1. `bChild0`

`u8 bChild0[0]`

18.72.1.2. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.72.1.3. `bDescriptorType`

`u8 bDescriptorType`

18.72.1.4. `bEntityId`

`u8 bEntityId`

18.72.1.5. `bFunctionLength`

`u8 bFunctionLength`

18.72.1.6. `bProtocol`

`u8 bProtocol`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.73. Структура `usb_class_report_descriptor`

Поля данных

- `u8 bData [0]`
- `u8 bDescriptorType`
- `u8 bLength`
- `u16 wLength`

18.73.1. Поля

18.73.1.1. `bData`

`u8 bData[0]`

18.73.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.73.1.3. `bLength`

`u8 bLength`

18.73.1.4. `wLength`

`u16 wLength`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.74. Структура `usb_class_telephone_call_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.74.1. Поля

18.74.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.74.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.74.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.74.1.4. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.75. Структура `usb_class_telephone_operational_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bmCapabilities`

18.75.1. Поля

18.75.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.75.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.75.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.75.1.4. `bmCapabilities`

`u8 bmCapabilities`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.76. Структура `usb_class_telephone_ringer_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bNumRingerPatterns`
- `u8 bRingerVolSeps`

18.76.1. Поля

18.76.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.76.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.76.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.76.1.4. `bNumRingerPatterns`

`u8 bNumRingerPatterns`

18.76.1.5. `bRingerVolSeps`

`u8 bRingerVolSeps`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.77. Структура `usb_class_union_function_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bFunctionLength`
- `u8 bMasterInterface`
- `u8 bSlaveInterface0`

18.77.1. Поля

18.77.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.77.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.77.1.3. `bFunctionLength`

`u8 bFunctionLength`

18.77.1.4. `bMasterInterface`

`u8 bMasterInterface`

18.77.1.5. `bSlaveInterface0`

`u8 bSlaveInterface0`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.78. Структура `usb_class_usb_terminal_descriptor`

Поля данных

- `u8 bChild0 [0]`
- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bEntityId`
- `u8 bFunctionLength`
- `u8 bInterfaceNo`
- `u8 bmOptions`
- `u8 bOutInterfaceNo`

18.78.1. Поля

18.78.1.1. `bChild0`

`u8 bChild0[0]`

18.78.1.2. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.78.1.3. `bDescriptorType`

`u8 bDescriptorType`

18.78.1.4. `bEntityId`

`u8 bEntityId`

18.78.1.5. `bFunctionLength`

`u8 bFunctionLength`

18.78.1.6. `bInterfaceNo`

`u8 bInterfaceNo`

18.78.1.7. `bmOptions`

`u8 bmOptions`

18.78.1.8. bOutInterfaceNo

u8 bOutInterfaceNo

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.79. Структура `usb_config`

Структура дескриптора конфигурации.

Поля данных

- struct `usb_configuration_descriptor desc`
- struct `usb_interface if_desc [USB_MAXINTERFACES]`
- unsigned char `no_of_if`

18.79.1. Подробное описание

Это вспомогательная структура, которая содержит дескриптор конфигурации, количество интерфейсов для устройства и массив структур `usb_interface` для всех интерфейсов устройства.

18.79.2. Поля

18.79.2.1. `desc`

```
struct usb_configuration_descriptor desc
```

Дескриптор конфигурации.

18.79.2.2. `if_desc`

```
struct usb_interface if_desc[USB_MAXINTERFACES]
```

Массив структур `usb_interface` для всех интерфейсов устройства.

18.79.2.3. `no_of_if`

```
unsigned char no_of_if
```

Число интерфейсов.

Объявления и описания членов структуры находятся в файле:

- `usb.h`

18.80. Структура `usb_configuration_descriptor`

Структура дескриптора конфигурации.

Поля данных

- `u8 bConfigurationValue`
- `u8 bDescriptorType`
- `u8 bLength`
- `u8 bmAttributes`
- `u8 bMaxPower`
- `u8 bNumInterfaces`
- `u8 iConfiguration`
- `u16 wTotalLength`

18.80.1. Подробное описание

Дескриптор конфигурации указывает величину потребляемой мощности от шины, питается ли устройство от собственного источника (*self powered*) либо от шины USB (*bus powered*) и количество интерфейсов, которые есть у конфигурации. Когда устройство проходит эnumерацию, хост читает дескриптор устройства и принимает решение, какую конфигурацию применить. Хост может разрешить только какую-то одну из конфигураций.

18.80.2. Поля

18.80.2.1. `bConfigurationValue`

`u8 bConfigurationValue`

Значение, используемое для выбора конфигурации. Это значение передается в запрос `USB setConfiguration`, как описано в версии 1.1 спецификации универсальной последовательной шины.

18.80.2.2. `bDescriptorType`

`u8 bDescriptorType`

Тип дескриптора. Всегда содержит `USB_DT_CONFIG = 0x02`.

18.80.2.3. `bLength`

`u8 bLength`

Длина дескриптора в байтах.

18.80.2.4. `bmAttributes`

`u8 bmAttributes`

Точечный рисунок для описания поведения этой конфигурации. Биты описаны и задаются в порядке с прямым порядком байтов.

Бит	Значение
0-4	Зарезервировано.
5	Конфигурация поддерживает удаленный пробуждение.
6	Конфигурация включается самостоятельно и не использует питание от шины.
7	Конфигурация работает на базе шины.

18.80.2.5. bMaxPower

u8 bMaxPower

Требования к питанию этого устройства в единицах кратных 2 мА. Этот элемент допустим, только если в *bmAttributes* установлен бит 7.

18.80.2.6. bNumInterfaces

u8 bNumInterfaces

Общее число интерфейсов, поддерживаемых данной конфигурацией.

18.80.2.7. iConfiguration

u8 iConfiguration

Определяемый устройством индекс дескриптора строки для этой конфигурации.

18.80.2.8. wTotalLength

u16 wTotalLength

Общая длина (в байтах) всех данных для конфигурации. Длина включает все дескрипторы интерфейса, конечной точки, класса или конкретного поставщика, возвращаемые с дескриптором конфигурации.

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.81. Структура `usb_descriptor`

Поля данных

- union {
 struct *usb_configuration_descriptor* configuration
 struct *usb_device_descriptor* device
 struct *usb_endpoint_descriptor* endpoint
 struct *usb_generic_descriptor* generic
 struct *usb_interface_descriptor* interface
 struct *usb_string_descriptor* string
} *descriptor*

18.81.1. Поля

18.81.1.1. configuration

struct *usb_configuration_descriptor* configuration

18.81.1.2.

union { ... } descriptor

18.81.1.3. device

struct *usb_device_descriptor* device

18.81.1.4. endpoint

struct *usb_endpoint_descriptor* endpoint

18.81.1.5. generic

struct *usb_generic_descriptor* generic

18.81.1.6. interface

struct *usb_interface_descriptor* interface

18.81.1.7. string

```
struct usb_string_descriptor string
```

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.82. Структура `usb_device`

Структура данных о подключении USB-устройства.

Поля данных

- int *act_len*
- struct *usb_device* * *children* [*USB_MAXCHILDREN*]
- struct *usb_config* *config*
- int *configno*
- struct *usb_device_descriptor* *descriptor*
- char * *devname*
- int *devnum*
- struct *usb_driver* * *driver*
- int *epmaxpacketin* [16]
- int *epmaxpacketout* [16]
- unsigned int *halted* [2]
- int *have_langid*
- void * *hcd*
- int *irq_act_len*
- int(* *irq_handle*)(struct *usb_device* *dev)
- void * *irq_q*
- unsigned long *irq_status*
- int *maxchild*
- int *maxpacketsize*
- char *mf* [32]
- struct *usb_device* * *parent*
- int *portnr*
- void * *privptr*
- char *prod* [32]
- char *serial* [32]
- int *speed*
- unsigned long *status*
- int *string_langid*
- unsigned int *toggle* [2]

18.82.1. Поля

18.82.1.1. `act_len`

```
int act_len
```

Переданные байты.

18.82.1.2. `children`

```
struct usb_device* children[USB_MAXCHILDREN]
```

18.82.1.3. `config`

```
struct usb_config config
```

Конфигурационный дескриптор.

18.82.1.4. configno

int configno

18.82.1.5. descriptor

struct *usb_device_descriptor* descriptor

Дескриптор устройства.

18.82.1.6. devname

char* devname

Указатель на строку имени устройства.

18.82.1.7. devnum

int devnum

Номер устройства на шине USB.

18.82.1.8. driver

struct *usb_driver** driver

18.82.1.9. epmaxpacketin

int epmaxpacketin[16]

INput endpoint specific maximums.

18.82.1.10. epmaxpacketout

int epmaxpacketout[16]

OUTput endpoint specific maximums.

18.82.1.11. halted

unsigned int halted[2]

Признак останова конечной точки.

18.82.1.12. have_langid

int have_langid

Признак является ли /b string_langid еще действительным.

18.82.1.13. hcd

void* hcd

18.82.1.14. irq_act_len

int irq_act_len

Переданные байты.

18.82.1.15. irq_handle

int(* irq_handle) (struct *usb_device* *dev)

18.82.1.16. irq_q

void* irq_q

18.82.1.17. irq_status

unsigned long irq_status

18.82.1.18. maxchild

int maxchild

Количество портов, если это концентратор.

18.82.1.19. maxpacketsize

int maxpacketsize

Максимальный размер пакета. Одно из значений: *PACKET_SIZE_8*, *PACKET_SIZE_16*, *PACKET_SIZE_32*, *PACKET_SIZE_64*.

18.82.1.20. mf

char mf[32]

Производитель устройства.

18.82.1.21. parent

struct *usb_device** parent

18.82.1.22. portnr

int portnr

18.82.1.23. privptr

void* privptr

18.82.1.24. prod

char prod[32]

Заводское наименование.

18.82.1.25. serial

char serial[32]

Серийный номер.

18.82.1.26. speed

int speed

Скорость подключения (full/low/high).

18.82.1.27. status

unsigned long status

18.82.1.28. string_langid

int string_langid

Идентификатор языка для строк.

18.82.1.29. toggle

unsigned int toggle[2]

По одному биту на каждую конечную точку ([0] = IN, [1] = OUT).

Объявления и описания членов структуры находятся в файле:

- *usb.h*

18.83. Структура `usb_device_descriptor`

Структура дескриптора устройства **USB**.

Поля данных

- u16 *bcdDevice*
- u16 *bcdUSB*
- u8 *bDescriptorType*
- u8 *bDeviceClass*
- u8 *bDeviceProtocol*
- u8 *bDeviceSubClass*
- u8 *bLength*
- u8 *bMaxPacketSize0*
- u8 *bNumConfigurations*
- u16 *idProduct*
- u16 *idVendor*
- u8 *iManufacturer*
- u8 *iProduct*
- u8 *iSerialNumber*

18.83.1. Подробное описание

Устройства USB могут иметь только один дескриптор. Дескриптор устройства включает в себя такую информацию, как поддерживаемую устройством ревизию USB, Product ID (**PID**, идентификатор продукта), Vendor ID (**VID**, идентификатор производителя), используемые для загрузки соответствующего устройству драйвера, и количество возможных конфигураций устройства. Число конфигураций указывает, сколько имеется ответвлений по дескрипторам конфигурации.

18.83.2. Поля

18.83.2.1. `bcdDevice`

u16 `bcdDevice`

Версия устройства. Это значение представляет собой двоичное десятичное число.

18.83.2.2. `bcdUSB`

u16 `bcdUSB`

Версия спецификации USB, которую эта структура дескриптора соответствует. Это значение представляет собой двоичное десятичное число.

18.83.2.3. `bDescriptorType`

u8 `bDescriptorType`

Тип дескриптора. Содержит значение **USB_DT_DEVICE** = **0x01**.

18.83.2.4. `bDeviceClass`

u8 `bDeviceClass`

Код класса устройства, назначенный группой спецификаций USB.

18.83.2.5. bDeviceProtocol

u8 bDeviceProtocol

Код протокола устройства, назначенный группой спецификаций USB.

18.83.2.6. bDeviceSubClass

u8 bDeviceSubClass

Код подкласса устройства, назначенный группой спецификаций USB.

18.83.2.7. bLength

u8 bLength

Длина дескриптора в байтах.

18.83.2.8. bMaxPacketSize0

u8 bMaxPacketSize0

Максимальный размер пакета (в байтах) для конечной точки устройства. Значение должно быть равно 8, 16, 32 или 64.

18.83.2.9. bNumConfigurations

u8 bNumConfigurations

Общее число возможных конфигураций для устройства.

18.83.2.10. idProduct

u16 idProduct

Идентификатор продукта. Это значение назначается изготовителем и зависит от конкретного устройства.

18.83.2.11. idVendor

u16 idVendor

Идентификатор поставщика для устройства, назначенный Комитетом спецификации USB.

18.83.2.12. iManufacturer

u8 iManufacturer

Определяемый устройством индекс строкового дескриптора, который предоставляет строку, содержащую имя производителя этого устройства.

18.83.2.13. iProduct

u8 iProduct

Определяемый устройством индекс строкового дескриптора, который предоставляет строку, содержащую описание устройства.

18.83.2.14. iSerialNumber

u8 iSerialNumber

Определяемый устройством индекс строкового дескриптора, который предоставляет строку, которая содержит серийный номер устройства, определяемый производителем.

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.84. Структура `usb_endpoint_descriptor`

Структура дескриптора конечной точки.

Поля данных

- `u8 bDescriptorType`
- `u8 bEndpointAddress`
- `u8 bInterval`
- `u8 bLength`
- `u8 bmAttributes`
- `u16 wMaxPacketSize`

18.84.1. Подробное описание

Дескрипторы конечной точки используются для описания конечных точек, отличных от конечной точки **0**. Конечная точка **0** всегда используется как конечная точка управления, и она конфигурируется сразу автоматически, даже перед запросом информации всех дескрипторов. Хост использует информацию, полученную из описателей конечных точек, чтобы определить требования по полосе пропускания шины.

18.84.2. Поля

18.84.2.1. `bDescriptorType`

`u8 bDescriptorType`

Тип дескриптора. Всегда **USB_DT_ENDPOINT = 0x05**.

18.84.2.2. `bEndpointAddress`

`u8 bEndpointAddress`

Адрес конечной точки, определяемый USB. Четыре младшие бита указывают номер конечной точки. Старший бит задаёт направление потока данных в этой конечной точке: **1** для **in**, **0** — для **out**.

18.84.2.3. `bInterval`

`u8 bInterval`

Интервал для того, чтобы опросить передачи данных конечной точки. Указывается в количестве фреймов. Поле игнорируется для конечных точек **Bulk** и **Control**. Для конечных точек **Isochronous** должно быть равно **1** и для конечных точек **interrupt** может лежать в диапазоне **1..255**.

18.84.2.4. `bLength`

`u8 bLength`

Длина дескриптора в байтах.

18.84.2.5. `bmAttributes`

`u8 bmAttributes`

Битовая маска атрибутов:

Бит	Значение
0-1	Тип передачи: <ul style="list-style-type: none">• 00 — Control• 01 — Isochronous• 10 — Bulk• 11 — Interrupt
2-3	Только для изохронной конечной точки (режим синхронизации Iso), иначе зарезервировано. Тип синхронизации: <ul style="list-style-type: none">• 00 — No Synchronisation.• 01 — Asynchronous.• 10 — Adaptive.• 11 — Synchronous.
4-5	Только для изохронной конечной точки (режим синхронизации Iso), иначе зарезервировано. Тип использования (Usage Type): <ul style="list-style-type: none">• 00 — Конечная точка данных.• 01 — Конечная точка обратной связи (Feedback Endpoint).• 10 — Явная конечная точка обратной связи данных (Explicit Feedback Data Endpoint).• 11 — Зарезервировано.

18.84.2.6. wMaxPacketSize

u16 wMaxPacketSize

Максимальный размер пакета, который может быть отправлен из этой конечной точки или в эту конечную точку.

Объявления и описания членов структуры находятся в файле:

- *usbdescriptors.h*

18.85. Структура `usb_generic_descriptor`

Поля данных

- `u8 bDescriptorSubtype`
- `u8 bDescriptorType`
- `u8 bLength`

18.85.1. Поля

18.85.1.1. `bDescriptorSubtype`

`u8 bDescriptorSubtype`

18.85.1.2. `bDescriptorType`

`u8 bDescriptorType`

18.85.1.3. `bLength`

`u8 bLength`

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

18.86. Структура `usb_interface`

Структура дескриптора интерфейса.

Поля данных

- unsigned char `act_altsetting`
- struct `usb_interface_descriptor desc`
- struct `usb_endpoint_descriptor ep_desc [USB_MAXENDPOINTS]`
- unsigned char `no_of_ep`
- unsigned char `num_altsetting`

18.86.1. Подробное описание

Это вспомогательная структура, которая содержит дескриптор интерфейса, число конечных точек для данного интерфейса, число альтернативных установок и дескрипторы для всех конечных точек интерфейса.

18.86.2. Поля

18.86.2.1. `act_altsetting`

unsigned char `act_altsetting`

Актуальная установка.

18.86.2.2. `desc`

struct `usb_interface_descriptor desc`

Дескриптор интерфейса.

18.86.2.3. `ep_desc`

struct `usb_endpoint_descriptor ep_desc[USB_MAXENDPOINTS]`

Дескрипторы конечных точек.

18.86.2.4. `no_of_ep`

unsigned char `no_of_ep`

Количество конечных точек.

18.86.2.5. `num_altsetting`

unsigned char `num_altsetting`

Число альт. установок.

Объявления и описания членов структуры находятся в файле:

- `usb.h`

18.87. Структура `usb_interface_descriptor`

Структура дескриптора интерфейса.

Поля данных

- `u8 bAlternateSetting`
- `u8 bDescriptorType`
- `u8 bInterfaceClass`
- `u8 bInterfaceNumber`
- `u8 bInterfaceProtocol`
- `u8 bInterfaceSubClass`
- `u8 bLength`
- `u8 bNumEndpoints`
- `u8 iInterface`

18.87.1. Подробное описание

Дескриптор интерфейса можно рассматривать как заголовок или группирование конечных точек в функциональную группу, выполняющую единственную особенность (*feature*) устройства. Например, для многофункционального устройства факса/сканера/принтера дескриптор интерфейса **1** может описывать конечные точки функции факса, дескриптор интерфейса **2** может описывать функцию сканера, и дескриптор интерфейса **3** может описывать функцию принтера. В отличие от дескриптора конфигурации, здесь нет ограничений на количество одновременно разрешенных интерфейсов. У устройства могут быть один и более интерфейсов, разрешенных одновременно.

18.87.2. Поля

18.87.2.1. `bAlternateSetting`

`u8 bAlternateSetting`

Номер индекса альтернативного параметра интерфейса.

18.87.2.2. `bDescriptorType`

`u8 bDescriptorType`

Тип дескриптора. Всегда `USB_DT_INTERFACE = 0x04`.

18.87.2.3. `bInterfaceClass`

`u8 bInterfaceClass`

Код класса устройства, которому назначена группа спецификаций USB.

18.87.2.4. `bInterfaceNumber`

`u8 bInterfaceNumber`

Номер индекса интерфейса.

18.87.2.5. **bInterfaceProtocol**

u8 bInterfaceProtocol

Код протокола устройства, которому назначена группа спецификаций USB.

18.87.2.6. **bInterfaceSubClass**

u8 bInterfaceSubClass

Код подкласса устройства, назначенный группе спецификаций USB.

18.87.2.7. **bLength**

u8 bLength

Длина дескриптора в байтах.

18.87.2.8. **bNumEndpoints**

u8 bNumEndpoints

Количество конечных точек, используемых интерфейсом, исключая конечную точку состояния по умолчанию.

18.87.2.9. **iInterface**

u8 iInterface

Индекс строкового дескриптора, который описывает интерфейс.

Объявления и описания членов структуры находятся в файле:

- [*usbdescriptors.h*](#)

18.88. Структура `usb_string_descriptor`

Структура дескриптора строки.

Поля данных

- `u8 bDescriptorType`
- `u8 bLength`
- `u16 wData [0]`

18.88.1. Подробное описание

Используется драйверами клиента USB для хранения дескриптора строки, определяемого USB. Члены этой структуры описаны в спецификации универсальной последовательной шины 3.1.

18.88.2. Поля

18.88.2.1. `bDescriptorType`

`u8 bDescriptorType`

Тип дескриптора. Всегда должен быть `USB_DT_STRING = 0x03`.

18.88.2.2. `bLength`

`u8 bLength`

Длина дескриптора в байтах.

18.88.2.3. `wData`

`u16 wData[0]`

Указатель на выделенный клиентом буфер, который содержит строку Юникода с запрошенным дескриптором строки.

Объявления и описания членов структуры находятся в файле:

- `usbdescriptors.h`

19. Файлы

19.1. Файл a20graph.h

Работа с графической подсистемой.

Структуры данных

- struct *Display*
Структура инициализации графического адаптера.
- struct *g2d_blt*
- struct *g2d_fillrect*
- struct *g2d_image*
- struct *g2d_rect*
- struct *g2d_stretchblt*

Определения типов

- typedef *surface_t* * *HDC*
- typedef *g2d_image* *surface_t*

Перечисления

- enum *g2d_blt_flags* {
G2D_BLT_NONE = 0x00000000, *G2D_BLT_PIXEL_ALPHA* = 0x00000001, *G2D_BLT_PLANE_ALPHA* = 0x00000002, *G2D_BLT_MULTI_ALPHA* = 0x00000004, *G2D_BLT_SRC_COLORKEY* = 0x00000008, *G2D_BLT_DST_COLORKEY* = 0x00000010, *G2D_BLT_FLIP_HORIZONTAL* = 0x00000020, *G2D_BLT_FLIP_VERTICAL* = 0x00000040, *G2D_BLT_ROTATE90* = 0x00000080, *G2D_BLT_ROTATE180* = 0x00000100, *G2D_BLT_ROTATE270* = 0x00000200, *G2D_BLT_MIRROR45* = 0x00000400, *G2D_BLT_MIRROR135* = 0x00000800 }
- enum *g2d_data_fmt* {
G2D_FMT_ARGB_AYUV8888 = (0x0), *G2D_FMT_BGRA_VUYA8888* = (0x1), *G2D_FMT_ABGR_AVUY8888* = (0x2), *G2D_FMT_RGBA_YUYA8888* = (0x3), *G2D_FMT_ARGB8888* = (0x0), *G2D_FMT_BGRA8888* = (0x1), *G2D_FMT_ABGR8888* = (0x2), *G2D_FMT_RGBA8888* = (0x3), *G2D_FMT_XRGB8888* = (0x4), *G2D_FMT_BGRX8888* = (0x5), *G2D_FMT_XBGR8888* = (0x6), *G2D_FMT_RGBX8888* = (0x7), *G2D_FMT_ARGB4444* = (0x8), *G2D_FMT_ABGR4444* = (0x9), *G2D_FMT_RGBA4444* = (0xA), *G2D_FMT_BGRA4444* = (0xB), *G2D_FMT_ARGB1555* = (0xC), *G2D_FMT_ABGR1555* = (0xD), *G2D_FMT_RGBA5551* = (0xE), *G2D_FMT_BGRA5551* = (0xF), *G2D_FMT_RGB565* = (0x10), *G2D_FMT_BGR565* = (0x11), *G2D_FMT_IYUV422* = (0x12), *G2D_FMT_8BPP_MONO* = (0x13), *G2D_FMT_4BPP_MONO* = (0x14), *G2D_FMT_2BPP_MONO* = (0x15), *G2D_FMT_1BPP_MONO* = (0x16), *G2D_FMT_PYUV422UVC* = (0x17), *G2D_FMT_PYUV420UVC* = (0x18), *G2D_FMT_PYUV411UVC* = (0x19), *G2D_FMT_PYUV422* = (0x1A), *G2D_FMT_PYUV420* = (0x1B), *G2D_FMT_PYUV411* = (0x1C), *G2D_FMT_8BPP_PALETTE* = (0x1D), *G2D_FMT_4BPP_PALETTE* = (0x1E), *G2D_FMT_2BPP_PALETTE* = (0x1F), *G2D_FMT_1BPP_PALETTE* = (0x20) }
- enum *g2d_fillrect_flags* { *G2D_FIL_NONE* = 0x00000000, *G2D_FIL_PIXEL_ALPHA* = 0x00000001, *G2D_FIL_PLANE_ALPHA* = 0x00000002, *G2D_FIL_MULTI_ALPHA* = 0x00000004 }
- enum *g2d_pixel_seq* {
G2D_SEQ_NORMAL = 0x0, *G2D_SEQ_VYUY* = 0x1, *G2D_SEQ_YVYU* = 0x2, *G2D_SEQ_VUVU* = 0x3, *G2D_SEQ_P10* = 0x4, *G2D_SEQ_P01* = 0x5, *G2D_SEQ_P3210* = 0x6, *G2D_SEQ_P0123* = 0x7, *G2D_SEQ_P76543210* = 0x8, *G2D_SEQ_P67452301* = 0x9, *G2D_SEQ_P10325476* = 0xA, *G2D_SEQ_P01234567* = 0xB,

```
G2D_SEQ_2BPP_BIG_BIG = 0xC , G2D_SEQ_2BPP_BIG_LITTER = 0xD , G2D_SEQ_2BPP_LITTER_BIG = 0xE ,
G2D_SEQ_2BPP_LITTER_LITTER = 0xF ,
G2D_SEQ_1BPP_BIG_BIG = 0x10 , G2D_SEQ_1BPP_BIG_LITTER = 0x11 , G2D_SEQ_1BPP_LITTER_BIG = 0x12 ,
G2D_SEQ_1BPP_LITTER_LITTER = 0x13 }
• enum lvds_param_t {
LVDS_1920x1080 = 0 , LVDS_1920x360 , LVDS_1024x768 , LVDS_800x600 ,
LVDS_800x480 , LVDS_1280x800 , LVDS_158x1920 , LVDS_1920x165 ,
LVDS_1400x1050 }
• enum VideoModes {
MODE_640x480 = 0 , MODE_800x480 , MODE_800x600 , MODE_1024x768 ,
MODE_1280x720 , MODE_1280x768 , MODE_1280x800 , MODE_1368x768 ,
MODE_1280x1024 , MODE_1920x1080 , MODE_TV }
```

Переменные

- struct *Display* *Display*
Глобальная переменная описатель экрана.

Указатели на поверхности

Видимая на экране поверхность и конструируемая поверхность также, как любая другая имеют заголовки типа *surface_t*. Чтобы получить указатели на них, можно воспользоваться функциями данного раздела.

- #define *ACONSTR* *getConstrAlpSurface* ()
Конструируемая поверхность с альфа-каналом.
- #define *ASCREEN* *getScreenAlpSurface* ()
Видимая поверхность с альфа-каналом.
- #define *CONSTR* *getConstrSurface* ()
Конструируемая поверхность.
- *surface_t* * *getConstrAlpSurface* (void)
Конструируемая поверхность с альфа-каналом.
- *surface_t* * *getConstrSurface* (void)
Конструируемая поверхность.
- *surface_t* * *getScreenAlpSurface* (void)
Видимая поверхность с альфа-каналом.
- *surface_t* * *getScreenSurface* (void)
Видимая поверхность.
- #define *SCREEN* *getScreenSurface* ()
Видимая поверхность.

Входные форматы оверлея YUV420 / YUV422

- #define *OT_INRGB888_PLANAR* (0x100)
- #define *OT_YUV420_COMBINED* (0x00)
- #define *OT_YUV420_INTERLEAVED* (0x04)
- #define *OT_YUV420_MB_COMBINED* (0x02)
- #define *OT_YUV420_MB_PLANAR* (0x03)
- #define *OT_YUV420_PLANAR* (0x01)
- #define *OT_YUV422_COMBINED* (0x10)
- #define *OT_YUV422_INTERLEAVED* (0x14)
- #define *OT_YUV422_MB_COMBINED* (0x12)
- #define *OT_YUV422_MB_PLANAR* (0x13)
- #define *OT_YUV422_PLANAR* (0x11)

Входные форматы оверлея YUV420

Устаревшие макросы - оставлены для совместимости со старыми проектами.

- `#define OT_MB_INTERLEAVED OT_YUV420_INTERLEAVED`
- `#define OT_MB_PLANAR OT_YUV420_MB_PLANAR`
- `#define OT_MB_UV_COMBINED OT_YUV420_MB_COMBINED`
- `#define OT_PLANAR OT_YUV420_PLANAR`
- `#define OT_UV_COMBINED OT_YUV420_COMBINED`

Инициализация и управление графическим адаптером

Выбор режима инициализации графического адаптера. Функции управления и опроса состояния.

- void * `getConstr2Buffer` (void)
Получить указатель на конструируемый 2-ой экран.
- void * `getConstrBuffer` (void)
Получить указатель на конструируемый экран.
- void * `getLFB` (void)
Получить указатель на начало видеопамати.
- void * `getScreen2Buffer` (void)
Получить указатель на видимый 2-ой экран.
- signed long `getScreenBitsPerPixel` (void)
Опрос количества бит на точку.
- void * `getScreenBuffer` (void)
Получить указатель на видимый экран.
- signed long `getScreenHeight` (void)
Опрос количества строк на экране.
- signed long `getScreenPitch` (void)
Опрос реальной длины строки в памяти.
- signed long `getScreenWidth` (void)
Опрос видимой ширины экрана.
- int `initLvdsDisplay` ()
Инициализация адаптера LVDS.
- void `set_lvds_mode` (int mode)
Смена режима LVDS.
- void `set_lvds_param` (lvds_param_t lp)
Выбор разрешения LVDS.
- int `setVideoMode` (int MODE, int BPP)
Инициализация адаптера HDMI / TV-Out.
- int `waitVerticalRetrace` ()
Ожидание обратного хода луча.

Работа с поверхностями

- int `bitBlt` (HDC dst, int dstX, int dstY, int Width, int Height, HDC src, int srcX, int srcY, unsigned char alpha, unsigned int color, int options)
Копирование прямоугольной области.
- void `deleteSurface` (surface_t *ps)
Удалить поверхность.
- void `deleteSurfaceDataArray` (surface_t *ps)
Освободить массив данных поверхности.
- int `fillRect` (HDC dst, int dstX, int dstY, int Width, int Height, unsigned char alpha, unsigned int color, int options)
Закрасить прямоугольную область на поверхности.
- int `FlipScreenAndConstr` ()
Смена поверхностей.
- int `init_2D_engine` ()

- *Инициализация 2D акселератора.*
• `surface_t * loadBMPSurface (char *fileName)`
Загрузить поверхность из BMP.
- `surface_t * loadJPEGSurface (char *fileName)`
Загрузить поверхность из JPG.
- `surface_t * loadPNGSurface (char *fileName)`
Загрузить поверхность из PNG.
- `surface_t * loadRawSurface (char *fileName, int W, int H, int F)`
Загрузить поверхность из файла.
- `surface_t * newSurface (int W, int H, g2d_data_fmt F)`
Создать новую поверхность.
- `int stretchBlit (HDC dst, int dstX, int dstY, int dstW, int dstH, HDC src, int srcX, int srcY, int srcW, int srcH, unsigned char alpha, unsigned int color, int options)`
Копирование прямоугольной области с масштабированием.

Работа с оверлеями

- `void OverlayClose (void)`
Закреть оверлей.
- `void OverlayInit (void)`
Инициализация режима оверлея.
- `void OverlayOpen (int BaseAddr[3], int X, int Y, int srcWidth, int srcHeight, int dstWidth, int dstHeight, int OType)`
Открыть оверлей на экране.
- `void OverlayOpenDI (int BaseAddr[3], int X, int Y, int srcWidth, int srcHeight, int dstWidth, int dstHeight, int OType)`
Открыть оверлей с аппаратным деинтерлейсингом источника.
- `void OverlaySetAddr (int A[3])`
Смена области оверлея.
- `int setOverlayPriority (int P)`
Смена приоритета области оверлея.

19.1.1. Подробное описание

Библиотека аппаратной поддержки 2D-графики для процессора **Allwinner A20**.

Подключение:

```
#include <multimedia/a20graph.h>
```

Makefile:

```
LIBRARIES += -l_a20graph -l_png -l_z
```

См. также

Общее описание работы с графической подсистемой в главе *Графическая подсистема*.

19.1.2. Макросы

19.1.2.1. ACONSTR

```
#define ACONSTR getConstrAlpSurface ()
```

Короткая запись функции *getConstrAlpSurface()*.

19.1.2.2. ASCREEN

```
#define ASCREEN getScreenAlpSurface ()
```

Короткая запись функции *getScreenAlpSurface()*.

19.1.2.3. CONSTR

```
#define CONSTR getConstrSurface ()
```

Короткая запись функции *getConstrSurface()*.

19.1.2.4. OT_INRGB888_PLANAR

```
#define OT_INRGB888_PLANAR (0x100)
```

19.1.2.5. OT_MB_INTERLEAVED

```
#define OT_MB_INTERLEAVED OT_YUV420_INTERLEAVED
```

19.1.2.6. OT_MB_PLANAR

```
#define OT_MB_PLANAR OT_YUV420_MB_PLANAR
```

19.1.2.7. OT_MB_UV_COMBINED

```
#define OT_MB_UV_COMBINED OT_YUV420_MB_COMBINED
```

19.1.2.8. OT_PLANAR

```
#define OT_PLANAR OT_YUV420_PLANAR
```

19.1.2.9. OT_UV_COMBINED

```
#define OT_UV_COMBINED OT_YUV420_COMBINED
```

19.1.2.10. OT_YUV420_COMBINED

```
#define OT_YUV420_COMBINED (0x00)
```

19.1.2.11. OT_YUV420_INTERLEAVED

```
#define OT_YUV420_INTERLEAVED (0x04)
```

19.1.2.12. OT_YUV420_MB_COMBINED

```
#define OT_YUV420_MB_COMBINED (0x02)
```

19.1.2.13. OT_YUV420_MB_PLANAR

```
#define OT_YUV420_MB_PLANAR (0x03)
```

19.1.2.14. OT_YUV420_PLANAR

```
#define OT_YUV420_PLANAR (0x01)
```

19.1.2.15. OT_YUV422_COMBINED

```
#define OT_YUV422_COMBINED (0x10)
```

19.1.2.16. OT_YUV422_INTERLEAVED

```
#define OT_YUV422_INTERLEAVED (0x14)
```

19.1.2.17. OT_YUV422_MB_COMBINED

```
#define OT_YUV422_MB_COMBINED (0x12)
```

19.1.2.18. OT_YUV422_MB_PLANAR

```
#define OT_YUV422_MB_PLANAR (0x13)
```


19.1.2.19. OT_YUV422_PLANAR

```
#define OT_YUV422_PLANAR (0x11)
```

19.1.2.20. SCREEN

```
#define SCREEN getScreenSurface ()
```

Короткая запись функции *getScreenSurface()*.

19.1.3. Типы

19.1.3.1. HDC

```
typedef surface_t* HDC
```

19.1.3.2. surface_t

```
typedef g2d_image surface_t
```

19.1.4. Перечисления

19.1.4.1. g2d_blt_flags

```
enum g2d_blt_flags
```

Правила (опции) копирования областей графики. Для альфа-канала значения параметра **alpha**: 0-255 соответствуют значению прозрачности от 0.0 до 1.0.



Перемножение двух параметров прозрачности так же дают значение от 0.0 до 1.0.

Элементы перечислений	
G2D_BLT_NONE	Простое копирование.
G2D_BLT_PIXEL_ALPHA	Копирование с использованием альфа-канала. Используется параметр прозрачности, указанной в качестве A формата цвета.
G2D_BLT_PLANE_ALPHA	Копирование с использованием альфа-канала. Используется параметр прозрачности, указанной как аргумент функции alpha .
G2D_BLT_MULTI_ALPHA	Копирование с использованием альфа-канала. Используется параметр прозрачности, полученным перемножением <i>G2D_BLT_PIXEL_ALPHA</i> и <i>G2D_BLT_PLANE_ALPHA</i> .

Продолжение на следующей странице

Элементы перечислений	
G2D_BLT_SRC_COLORKEY	Копирование с отбрасыванием ключевого цвета исходной поверхности.
G2D_BLT_DST_COLORKEY	Копирование с отбрасыванием ключевого цвета результирующей поверхности. См. также <i>Ошибка ColorKey для результирующей поверхности</i>
G2D_BLT_FLIP_HORIZONTAL	Копирование с отражением по горизонтали.
G2D_BLT_FLIP_VERTICAL	Копирование с отражением по вертикали.
G2D_BLT_ROTATE90	Копирование с поворотом на 90° по направлению против часовой стрелки. См. также <i>Ошибка при повороте поверхности на 45°</i>
G2D_BLT_ROTATE180	Копирование с поворотом на 180° по направлению против часовой стрелки.
G2D_BLT_ROTATE270	Копирование с поворотом на 270° по направлению против часовой стрелки. См. также <i>Ошибка при повороте поверхности на 45°</i>
G2D_BLT_MIRROR45	Копирование с отражением по диагонали 45°. См. также <i>Ошибка при повороте поверхности на 45°</i>
G2D_BLT_MIRROR135	Копирование с отражением по диагонали 135°. См. также <i>Ошибка при повороте поверхности на 45°</i>

```

00316      {
00318      G2D_BLT_NONE           = 0x00000000,
00323      G2D_BLT_PIXEL_ALPHA  = 0x00000001,
00328      G2D_BLT_PLANE_ALPHA  = 0x00000002,
00334      G2D_BLT_MULTI_ALPHA  = 0x00000004,
00338      G2D_BLT_SRC_COLORKEY = 0x00000008,
00343      G2D_BLT_DST_COLORKEY = 0x00000010,
00345      G2D_BLT_FLIP_HORIZONTAL = 0x00000020,
00347      G2D_BLT_FLIP_VERTICAL   = 0x00000040,
00351      G2D_BLT_ROTATE90       = 0x00000080,
00353      G2D_BLT_ROTATE180      = 0x00000100,
00357      G2D_BLT_ROTATE270      = 0x00000200,
00361      G2D_BLT_MIRROR45       = 0x00000400,
00365      G2D_BLT_MIRROR135     = 0x00000800,
00366 } g2d_blt_flags;
    
```

19.1.4.2. `g2d_data_fmt`enum `g2d_data_fmt`

Форматы смешивания цвета.

Элементы перечислений

`G2D_FMT_ARGB_AYUV8888`**Усм.** То же, что `G2D_FMT_ARGB8888` (не рекомендуется использовать в новых проектах).`G2D_FMT_BGRA_VUYA8888`**Усм.** То же, что `G2D_FMT_BGRA8888` (не рекомендуется использовать в новых проектах).`G2D_FMT_ABGR_AVUY8888`**Усм.** То же, что `G2D_FMT_ABGR8888` (не рекомендуется использовать в новых проектах).`G2D_FMT_RGBA_YUVA8888`**Усм.** То же, что `G2D_FMT_RGBA8888` (не рекомендуется использовать в новых проектах).`G2D_FMT_ARGB8888`Формат цвета с альфа-каналом **ARGB** 8-8-8-8.`G2D_FMT_BGRA8888`Формат цвета с альфа-каналом **BGRA** 8-8-8-8.`G2D_FMT_ABGR8888`Формат цвета с альфа-каналом **ABGR** 8-8-8-8.`G2D_FMT_RGBA8888`Формат цвета с альфа-каналом **RGBA** 8-8-8-8.`G2D_FMT_XRGB8888``G2D_FMT_BGRX8888``G2D_FMT_XBGR8888``G2D_FMT_RGBX8888``G2D_FMT_ARGB4444``G2D_FMT_ABGR4444``G2D_FMT_RGBA4444``G2D_FMT_BGRA4444``G2D_FMT_ARGB1555``G2D_FMT_ABGR1555``G2D_FMT_RGBA5551``G2D_FMT_BGRA5551``G2D_FMT_RGB565`

Продолжение на следующей странице

Элементы перечислений

G2D_FMT_BGR565

G2D_FMT_IYUV422

G2D_FMT_8BPP_MONO

G2D_FMT_4BPP_MONO

G2D_FMT_2BPP_MONO

G2D_FMT_1BPP_MONO

G2D_FMT_PYUV422UVC

G2D_FMT_PYUV420UVC

G2D_FMT_PYUV411UVC

G2D_FMT_PYUV422

G2D_FMT_PYUV420

G2D_FMT_PYUV411

G2D_FMT_8BPP_PALETTE

G2D_FMT_4BPP_PALETTE

G2D_FMT_2BPP_PALETTE

G2D_FMT_1BPP_PALETTE

```

00204      {
00205      G2D_FMT_ARGB_AYUV8888 = (0x0),
00206      G2D_FMT_BGRA_VUYA8888 = (0x1),
00207      G2D_FMT_ABGR_AVUY8888 = (0x2),
00208      G2D_FMT_RGBA_YUVA8888 = (0x3),
00209
00210      G2D_FMT_ARGB8888 = (0x0),
00211      G2D_FMT_BGRA8888 = (0x1),
00212      G2D_FMT_ABGR8888 = (0x2),
00213      G2D_FMT_RGBA8888 = (0x3),
00214
00215      G2D_FMT_XRGB8888 = (0x4),
00216      G2D_FMT_BGRX8888 = (0x5),
00217      G2D_FMT_XBGR8888 = (0x6),
00218      G2D_FMT_RGBX8888 = (0x7),
00219
00220      G2D_FMT_ARGB4444 = (0x8),
00221      G2D_FMT_ABGR4444 = (0x9),
00222      G2D_FMT_RGBA4444 = (0xA),
00223      G2D_FMT_BGRA4444 = (0xB),
00224
00225      G2D_FMT_ARGB1555 = (0xC),
00226      G2D_FMT_ABGR1555 = (0xD),
00227      G2D_FMT_RGBA5551 = (0xE),
00228      G2D_FMT_BGRA5551 = (0xF),
00229
00230      G2D_FMT_RGB565 = (0x10),
00231      G2D_FMT_BGR565 = (0x11),
    
```

```

00232
00233     G2D_FMT_IYUV422 = (0x12),
00234
00235     G2D_FMT_8BPP_MONO = (0x13),
00236     G2D_FMT_4BPP_MONO = (0x14),
00237     G2D_FMT_2BPP_MONO = (0x15),
00238     G2D_FMT_1BPP_MONO = (0x16),
00239
00240     G2D_FMT_PYUV422UVC = (0x17),
00241     G2D_FMT_PYUV420UVC = (0x18),
00242     G2D_FMT_PYUV411UVC = (0x19),
00243
00244     /* just for output format */
00245     G2D_FMT_PYUV422 = (0x1A),
00246     G2D_FMT_PYUV420 = (0x1B),
00247     G2D_FMT_PYUV411 = (0x1C),
00248
00249     /* just for input format */
00250     G2D_FMT_8BPP_PALETTE = (0x1D),
00251     G2D_FMT_4BPP_PALETTE = (0x1E),
00252     G2D_FMT_2BPP_PALETTE = (0x1F),
00253     G2D_FMT_1BPP_PALETTE = (0x20),
00254
00255 } g2d_data_fmt;

```

19.1.4.3. g2d_fillrect_flags

enum *g2d_fillrect_flags*

Правила использования альфа-канала при заливке полигонов. Значения параметра **alpha**: 0-255 соответствуют значению прозрачности от 0.0 до 1.0.



Перемножение двух параметров прозрачности так же дают значение от 0.0 до 1.0.

Элементы перечислений

G2D_FIL_NONE	Заливка без учёта прозрачности.
G2D_FIL_PIXEL_ALPHA	Заливка с прозрачностью, указанной в качестве A формата цвета.
G2D_FIL_PLANE_ALPHA	Заливка с прозрачностью, указанной как аргумент функции alpha .
G2D_FIL_MULTI_ALPHA	При заливке используется прозрачность, полученная как результат перемножения значений <i>G2D_FIL_PIXEL_ALPHA</i> и <i>G2D_FIL_PLANE_ALPHA</i> .

```

00301     {
00302     G2D_FIL_NONE = 0x00000000,
00303     G2D_FIL_PIXEL_ALPHA = 0x00000001,
00304     G2D_FIL_PLANE_ALPHA = 0x00000002,
00305     G2D_FIL_MULTI_ALPHA = 0x00000004,
00307 } g2d_fillrect_flags;

```

19.1.4.4. g2d_pixel_seq

enum *g2d_pixel_seq*

Элементы перечислений

G2D_SEQ_NORMAL

G2D_SEQ_VYUY

G2D_SEQ_YVYU

G2D_SEQ_VUVU

G2D_SEQ_P10

G2D_SEQ_P01

G2D_SEQ_P3210

G2D_SEQ_P0123

G2D_SEQ_P76543210

G2D_SEQ_P67452301

G2D_SEQ_P10325476

G2D_SEQ_P01234567

G2D_SEQ_2BPP_BIG_BIG

G2D_SEQ_2BPP_BIG_LITTER

G2D_SEQ_2BPP_LITTER_BIG

G2D_SEQ_2BPP_LITTER_LITTER

G2D_SEQ_1BPP_BIG_BIG

G2D_SEQ_1BPP_BIG_LITTER

G2D_SEQ_1BPP_LITTER_BIG

G2D_SEQ_1BPP_LITTER_LITTER

```
00257         {
00258     G2D_SEQ_NORMAL = 0x0,
00259
00260     /* for interleaved yuv422 */
00261     G2D_SEQ_VYUY = 0x1,
00262     G2D_SEQ_YVYU = 0x2,
00263
00264     /* for uv_combined yuv420 */
00265     G2D_SEQ_VUVU = 0x3,
00266
00267     /* for 16bpp rgb */
```

```

00268     G2D_SEQ_P10 = 0x4,
00269     G2D_SEQ_P01 = 0x5,
00270
00271     /* planar format or 8bpp rgb */
00272     G2D_SEQ_P3210 = 0x6,
00273     G2D_SEQ_P0123 = 0x7,
00274
00275     /* for 4bpp rgb */
00276     G2D_SEQ_P76543210 = 0x8, /* 7,6,5,4,3,2,1,0 */
00277     G2D_SEQ_P67452301 = 0x9, /* 6,7,4,5,2,3,0,1 */
00278     G2D_SEQ_P10325476 = 0xA, /* 1,0,3,2,5,4,7,6 */
00279     G2D_SEQ_P01234567 = 0xB, /* 0,1,2,3,4,5,6,7 */
00280
00281     /* for 2bpp rgb */
00282     G2D_SEQ_2BPP_BIG_BIG = 0xC, /*
15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0 */
00283     G2D_SEQ_2BPP_BIG_LITTER = 0xD, /*
12,13,14,15,8,9,10,11,4,5,6,7,0,1,2,3 */
00284     G2D_SEQ_2BPP_LITTER_BIG = 0xE, /*
3,2,1,0,7,6,5,4,11,10,9,8,15,14,13,12 */
00285     G2D_SEQ_2BPP_LITTER_LITTER = 0xF, /*
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 */
00286
00287     /* for 1bpp rgb */
00288     G2D_SEQ_1BPP_BIG_BIG = 0x10, /*
31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
*/
00289     G2D_SEQ_1BPP_BIG_LITTER = 0x11, /*
24,25,26,27,28,29,30,31,16,17,18,19,20,21,22,23,8,9,10,11,12,13,14,15,0,1,2,3,4,5,6,7
*/
00290     G2D_SEQ_1BPP_LITTER_BIG = 0x12, /*
7,6,5,4,3,2,1,0,15,14,13,12,11,10,9,8,23,22,21,20,19,18,17,16,31,30,29,28,27,26,25,24
*/
00291     G2D_SEQ_1BPP_LITTER_LITTER = 0x13, /*
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
*/
00292 } g2d_pixel_seq;

```

19.1.4.5. lvds_param_t

enum *lvds_param_t*

Выбор режима работы графического адаптера (LVDS).

Элементы перечислений

LVDS_1920x1080	Режим работы адаптера: LVDS 1920x1080.
LVDS_1920x360	Режим работы адаптера: LVDS 1920x360.
LVDS_1024x768	Режим работы адаптера: LVDS 1024x768.
LVDS_800x600	Режим работы адаптера: LVDS 800x600.
LVDS_800x480	Режим работы адаптера: LVDS 800x480.
LVDS_1280x800	Режим работы адаптера: LVDS 1280x800.

Продолжение на следующей странице

Элементы перечислений

LVDS_158x1920	Режим работы адаптера: LVDS 158x1920.
LVDS_1920x165	Режим работы адаптера: LVDS 1920x165.
LVDS_1400x1050	Режим работы адаптера: LVDS 1400x1050.

```

00048      {
00049      LVDS_1920x1080 = 0,
00050      LVDS_1920x360,
00051      LVDS_1024x768,
00052      LVDS_800x600,
00053      LVDS_800x480,
00054      LVDS_1280x800,
00055      LVDS_158x1920,
00056      LVDS_1920x165,
00057      LVDS_1400x1050,
00058 } lvds_param_t;
    
```

19.1.4.6. VideoModes

enum *VideoModes*

Выбор режима работы графического адаптера (HDMI/TV-Out).

Элементы перечислений

MODE_640x480	Режим работы адаптера: HDMI 640x480.
MODE_800x480	Режим работы адаптера: HDMI 800x480.
MODE_800x600	Режим работы адаптера: HDMI 800x600.
MODE_1024x768	Режим работы адаптера: HDMI 1024x768.
MODE_1280x720	Режим работы адаптера: HDMI 1280x720.
MODE_1280x768	Режим работы адаптера: HDMI 1280x768.
MODE_1280x800	Режим работы адаптера: HDMI 1280x800.
MODE_1368x768	Режим работы адаптера: HDMI 1368x768.
MODE_1280x1024	Режим работы адаптера: HDMI 1280x1024.
MODE_1920x1080	Режим работы адаптера: HDMI 1920x1080.
MODE_TV	Режим работы адаптера: TV-Out.

```

00031      {
00032      MODE_640x480 = 0,
    
```



```

00033     MODE_800x480,
00034     MODE_800x600,
00035     MODE_1024x768,
00036     MODE_1280x720,
00037     MODE_1280x768,
00038     MODE_1280x800,
00039     MODE_1368x768,
00040     MODE_1280x1024,
00041     MODE_1920x1080,
00042     MODE_TV
00043 } VideoModes;
    
```

19.1.5. Функции

19.1.5.1. bitBlt()

```

int bitBlt (
    HDC dst,
    int dstX,
    int dstY,
    int Width,
    int Height,
    HDC src,
    int srcX,
    int srcY,
    unsigned char alpha,
    unsigned int color,
    int options )
    
```

Функция копирует прямоугольную область из одной поверхности в другую.

Аргументы

<i>dst</i>	Указатель на поверхность – получатель.
<i>dstX, dstY</i>	Координаты левого верхнего угла на поверхности – получателе.
<i>Width, Height</i>	Ширина и высота копируемой области.
<i>src</i>	Указатель на поверхность – источник.
<i>srcX, srcY</i>	Координаты левого верхнего угла на поверхности – источнике.
<i>alpha</i>	Прозрачность копии.
<i>color</i>	Ключевой цвет в формате, выбранном при создании поверхности.
<i>options</i>	Опции — комбинация из значений перечисления <i>g2d_blt_flags</i> .

Возвращает

OK При нормальном завершении, или код ошибки.

См. также

Ошибка при повороте поверхности на 45°. Ошибка ColorKey для результирующей поверхности.

19.1.5.2. deleteSurface()

```
void deleteSurface (  
    surface_t * ps )
```

Функция удаляет из памяти указанную поверхность и все связанные с ней ресурсы, созданные с помощью `newSurface()`.



Для удаления поверхностей с заранее освобожденным массивом данных (см. `deleteSurfaceDataArray()`) следует воспользоваться обычным способом освобождения памяти `free()`.

Аргументы

`ps` Указатель на поверхность.

19.1.5.3. deleteSurfaceDataArray()

```
void deleteSurfaceDataArray (  
    surface_t * ps )
```

Функция освобождает память, выделенную под массив данных цвета при создании поверхности. Это нужно для подключения к поверхности массива памяти, полученных из других источников, например, из аппаратного декодера. Если в последующем нужно удалить саму поверхность (без освобождения памяти массива данных) следует делать просто через `free()`.



После освобождения памяти следует сразу же подключить к поверхности новый массив данных, иначе попытка вывода поверхности на экран приведет к ошибке.

Аргументы

`ps` Указатель на поверхность.

19.1.5.4. fillRect()

```
int fillRect (  
    HDC dst,  
    int dstX,  
    int dstY,  
    int Width,  
    int Height,  
    unsigned char alpha,  
    unsigned int color,
```

int options)

Функция закрашивает на указанной поверхности прямоугольную область в указанном месте указанным цветом.

Аргументы	
<i>dst</i>	Указатель на поверхность, в которой требуется проводить действие.
<i>dstX, dstY</i>	Координаты левого верхнего угла прямоугольника на поверхности.
<i>Width, Height</i>	Ширина и высота закрашиваемой области.
<i>alpha</i>	Прозрачность заливки (255 — не прозрачная, 0 — полностью прозрачная).
<i>color</i>	Цвет заливки в формате, выбранном при создании поверхности.
<i>options</i>	Опции заливки из перечисления <i>g2d_fillrect_flags</i> .

Возвращает

OK При нормальном завершении, или код ошибки.

См. также

Ошибка при повороте поверхности на 45°. Ошибка ColorKey для результирующей поверхности.

19.1.5.5. FlipScreenAndConstr()

int FlipScreenAndConstr ()

Функция меняет местами видимую (*SCREEN*) и конструируемую (*CONSTR*) поверхности.

Возвращает

Всегда 0.

19.1.5.6. getConstr2Buffer()

void* getConstr2Buffer (void)

Возвращает

Указатель на конструируемый 2-ой экран.

19.1.5.7. getConstrAlpSurface()

*surface_t** getConstrAlpSurface (

```
void )
```

Функция возвращает указатель на конструируемую поверхность с альфа-каналом.

Возвращает

Указатель на конструируемую поверхность с альфа-каналом *surface_t*.

19.1.5.8. getConstrBuffer()

```
void* getConstrBuffer (  
    void )
```

Возвращает

Указатель на конструируемый экран.

19.1.5.9. getConstrSurface()

```
surface_t* getConstrSurface (  
    void )
```

Функция возвращает указатель на конструируемую поверхность.

Возвращает

Указатель на конструируемую поверхность *surface_t*.

19.1.5.10. getLFB()

```
void* getLFB (  
    void )
```

Возвращает

Указатель на начало видеопамати.

19.1.5.11. getScreen2Buffer()

```
void* getScreen2Buffer (  
    void )
```

Возвращает

Указатель на видимый 2-ой экран.

19.1.5.12. getScreenAlpSurface()

```
surface_t* getScreenAlpSurface (  
    void )
```

Функция возвращает указатель на видимую на экране поверхность с алфа-каналом.

Возвращает

Указатель на видимую на экране поверхность *surface_t*.

19.1.5.13. getScreenBitsPerPixel()

```
signed long getScreenBitsPerPixel (  
    void )
```

Возвращает

Количество бит на точку в выданном формате цвета.

19.1.5.14. getScreenBuffer()

```
void* getScreenBuffer (  
    void )
```

Возвращает

Указатель на видимый экран.

19.1.5.15. getScreenHeight()

```
signed long getScreenHeight (  
    void )
```

Возвращает

Количество строк выводимых на экран.

19.1.5.16. getScreenPitch()

```
signed long getScreenPitch (  
    void )
```

Возвращает

Длину строки в памяти в байтах.

19.1.5.17. getScreenSurface()

```
surface_t* getScreenSurface (
    void )
```

Функция возвращает указатель на видимую на экране поверхность.

Возвращает

Указатель на видимую на экране поверхность *surface_t*.

19.1.5.18. getScreenWidth()

```
signed long getScreenWidth (
    void )
```

Возвращает

Видимую ширину экрана в пикселях.

19.1.5.19. init_2D_engine()

```
int init_2D_engine ( )
```

Функция инициализирует аппаратный **2D** акселератор для работы с поверхностями.

Возвращает

Всегда 0.

19.1.5.20. initLvdsDisplay()

```
int initLvdsDisplay ( )
```

Функция инициализирует видеоадаптер в режиме **LVDS**. По умолчанию **LVDS** будет инициализирован в режиме **NS** с разрешением **FULL HD**. Для инициализации в режиме **JEDA** нужно вызвать функцию *set_lvds_mode()* с параметром **1**. Если требуется установить другое разрешение, то перед инициализацией следует вызвать функцию *set_lvds_param()*.

Возвращает

OK При успешном завершении.

ERROR При неудаче.

19.1.5.21. loadBMPSurface()

```
surface_t* loadBMPSurface (
    char * fileName )
```

Функция загружает поверхность данными из **BMP** – файла. Поддерживаемые форматы файла: **RGB 24-бита, RGBX и RGBA 32-бита**. Форматы 16-бит и ниже не поддерживаются

Аргументы

fileName Имя файла на диске.

Возвращает

Указатель на созданную поверхность *surface_t*.

19.1.5.22. loadJPEGSurface()

```
surface_t* loadJPEGSurface (  
    char * fileName )
```

Функция загружает поверхность данными из **JPG** - Файла. Поддерживаемый формат файла: **YCbCr 24-бита в baseline кодировке. Ограничения декодера:**

- Изображение декодируется некорректно, если одна из его сторон нечётной длины.
- Максимальный размер изображения 3840x2160.
- Не поддерживается **progressive** формат.
- Не поддерживаются файлы, имеющие не 3 компонента (Y, Cb, Cr). Это могут быть некоторые чёрно-белые изображения или изображения с другой цветовой моделью.
- **Важно!** Для корректной работы функции требуется перед вызовом включить **VE** (видеоэнкодер), а после — выключить командами *ve_open()* и *ve_close()*. Файлы формата **JPEG** распознаются системой некорректно, чтобы открыть изображение, введите имя в виде: **f_name~1.jpe**

Аргументы

fileName Имя файла на диске.

Возвращает

Указатель на созданную поверхность *surface_t*.

19.1.5.23. loadPNGSurface()

```
surface_t* loadPNGSurface (  
    char * fileName )
```

Функция загружает поверхность данными из **PNG** - файла. Для работы функции понадобится подключение дополнительных библиотек **-l_png** и **-l_z** в *Makefile*.

Аргументы

<i>fileName</i>	Имя файла на диске.
-----------------	---------------------

Возвращает

Указатель на созданную поверхность *surface_t*.**19.1.5.24. loadRawSurface()**

```
surface_t* loadRawSurface (  
    char * fileName,  
    int W,  
    int H,  
    int F)
```

Функция загружает поверхность *сырыми* данными из файла.

Аргументы

<i>fileName</i>	Имя файла на диске.
-----------------	---------------------

<i>W</i>	Ширина в пикселях.
----------	--------------------

<i>H</i>	Высота в пикселях.
----------	--------------------

<i>F</i>	Формат поверхности из перечисления <i>g2d_data_fmt</i> .
----------	--

Возвращает

Указатель на созданную поверхность *surface_t*.**19.1.5.25. newSurface()**

```
surface_t* newSurface (  
    int W,  
    int H,  
    g2d_data_fmt F)
```

Функция создает новую поверхность с указанными параметрами.

Аргументы

<i>W</i>	Ширина в пикселях.
----------	--------------------

<i>H</i>	Высота в пикселях.
----------	--------------------

<i>F</i>	Формат цвета поверхности из перечисления <i>g2d_data_fmt</i> .
----------	--

Возвращает

Указатель на созданную поверхность *surface_t*.

19.1.5.26. OverlayClose()

```
void OverlayClose (
    void )
```

Функция закрывает ранее открытый на экране оверлей.

19.1.5.27. OverlayInit()

```
void OverlayInit (
    void )
```

Функция инициализирует режим аппаратного оверлея.

19.1.5.28. OverlayOpen()

```
void OverlayOpen (
    int BaseAddr[3],
    int X,
    int Y,
    int srcWidth,
    int srcHeight,
    int dstWidth,
    int dstHeight,
    int OType )
```

Функция открывает аппаратный оверлей на экране.

Аргументы	
<i>BaseAddr</i>	Адреса в памяти зоны оверлея.
<i>X,Y</i>	Координаты на экране левого верхнего угла оверлея.
<i>srcWidth,srcHeight</i>	Размеры оверлея в памяти в точках.
<i>dstWidth,dstHeight</i>	Размеры оверлея на экране в точках.
<i>OType</i>	Тип оверлея — одно из значений, перечисленных в группе <i>Входные форматы оверлея YUV422</i> .

19.1.5.29. OverlayOpenDI()

```
void OverlayOpenDI (
    int BaseAddr[3],
    int X,
    int Y,
    int srcWidth,
    int srcHeight,
```

```
int dstWidth,
int dstHeight,
int OType )
```

Функция открывает аппаратный оверлей на экране. Используется функция аппаратного деинтерлейсинга источника.

Аргументы	
<i>BaseAddr</i>	Адреса в памяти зоны оверлея.
<i>X,Y</i>	Координаты на экране левого верхнего угла оверлея.
<i>srcWidth,srcHeight</i>	Размеры оверлея в памяти в точках.
<i>dstWidth,dstHeight</i>	Размеры оверлея на экране в точках.
<i>OType</i>	Тип оверлея — одно из значений, перечисленных в группе Входные форматы оверлея YUV422 .

19.1.5.30. OverlaySetAddr()

```
void OverlaySetAddr (
    int A[3] )
```

Функция позволяет оперативно поменять адрес оверлейной области.

Аргументы	
<i>A</i>	Новые адреса в памяти зоны оверлея.

19.1.5.31. set_lvds_mode()

```
void set_lvds_mode (
    int mode )
```

Для выбора режима **LVDS** отличного от режима по умолчанию (**NS**) нужно вызвать данную функцию с параметром **1** перед инициализацией [initLvdsDisplay\(\)](#).

Аргументы	
<i>mode</i>	

19.1.5.32. set_lvds_param()

```
void set_lvds_param (
    lvds_param_t lp )
```

Для выбора разрешения **LVDS** отличного от разрешения по умолчанию (**FULL HD**) нужно вызвать

функцию перед инициализацией *initLvdsDisplay()*.

Аргументы

lp Разрешение экрана **LVDS** выбирается из списка *lvds_param_t*.

19.1.5.33. setOverlayPriority()

```
int setOverlayPriority (  
    int P )
```

Функция позволяет оперативно поменять приоритет оверлейной области. В зависимости от приоритета меняется порядок вывода областей памяти на экран. Самый высокий приоритет выводится поверх остальных.

Аргументы

P Приоритет области оверлея. Возможные значения 0–3.

19.1.5.34. setVideoMode()

```
int setVideoMode (  
    int MODE,  
    int BPP )
```

Функция инициализирует адаптер в режиме **HDMI** или **TV-Out**.

Аргументы

MODE Режим работы и разрешение выбирается из списка *VideoModes*.

BPP Количество бит на точку. Следует указать 32 или 16.

Возвращает

OK При успешном завершении.

ERROR При неудаче.

19.1.5.35. stretchBlit()

```
int stretchBlit (  
    HDC dst,  
    int dstX,  
    int dstY,  
    int dstW,  
    int dstH,  
    HDC src,
```

```
int srcX,  
int srcY,  
int srcW,  
int srcH,  
unsigned char alpha,  
unsigned int color,  
int options )
```

Функция копирует прямоугольную область из одной поверхности в другую с изменением масштаба.

Аргументы	
<i>dst</i>	Указатель на поверхность – получатель.
<i>dstX,dstY</i>	Координаты левого верхнего угла на поверхности – получателе.
<i>dstW,dstH</i>	Размеры на поверхности – получателе.
<i>src</i>	Указатель на поверхность – источник.
<i>srcX,srcY</i>	Координаты левого верхнего угла на поверхности – источнике.
<i>srcW,srcH</i>	Размеры на поверхности – источнике.
<i>alpha</i>	Прозрачность копии.
<i>color</i>	Ключевой цвет в формате, выбранном при создании поверхности.
<i>options</i>	Опции из перечисления <i>g2d_blt_flags</i> .

Возвращает

OK При нормальном завершении, или код ошибки.

См. также

Ошибка при повороте поверхности на 45°. Ошибка ColorKey для результирующей поверхности.

19.1.5.36. waitVerticalRetrace()

```
int waitVerticalRetrace ( )
```

Функция ожидает начала обратного хода луча на мониторе для избежания появления строба. Является синонимом функции `supxi_wait_retrace()`.

Возвращает

Всегда 0.

19.1.6. Переменные

19.1.6.1. Display

```
struct Display Display [extern]
```

19.2. Файл arch.h

Описание аппаратной части текущего проекта.

Структуры данных

- struct *tDrvBit*
Описание одного бита регистра.
- struct *tDrvBitGroup*
Описание группы бит регистра.
- struct *tDrvGpio*
Описание аппаратных линий вывода.

Добавление параметров

Дополнительные параметры конфигурации могут быть добавлены в конце списка из программы пользователя.

- void *archAddInt* (const char *key, int value, const char *description)
Добавление целочисленного параметра.
- void *archAddIntArray* (const char *key, int *values, int count, const char *description)
Добавление массива целых чисел.
- void *archAddString* (const char *key, const char *text)
Добавление строкового параметра.

Чтение параметров из списка

Параметры конфигурации используются при инициализации библиотек и могут быть использованы в программе пользователя.

- bool *archCheckString* (const char *value, const char *pattern)
Функция сравнения строк.
- unsigned int *archGetGpio* (const char *key, bool *ok)
Получить номер GPIO из списка.
- int *archGetInt* (const char *key, bool *ok)
Получить целое значение из списка.
- const int * *archGetIntArray* (const char *key, int *count)
Получить массив целых чисел из списка.
- const char * *archGetString* (const char *key, bool *ok)
Получить строковое значение из списка.

Элементы драйверов устройств

Функции настройки регистров конфигурации используются при написании драйверов устройств.

- unsigned int *drvGetBit* (const *tDrvBit* *bit)
Получить значение бита в регистре.
- unsigned int *drvGetBitGroup* (const *tDrvBitGroup* *group)
Получить значение группы бит в регистре.
- void *drvInitBit* (*tDrvBit* *bit, unsigned int addr, unsigned int n)
Задать описание бита регистра.
- void *drvInitBitGroup* (*tDrvBitGroup* *group, unsigned int addr, unsigned int n, unsigned int mask)
Задать описание группы бит.
- void *drvInitGpio* (*tDrvGpio* *gpio, unsigned int pin, unsigned int mux)
Задать параметры линии ввода / вывода.
- bool *drvResetBit* (const *tDrvBit* *bit, unsigned int delay)
Запись бита и проверка автоматического снятия.

- void `drvSetBit` (const `tDrvBit` *bit, unsigned int value)
Установить бит в регистре.
- void `drvSetBitGroup` (const `tDrvBitGroup` *group, unsigned int value)
Установить группу бит в регистре.
- void `drvSetGpio` (const `tDrvGpio` *gpio)
Подключить линию ввода / вывода.

Дополнительно

- void `archFree` ()
Освобождение выделенной памяти.
- void `archPrint` ()
Печать таблицы параметров.

19.2.1. Подробное описание

Подключение:

```
#include <arch.h>
```

См. также

Подробное описание в разделе *Конфигурация аппаратной части*.

19.2.2. Функции

19.2.2.1. archAddInt()

```
void archAddInt (
    const char * key,
    int value,
    const char * description )
```

Добавление целого числа в конец списка параметров.

Аргументы	
<code>key</code>	Уникальное имя параметра. Рекомендуется брать из группы <i>макросов</i> , описывающих поля данных.
<code>value</code>	Добавляемое значение.
<code>description</code>	Описание параметра — актуально для вывода списка параметров на печать.

19.2.2.2. archAddIntArray()

```
void archAddIntArray (
    const char * key,
    int * values,
```

```
int count,  
const char * description )
```

Добавление массива целых чисел в конец списка параметров.

Аргументы	
<i>key</i>	Уникальное имя параметра. Рекомендуется брать из группы <i>макросов</i> , описывающих поля данных.
<i>values</i>	Указатель на массив значений.
<i>count</i>	Количество элементов массива.
<i>description</i>	Описание массива чисел — актуально для вывода списка параметров на печать.

19.2.2.3. archAddString()

```
void archAddString (  
    const char * key,  
    const char * text )
```

Добавление строки в конец списка параметров.

Аргументы	
<i>key</i>	Уникальное имя параметра. Рекомендуется брать из группы <i>макросов</i> , описывающих поля данных.
<i>text</i>	Добавляемая строка.

19.2.2.4. archCheckString()

```
bool archCheckString (  
    const char * value,  
    const char * pattern )
```

Функция не работает со списком и написана для удобства работы со строковыми параметрами. Найденное с помощью *archGetString()* значение можно сравнивать с зарезервированными строковыми константами.

Аргументы	
<i>value</i>	Найденное значение.
<i>pattern</i>	Шаблон – строковая константа.

Возвращает

Результат сравнения – *true* при полном совпадении, иначе *false*.

19.2.2.5. archFree()

```
void archFree ( )
```

Память может быть освобождена после инициализации всех библиотек.

19.2.2.6. archGetGpio()

```
unsigned int archGetGpio (
    const char * key,
    bool * ok )
```

Чтение номера **GPIO** из списка по ключу. Значение в списке должно быть строковым и содержать имя порта и номер пина. Стока может быть записана без пробелов, как в документации на процессор (**PG1**), либо в стиле *MULTEX-ARM* (**P_G + 1**). Если ключ не найден или найденное значение не является строкой – параметр **ok** будет содержать значение *false*.

Аргументы

key Уникальное имя параметра. Рекомендуется брать из группы *макросов*, описывающих поля данных.

ok Признак актуальности данных.

Возвращает

Номер линии **GPIO**.

19.2.2.7. archGetInt()

```
int archGetInt (
    const char * key,
    bool * ok )
```

Чтение целого числа из списка по ключу. Если ключ не найден или найденное значение не является числом – параметр **ok** будет содержать значение *false*.

Аргументы

key Уникальное имя параметра. Рекомендуется брать из группы *макросов*, описывающих поля данных.

ok Признак актуальности данных.

Возвращает

Найденное число если данные найдены, иначе *NULL*.

19.2.2.8. archGetIntArray()

```
const int* archGetIntArray (
    const char * key,
    int * count )
```

Поиск массива целых чисел в списке по ключу. Если ключ не найден или найденное значение не является массивом чисел – параметр **count** будет равен нулю.

Аргументы		
in	<i>key</i>	Уникальное имя параметра. Рекомендуется брать из группы <i>макросов</i> , описывающих поля данных.
out	<i>count</i>	Количество элементов массива, либо ноль, если данные не найдены.

Возвращает

Указатель на найденный массив, либо *NULL*.

19.2.2.9. archGetString()

```
const char* archGetString (
    const char * key,
    bool * ok )
```

Чтение строкового значения из списка по ключу. Если ключ не найден или найденное значение не является строкой – параметр **ok** будет содержать значение *false*.

Аргументы		
	<i>key</i>	Уникальное имя параметра. Рекомендуется брать из группы <i>макросов</i> , описывающих поля данных.
	<i>ok</i>	Признак актуальности данных.

Возвращает

Указатель на найденную строку если данные найдены и актуальны, иначе *NULL*.

19.2.2.10. archPrint()

```
void archPrint ( )
```

Список параметров выводится в консоль в режиме **DEBUG** после инициализации операционной системы. Можно вызвать дополнительно в программе пользователя.

19.2.2.11. drvGetBit()

```
unsigned int drvGetBit (
```

```
const tDrvBit * bit )
```

Функция возвращает значение указанного бита регистра конфигурации.

Аргументы

bit Указатель на описание запрашиваемого бита.

Возвращает

Прочитанное значение.

19.2.2.12. drvGetBitGroup()

```
unsigned int drvGetBitGroup (  
    const tDrvBitGroup * group )
```

Функция возвращает значение указанной группы бит регистра конфигурации.

Аргументы

group Указатель на описание запрашиваемой группы бит.

Возвращает

Прочитанное значение.

19.2.2.13. drvInitBit()

```
void drvInitBit (  
    tDrvBit * bit,  
    unsigned int addr,  
    unsigned int n )
```

Функция используется в инициализации конфигурации драйверов и заполняет параметры структуры описания бита регистра.

Аргументы

bit Указатель на описание бита регистра.

addr Адрес регистра – абсолютное значение.

n Номер бита в регистре.

19.2.2.14. drvInitBitGroup()

```
void drvInitBitGroup (  
    tDrvBitGroup * group,  
    unsigned int addr,  
    unsigned int n,  
    unsigned int mask )
```

Функция используется в инициализации конфигурации драйверов и заполняет параметры структуры описания группы бит регистра.

Аргументы

<i>group</i>	Указатель на описание группы бит.
<i>addr</i>	Адрес регистра – абсолютное значение.
<i>n</i>	Номер младшего бита группы в регистре.
<i>mask</i>	Маска группы бит.

19.2.2.15. drvInitGpio()

```
void drvInitGpio (  
    tDrvGpio * gpio,  
    unsigned int pin,  
    unsigned int mux )
```

Функция используется в инициализации конфигурации драйверов и заполняет параметры структуры описания линии порта ввода / вывода. Изначально все линии не активны. Активировать нужные линии следует при инициализации драйвера.

Аргументы

<i>gpio</i>	Заполняемое описание аппаратной части линии.
<i>pin</i>	Номер порта + номер бита, например, P_B+1 .
<i>mux</i>	Значение мультиплексора для подключения к аппаратному модулю.

19.2.2.16. drvResetBit()

```
bool drvResetBit (  
    const tDrvBit * bit,  
    unsigned int delay )
```

Записать бит в регистр конфигурации и дождаться его автоматического снятия в течение заданного времени. Сразу после записи бита выдерживается короткая пауза для контроля быстрых процессов. Если бит не снимается за короткое время функция ожидает снятия в течение **delay** тиков системного таймера.

Аргументы

bit Указатель на описание бита.

delay Количество миллисекунд. Допускается установка параметра *NO_WAIT*.

Возвращает

true если выставленный бит был автоматически снят в течение заданного времени, иначе – *false*.

19.2.2.17. drvSetBit()

```
void drvSetBit (  
    const tDrvBit * bit,  
    unsigned int value )
```

Функция записывает значение в указанный бит регистра конфигурации, не затрагивая остальные биты.

Аргументы

bit Указатель на описание изменяемого бита.

value Записываемое значение.

19.2.2.18. drvSetBitGroup()

```
void drvSetBitGroup (  
    const tDrvBitGroup * group,  
    unsigned int value )
```

Функция записывает значение в группу бит в регистре конфигурации, не затрагивая остальные биты.

Аргументы

group Указатель на описание изменяемой группы бит.

value Записываемое значение.

19.2.2.19. drvSetGpio()

```
void drvSetGpio (  
    const tDrvGpio * gpio )
```

Функция используется для подключения линий ввода / вывода при инициализации драйвера устройства. Для успешного подключения линия должна быть доступна для инициализируемого

устройства и активирована драйвером. При несоблюдении этих условий линия подключена не будет. Линии в драйверах подключаются наборами и игнорирование неактивированных линий является нормальной практикой.

Аргументы

prio Описание линии ввода / вывода.

19.3. Файл archdef.h

Зарезервированные строки описания аппаратной части.

Поля данных

Зарезервированные уникальные **ключи** для поиска параметров конфигурации проекта.

- #define `ARCH_BACKLIGHT_FREQUENCY` "backlight-freq"
Частота ШИМ канала подсветки.
- #define `ARCH_BACKLIGHT_GPIO` "backlight-pwm"
ШИМ канал подсветки.
- #define `ARCH_BOARD` "board-name"
Название платы.
- #define `ARCH_CPU` "cpu-name"
Название процессора.
- #define `ARCH_LED_DISK` "led-disk"
Индикация работы файловой системы.
- #define `ARCH_LED_SYSTEM` "led-system"
Индикация работы системного таймера.
- #define `ARCH_LEDLINE_PWM` "ledline-pwm"
ШИМ управления светодиодной лентой.
- #define `ARCH_LEDLINE_TIMER` "ledline-timer"
Таймер управления светодиодной лентой.

Названия плат

Зарезервированные строковые константы названий известных плат.

- #define `ARCH_BOARD_SE8350_00` "SE8350-00"
- #define `ARCH_BOARD_SE8351_00` "SE8351-00"
- #define `ARCH_BOARD_SE_DB_A20_B254` "SE-DB-A20-B254"

Названия процессоров

Зарезервированные строковые константы названий используемых процессоров.

- #define `ARCH_PROC_A20` "A20"
- #define `ARCH_PROC_A40` "A40"
- #define `ARCH_PROC_H3` "H3"
- #define `ARCH_PROC_V3S` "V3s"

19.3.1. Подробное описание

Подключение:

```
#include <arch/archdef.h>
```

См. также

Подробное описание в разделе *Конфигурация аппаратной части*.

19.3.2. Макросы

19.3.2.1. ARCH_BACKLIGHT_FREQUENCY

```
#define ARCH_BACKLIGHT_FREQUENCY "backlight-freq"
```

Число – рекомендованная частота ШИМ сигнала в герцах, используемого для управления подсветкой дисплея. Может варьироваться от установленного на плате аппаратного драйвера подсветки.

19.3.2.2. ARCH_BACKLIGHT_GPIO

```
#define ARCH_BACKLIGHT_GPIO "backlight-pwm"
```

Число – номер канала ШИМ, используемый для управления подсветкой дисплея.

19.3.2.3. ARCH_BOARD

```
#define ARCH_BOARD "board-name"
```

Наименование целевой платы – строковый параметр, содержащий значение из группы *макросов* известных названий плат. Для перечисленных плат большинство параметров конфигурации будет загружено при инициализации операционной системы.

19.3.2.4. ARCH_BOARD_SE8350_00

```
#define ARCH_BOARD_SE8350_00 "SE8350-00"
```

Плата светодиодной подсветки проекта "Cyclops".

19.3.2.5. ARCH_BOARD_SE8351_00

```
#define ARCH_BOARD_SE8351_00 "SE8351-00"
```

Плата проекта "Стериус".

19.3.2.6. ARCH_BOARD_SE_DB_A20_B254

```
#define ARCH_BOARD_SE_DB_A20_B254 "SE-DB-A20-B254"
```

Отладочная плата модуля SE-SOM-A20.

19.3.2.7. ARCH_CPU

```
#define ARCH_CPU "cpu-name"
```

Название процессора – строковый параметр, содержащий значение из группы *макросов* имён процессоров.

19.3.2.8. ARCH_LED_DISK

```
#define ARCH_LED_DISK "led-disk"
```

Имя линии **GPIO** подключенной к светодиоду индикации работы файловой системы. Имя линии может быть записано без пробелов, как в документации на процессор (**PG1**), либо в стиле *MULTEX-ARM (P_G + 1)*.



Если данного светодиода нет на используемой плате, его можно указать с пустой строкой в качестве значения.

19.3.2.9. ARCH_LED_SYSTEM

```
#define ARCH_LED_SYSTEM "led-system"
```

Имя линии **GPIO** подключенной к светодиоду индикации работы системного таймера. Имя линии может быть записано без пробелов, как в документации на процессор (**PG1**), либо в стиле **MULTEX-ARM (P_G + 1)**.



Если данного светодиода нет на используемой плате, его можно указать с пустой строкой в качестве значения.

19.3.2.10. ARCH_LEDLINE_PWM

```
#define ARCH_LEDLINE_PWM "ledline-pwm"
```

Число – номер канала ШИМ, используемый для управления светодиодной подсветкой. Для управления используется однопроводная шина данных.

19.3.2.11. ARCH_LEDLINE_TIMER

```
#define ARCH_LEDLINE_TIMER "ledline-timer"
```

Число – номер таймера, используемый для управления светодиодной подсветкой. Для управления используется однопроводная шина данных.

19.3.2.12. ARCH_PROC_A20

```
#define ARCH_PROC_A20 "A20"
```

Процессор *Allwinner A20*.

19.3.2.13. ARCH_PROC_A40

```
#define ARCH_PROC_A40 "A40"
```

Процессор *Allwinner A40*.

19.3.2.14. ARCH_PROC_H3

```
#define ARCH_PROC_H3 "H3"
```

Процессор *Allwinner H3*.

19.3.2.15. ARCH_PROC_V3S

```
#define ARCH_PROC_V3S "V3s"
```

Процессор *Allwinner V3s*.

19.4. Файл `assert.h`

Механизмы диагностики и проверки.

Макросы

- `#define __ASSERT_NOOP ((void) 0)`
- `#define assert(expr) ((expr) ? __ASSERT_NOOP : __assert_fail (#expr, __FILE__, __LINE__, __func__))`
- `#define static_assert(x...)`

Функции

- `void __assert_fail (__const char *__assertion, __const char *__file, unsigned int __line, __const char *__function)`

19.4.1. Подробное описание

См. стандарт C11 7.2.

См. также

[C11 standard 7.2.](#)

19.4.2. Макросы

19.4.2.1. `__ASSERT_NOOP`

```
#define __ASSERT_NOOP ((void) 0)
```

Макрос-заглушка.

19.4.2.2. `assert`

```
#define assert(  
    expr) ((expr) ? __ASSERT_NOOP : __assert_fail (#expr, __FILE__, __LINE__, __func__))
```

Механизм проверки. В случае ложности утверждения `x` процесс исполнения будет прерван, а в `stderr` будет распечатано сообщение о сработавшей диагностике.

Аргументы

`expr` Утверждение для проверки.

19.4.2.3. `static_assert`

```
#define static_assert(  
    x... )
```

19.4.3. Функции

19.4.3.1. `__assert_fail()`

```
void __assert_fail (
    __const char * __assertion,
    __const char * __file,
    unsigned int __line,
    __const char * __function )
```

Обработать сработавший ассерт.

Аргументы	
<code>__assertion</code>	Текст ассерта.
<code>__file</code>	Имя файла с ассертом.
<code>__line</code>	Номер строки с ассертом.
<code>__function</code>	Имя функции с ассертом.

19.5. Файл avi.dox

19.6. Файл avilib.h

Работа с AVI файлами.

Определения типов

- typedef void * **AVI_HANDLE**
Дескриптор AVI-файла.

Функции

- int **closeAVIFile** (**AVI_HANDLE** handle)
Корректировка заголовков, запись индексов и закрытие AVI-файла.
- **AVI_HANDLE newAVIFile** (const char *name, int width, int height)
Создание пустого AVI-файла без звука.
- **AVI_HANDLE newAVIFileSnd** (char *name, int width, int height)
Создание пустого AVI-файла со звуком.
- **AVI_HANDLE openAVIFile** (const char *name, int *width, int *height, int *frames)
Открытие файла AVI на чтение и получение размеров.
- int **readAVIAudio** (**AVI_HANDLE** handle, char *buf)
Чтение звука из AVI-файла.
- int **readAVIFrame** (**AVI_HANDLE** handle, char *buf)
Чтение фрейма из AVI-файла.
- int **seekToFirstVideoFrame** (**AVI_HANDLE** handle)
Установить указатель в начало файла.
- int **setAVIType** (int T)
Установить формат записи.
- int **writeAVIFrame** (**AVI_HANDLE** handle, int flags, char *buffer, int size)
Запись фрейма в AVI-файл.
- int **writeSNDFrame** (**AVI_HANDLE** ah, char *buffer, int size)
Запись звукового фрагмента в AVI-файл.

Значения, возвращаемые readAVIFrame()

- #define **AVI_RETURN_EOF** (-1)
- #define **AVI_RETURN_ERROR** (-2)

19.6.1. Подробное описание

Библиотека для работы с файлами-контейнерами стандарта **AVI** (*Audio Video Interleave*).

Подключение:

```
#include <multimedia/a20graph.h>
#include <multimedia/avilib.h>
#include <multimedia/mpeg4codec.h>
```

Makefile:

```
LIBRARIES += -l_a20graph -l_avi -l_mpeg4decode
```

См. также

Общее описание работы с **AVI**-файлами в главе *Работа с AVI-файлами*.

19.6.2. Макросы

19.6.2.1. AVI_RETURN_EOF

```
#define AVI_RETURN_EOF (-1)
```

Конец файла.

19.6.2.2. AVI_RETURN_ERROR

```
#define AVI_RETURN_ERROR (-2)
```

Ошибка чтения или некорректный контейнер.

19.6.3. Типы

19.6.3.1. AVI_HANDLE

```
typedef void* AVI_HANDLE
```

19.6.4. Функции

19.6.4.1. closeAVIFile()

```
int closeAVIFile (  
    AVI_HANDLE handle )
```

Функция корректирует заголовок, записывает индексы и закрывает **AVI**-файл. Если **AVI**-файл был открыт на чтение, то просто закрывает его.

Аргументы

handle Дескриптор файла.

Возвращает

OK при удачном завершении либо код ошибки.

19.6.4.2. newAVIFile()

```
AVI_HANDLE newAVIFile (  
    const char * name,  
    int width,  
    int height )
```

Функция создает пустой **AVI**-файл с заданными параметрами (для записи без звука).

Аргументы

<i>name</i>	Имя файла, под которым он будет создан.
<i>width,height</i>	Размеры кадра.

Возвращает

Дескриптор создаваемого **AVI**-файла *AVI_HANDLE*.

19.6.4.3. newAVIFileSnd()

```
AVI_HANDLE newAVIFileSnd (  
    char * name,  
    int width,  
    int height )
```

Функция создает пустой **AVI**-файл содержащий звуковые данные.

Аргументы

<i>name</i>	Имя файла, под которым он будет создан.
<i>width,height</i>	Размеры кадра.

Возвращает

Дескриптор создаваемого **AVI**-файла *AVI_HANDLE*.

19.6.4.4. openAVIFile()

```
AVI_HANDLE openAVIFile (  
    const char * name,  
    int * width,  
    int * height,  
    int * frames )
```

Функция открывает на чтение имеющийся **AVI**-файл и получает размеры кадра.

Аргументы

<i>name</i>	Имя файла, под которым он будет создан.
<i>width,height</i>	Указатели на переменные, в которых поместятся размеры кадра.
<i>frames</i>	Указатель на количество кадров в файле.

Возвращает

Дескриптор открытого **AVI**-файла.

19.6.4.5. readAVIAudio()

```
int readAVIAudio (  
    AVI_HANDLE handle,  
    char * buf )
```

Функция читает из **AVI**-файла очередной звуковой буфер в память.

Аргументы

handle Дескриптор.

buf Указатель на буфер в памяти.

Возвращает

Размер считанного буфера в байтах.

19.6.4.6. readAVIFrame()

```
int readAVIFrame (  
    AVI_HANDLE handle,  
    char * buf )
```

Функция читает из **AVI**-файла очередной кадр в память.

Аргументы

handle Дескриптор.

buf Указатель на буфер в памяти.

Возвращает

Размер считанного буфера в байтах либо одно из значений, описанных *макросами*.

19.6.4.7. seekToFirstVideoFrame()

```
int seekToFirstVideoFrame (  
    AVI_HANDLE handle )
```

Функция устанавливает указатель на первый кадр в **AVI**-файле.

Аргументы

handle Дескриптор.

Возвращает

OK при удачном завершении либо код ошибки.

19.6.4.8. setAVIType()

```
int setAVIType (  
    int T )
```

Функция устанавливает формат записи для видео.

Аргументы

T Тип записи: **0** – MPEG4, **1** – H.264.

Возвращает

OK при удачном завершении.

ERROR при ошибке.

19.6.4.9. writeAVIFrame()

```
int writeAVIFrame (  
    AVI_HANDLE handle,  
    int flags,  
    char * buffer,  
    int size )
```

Функция записывает фрейм в файл **AVI**.

Аргументы

handle Дескриптор файла.

flags Флаги для кадра — ключевой/промежуточный(не ключевой).

buffer Указатель на данные.

size Размер данных в байтах.

Возвращает

OK при удачном завершении либо код ошибки.

19.6.4.10. writeSNDFrame()

```
int writeSNDFrame (  
    AVI_HANDLE ah,  
    char * buffer,  
    int size )
```

Функция записывает звуковой фрагмент в файл **AVI**, создавать файл нужно функцией *newAVIFileSnd()*.

Аргументы

<i>ah</i>	Дескриптор файла.
<i>buffer</i>	Указатель на данные.
<i>size</i>	Размер данных в байтах.

Возвращает

OK при удачном завершении либо код ошибки.

19.7. Файл blkcache.h

Кэширование записи / чтения блоков данных.

Структуры данных

- struct *blk_cache*

Определения типов

- typedef int(* *blk_cache_proc*) (void *hDrv, int start, int num, char *buf)

Функции

- void *blk_cache_callback* (struct *blk_cache* *sc, void *drv, *blk_cache_proc* read, *blk_cache_proc* write)
Задать процедуру загрузки / выгрузки.
- struct *blk_cache* * *blk_cache_create* (int CacheSize, int BlkSize)
Создать КЭШ заданного размера.
- int *blk_cache_flush* (struct *blk_cache* *sc)
Выгрузить КЭШ.
- void *blk_cache_free* (struct *blk_cache* *sc)
Удалить КЭШ.
- int *blk_cache_load* (struct *blk_cache* *sc, int startBlk)
Загрузить КЭШ.
- int *blk_cache_read* (struct *blk_cache* *sc, int startBlk, int nBlock, char *pBuf)
Считать сектора через КЭШ.
- int *blk_cache_write* (struct *blk_cache* *sc, int startBlk, int nBlock, char *pBuf)
Записать сектора через КЭШ.

19.7.1. Типы

19.7.1.1. blk_cache_proc

```
typedef int(* blk_cache_proc) (void *hDrv, int start, int num, char *buf)
```

Тип указателя на процедуру чтения/записи драйвера.

19.7.2. Функции

19.7.2.1. blk_cache_callback()

```
void blk_cache_callback (
    struct blk_cache * sc,
    void * drv,
    blk_cache_proc read,
    blk_cache_proc write )
```

Функция задаёт процедуры загрузки / выгрузки КЭШ.

Аргументы

sc	Указатель на КЭШ.
----	-------------------

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>drv</i>	Указатель на драйвер.
<i>read</i>	Указатель на процедуру чтения драйвера.
<i>write</i>	Указатель на процедуру записи драйвера.

19.7.2.2. blk_cache_create()

```
struct blk_cache* blk_cache_create (  
    int CacheSize,  
    int BlkSize )
```

Функция создания КЭШ заданного размера.

Аргументы

<i>CacheSize</i>	Заданный размер КЭШ-памяти в блоках.
<i>BlkSize</i>	Размер блока памяти.

Возвращает

Указатель на созданный КЭШ.

19.7.2.3. blk_cache_flush()

```
int blk_cache_flush (  
    struct blk_cache * sc )
```

Выгрузить КЭШ.

Аргументы

<i>sc</i>	Указатель на КЭШ.
-----------	-------------------

Возвращает

OK при успехе.
ERROR при неудаче.

19.7.2.4. blk_cache_free()

```
void blk_cache_free (  
    struct blk_cache * sc )
```

Функция удаления КЭШ.

Аргументы

sc Указатель на КЭШ.

19.7.2.5. blk_cache_load()

```
int blk_cache_load (  
    struct blk_cache * sc,  
    int startBlk )
```

Функция загрузки КЭШ.

Аргументы

sc Указатель на КЭШ.

startBlk Номер начального сектора.

Возвращает

OK при успехе.

ERROR при неудаче.

19.7.2.6. blk_cache_read()

```
int blk_cache_read (  
    struct blk_cache * sc,  
    int startBlk,  
    int nBlock,  
    char * pBuf )
```

Функция считывает через КЭШ **nBlock** секторов начиная с сектора **startBlk** в буфер **pBuf**.

Аргументы

sc Указатель на КЭШ.

startBlk Номер начального сектора.

nBlock Количество считываемых секторов.

pBuf Указатель на массив-приёмник данных.

Возвращает

OK при успехе.

ERROR при неудаче.

19.7.2.7. blk_cache_write()

```
int blk_cache_write (  
    struct blk_cache * sc,  
    int startBlk,  
    int nBlock,  
    char * pBuf)
```

Функция записывает через КЭШ **nBlock** секторов начиная с сектора **startBlk** из буфера **pBuf**.

Аргументы

<i>sc</i>	Указатель на КЭШ.
<i>startBlk</i>	Номер начального сектора.
<i>nBlock</i>	Количество считываемых секторов.
<i>pBuf</i>	Указатель на массив-источник данных.

Возвращает

OK при успехе.

ERROR при неудаче.

19.8. Файл `cache.h`

Методы работы с КЭШ-памятью.

Функции

- void `dcache_disable` (void)
Отключить кэш данных.
- void `dcache_enable` (void)
Включить кэш данных.
- int `dcache_status` (void)
Статус кэша данных.
- void `flush_dcache_all` (void)
Сбросить кэш данных.
- void `flush_dcache_range` (unsigned long start, unsigned long stop)
Сбросить в кэш указанный блок памяти.
- int `icache_status` (void)
Статус кэша команд.
- void `invalidate_dcache_all` (void)
Аннулировать кэш данных.
- void `invalidate_dcache_range` (unsigned long start, unsigned long stop)
Аннулировать кэш в области памяти.
- void `invalidate_icache_all` (void)
Аннулировать весь кэш команд.

19.8.1. Подробное описание

Файл содержит объявления функций работы с КЭШ-памятью процессора.

См. также

Общее описание см. в главе *Работа с встроенной КЭШ-памятью процессора*.

19.8.2. Функции

19.8.2.1. `dcache_disable()`

```
void dcache_disable (  
    void )
```

Функция отключает кэширование данных. Возможно использовать в ходе выполнения программы.

19.8.2.2. `dcache_enable()`

```
void dcache_enable (  
    void )
```

Функция включает кэширование данных. Возможно использовать в ходе выполнения программы.

19.8.2.3. `dcache_status()`

```
int dcache_status (  
    void )
```

Проверка состояния кэша данных.

Возвращает

0 - кэш отключен, **иначе** - кэш включен.

19.8.2.4. flush_dcache_all()

```
void flush_dcache_all (  
    void )
```

Функция сбрасывает все данные из кэш в память.

19.8.2.5. flush_dcache_range()

```
void flush_dcache_range (  
    unsigned long start,  
    unsigned long stop )
```

Функция сбрасывает в память кэш для блока в указанном месте.

Аргументы

start Адрес начала блока памяти данных.

stop Адрес конца блока памяти данных.

19.8.2.6. icache_status()

```
int icache_status (  
    void )
```

Проверка состояния кэша команд.

Возвращает

0 - кэш отключен, **иначе** - кэш включен.

19.8.2.7. invalidate_dcache_all()

```
void invalidate_dcache_all (  
    void )
```

Функция аннулирует весь кэш данных.

19.8.2.8. invalidate_dcache_range()

```
void invalidate_dcache_range (  
    unsigned long start,  
    unsigned long stop )
```

Функция аннулирует кэш для блока в указанном месте памяти.

Аргументы

start Адрес начала блока памяти данных.

stop Адрес конца блока памяти данных.

19.8.2.9. invalidate_icache_all()

```
void invalidate_icache_all (  
    void )
```

Функция аннулирует весь кэш команд.

19.9. Файл cedrus.h

Работа с кодером/декодером видео h.264 CEDRUS.

Функции

- void `ve_close` (void)
Закреть кодер.
- int `ve_open` (void)
Открыть кодер.

Кодирование видео потока

- enum `color_format` { `H264_FMT_NV12` = 0 , `H264_FMT_NV16` = 1 }
- unsigned int `h264_encode` (char *bitstream)
Кодировать один кадр.
- int `h264_encoder_free` (void)
Освободить ресурсы кодера.
- char * `h264_encoder_init` (int width, int height, int qp, int key_interval)
Задать разрешение кодера.
- int `h264_encoder_set_qp` (int qp)
Изменить качество сжатия видео.
- bool `h264_is_keyframe` (void)
Определение ключевых кадров.
- void `h264_set_src_format` (int format)
Задать формат кодирования.

Декодирование видео потока

- int `h264_decode` (int dd, char *bitstream, unsigned int len, char *output)
Декодировать один кадр.
- int `h264_decode_part` (int dd, char *bitstream, unsigned int len, char *output, int x, int y, int w, int h)
Декодировать часть кадра.
- int `h264_decoder_free` (int dd)
Освободить ресурсы декодера.
- int `h264_decoder_init` (int width, int height, void(*wait_retrace)(void))
Инициализация декодера.
- char * `h264_get_out` (int dd)
Кадр декодера.

Настройки декодера.

Эти функции должны быть вызваны до инициализации декодера.

- void `h264_decoder_set_mk` (int use)
Использовать второе ядро для пересчета выходного буфера.
- int `h264_decoder_set_qp` (int qp)
Задать качество декодирования.
- void `h264_decoder_set_wm` (int no_wait)
Взаимодействие 2-х ядер.

19.9.1. Подробное описание

Библиотека аппаратного кодирования/декодирования видеопотоков в стандарте h.264.

См. также

Общее описание работы с кодером/декодером видео в главе
[Кодер/декодер видео h.264 CEDRUS.](#)

19.9.2. Перечисления

19.9.2.1. color_format

enum *color_format*

Элементы перечислений

H264_FMT_NV12

H264_FMT_NV16

```
00069          { H264_FMT_NV12 = 0,
00070              H264_FMT_NV16 = 1 };
```

19.9.3. Функции

19.9.3.1. h264_decode()

```
int h264_decode (
    int dd,
    char * bitstream,
    unsigned int len,
    char * output )
```

Функция декодирует один кадр. Если задан указатель на выходное изображение **output**, то декодированный кадр будет целиком переведён из формата **Tiled YUV** в **Planar**.

Аргументы

<i>dd</i>	Указатель на экземпляр декодера.
<i>bitstream</i>	Указатель на входной кодированный поток.
<i>len</i>	Длина входного потока.
<i>output</i>	Указатель на выходное изображение, либо 0 .

Возвращает

OK при удачном завершении, иначе **ERROR**.

19.9.3.2. h264_decode_part()

```
int h264_decode_part (
```

```

int dd,
char * bitstream,
unsigned int len,
char * output,
int x,
int y,
int w,
int h )

```

Функция декодирует один кадр и переводит часть декодированного кадра из формата **Tiled YUV** в формат **Planar** часть кадра, заданную координатами. Координаты кадра должны быть кратны размеру *плитки* - 32 пикселя. Функция является обёрткой для *h264_decode()* и имеет смысл только если задан указатель на выходное изображение **output**.

Аргументы	
<i>dd</i>	Указатель на экземпляр декодера.
<i>bitstream</i>	Указатель на входной кодированный поток.
<i>len</i>	Длина входного потока.
<i>output</i>	Указатель на выходное изображение, либо 0 .
<i>x,y</i>	Координаты верхнего левого угла декодируемого кадра.
<i>w,h</i>	Размеры декодируемого кадра.

Возвращает

OK при удачном завершении, иначе **ERROR**.

19.9.3.3. h264_decoder_free()

```

int h264_decoder_free (
    int dd )

```

Функция освобождает ресурсы, занятые декодером.

Аргументы	
<i>dd</i>	Указатель на экземпляр декодера.

Возвращает

OK при удачном завершении.
ERROR при ошибке.

19.9.3.4. h264_decoder_init()

```

int h264_decoder_init (

```

```
int width,  
int height,  
void*(void) wait_retrace )
```

Функция инициализирует декодер.

Аргументы

<i>width,height</i>	Оригинальные размеры выходного кадра в пикселях.
<i>wait_retrace</i>	Указатель на функцию ожидания обратного хода или <i>NULL</i> , если не нужна.

Возвращает

Указатель на экземпляр декодера.

19.9.3.5. h264_decoder_set_mk()

```
void h264_decoder_set_mk (  
    int use )
```

Функция устанавливает двухядерный режим обработки декодера.

Аргументы

<i>use</i>	1 — использовать 2-е ядро, 0 — не использовать.
------------	---

19.9.3.6. h264_decoder_set_qp()

```
int h264_decoder_set_qp (  
    int qp )
```

Функция устанавливает начальное значение параметра качества.

Аргументы

<i>qp</i>	Начальный параметр качества.
-----------	------------------------------

Возвращает

OK при удачном завершении.
ERROR при ошибке.

19.9.3.7. h264_decoder_set_wm()

```
void h264_decoder_set_wm (  
    int wm )
```

```
int no_wait )
```

Функция указывает (в двухядерном режиме) ждать ли первому ядру завершения пересчета выходного буфера вторым ядром.

Аргументы

no_wait **1** — не ждать, **0** — ждать.

19.9.3.8. h264_encode()

```
unsigned int h264_encode (  
    char * bitstream )
```

Функция кодирует один кадр исходного изображения.

Аргументы

bitstream Указатель на выходной поток кодера.

Возвращает

Длина в байтах закодированного в поток видео.

19.9.3.9. h264_encoder_free()

```
int h264_encoder_free (  
    void )
```

Функция освобождает занятые кодером ресурсы.

Возвращает

OK при удачном завершении.
ERROR если кодер не был создан.

19.9.3.10. h264_encoder_init()

```
char* h264_encoder_init (  
    int width,  
    int height,  
    int qp,  
    int key_interval )
```

Функция подготавливает кодер для указанного разрешения.

Аргументы

<i>width,height</i>	Размеры кадра для кодирования в пикселях.
<i>qp</i>	Качество сжатия видео (0 – мин. 47 – макс.)
<i>key_interval</i>	Период следования ключевых кадров (в кадрах).

Возвращает

Указатель на входной буфер, в котором надо разместить исходное изображение.

19.9.3.11. `h264_encoder_set_qp()`

```
int h264_encoder_set_qp (  
    int qp )
```

Функция изменяет текущее качество сжатия видео при кодировании видео.

Аргументы

<i>qp</i>	Качество сжатия видео (1 – мин. 47 – макс.)
-----------	---

Возвращает

OK при удачном изменении качества сжатия, иначе *ERROR*.

19.9.3.12. `h264_get_out()`

```
char* h264_get_out (  
    int dd )
```

Функция возвращает указатель на декодированный кадр.

Аргументы

<i>dd</i>	Указатель на экземпляр декодера.
-----------	----------------------------------

Возвращает

*char** Указатель на декодированный кадр.

19.9.3.13. `h264_is_keyframe()`

```
bool h264_is_keyframe (
```

```
void )
```

Функция сообщает, был ли последний кодированный кадр ключевым.

Возвращает

- 1 для ключевого кадра.
- Ненулевое значение для всех прочих кадров.

19.9.3.14. h264_set_src_format()

```
void h264_set_src_format (  
    int format )
```

Функция устанавливает формат видео для кодирования (**NV12** или **NV16**).

Аргументы

format Один из форматов перечисленных в *color_format*.

19.9.3.15. ve_close()

```
void ve_close (  
    void )
```

Функция закрывает (деактивирует) систему аппаратного кодирования/декодирования.

19.9.3.16. ve_open()

```
int ve_open (  
    void )
```

Функция открывает систему аппаратного кодирования/декодирования.

Возвращает

- OK при успешном завершении или код ошибки.

19.10. Файл `console.h`

Дополнительные функции для работы с текстовым вводом-выводом.

Функции

- char `get_c` (void)
- unsigned int `geth` (void)
- void `putb` (unsigned char b)
- void `puti` (int N)
- void `putl` (unsigned int l)
- void `putw` (unsigned short w)
- unsigned int `strtoh` (unsigned char *str)

19.10.1. Подробное описание

Некоторые функции, дополняющие стандартный функционал `stdio.h`.

19.10.2. Функции

19.10.2.1. `get_c()`

```
char get_c (
    void )
```

Получить следующий символ из uart-консоли.

Возвращает

Символ.

Функция блокирует исполнение до тех пор, пока не получит символ.



Символ берется из uart-консоли, а не из stdin.

19.10.2.2. `geth()`

```
unsigned int geth (
    void )
```

Считать из uart-консоли беззнаковое число в шестнадцатеричном формате.

Число будет считываться пока не будет получен не-шестнадцатеричный символ.

Возвращает

Считанное число.

19.10.2.3. putb()

```
void putb (
    unsigned char b )
```

Вывести беззнаковый байт в stdout в шестнадцатеричном формате.

Аргументы

b Беззнаковый байт, который будет выведен.

19.10.2.4. puti()

```
void puti (
    int N )
```

Распечатать целое число в stdout в десятичном формате.

Аргументы

N Целое число, которое будет выведено.

19.10.2.5. putl()

```
void putl (
    unsigned int l )
```

Вывести беззнаковое 32-битное число в stdout в шестнадцатеричном формате.

Аргументы

l Беззнаковое 32-битное число, которое будет выведено.

19.10.2.6. putw()

```
void putw (
    unsigned short w )
```

Вывести беззнаковое 16-битное число в stdout в шестнадцатеричном формате.

Аргументы

w Беззнаковое 16-битное число, которое будет выведено.

19.10.2.7. strtouh()

```
unsigned int strtouh (  
    unsigned char * str )
```

Считать из строки беззнаковое число в шестнадцатеричном формате.

Число будет считываться пока не будет получен не-шестнадцатеричный символ.

Аргументы

str Указатель на строку.

Возвращает

Считанное число или 0, в случае ошибки.

19.11. Файл `crc32.h`

Имплементация `crc32` из GCC.

Функции

- `uint32_t crc32_compute` (`const void *pData`, `size_t len`, `uint32_t init`)

19.11.1. Функции

19.11.1.1. `crc32_compute()`

```
uint32_t crc32_compute (  
    const void * pData,  
    size_t len,  
    uint32_t init )
```

Имплементация `crc32` из GCC, полином - 0x04c11db7.

Аргументы

`pData` Указатель на данные.

`len` Длина данных.

`init` Параметр, обычно 0xffffffff.

Возвращает

Контрольная сумма.

19.12. Файл `crc8.h`

Чек-сумма `crc8`.

Функции

- `uint8_t crc8_1step (uint8_t byte, uint8_t crc8)`
- `uint8_t crc8_compute (const void *pData, uint8_t size)`

19.12.1. Функции

19.12.1.1. `crc8_1step()`

```
uint8_t crc8_1step (  
    uint8_t byte,  
    uint8_t crc8 )
```

Получить промежуточный результат вычисления `crc8`.

Аргументы

`byte` Байт, который нужно добавить к чек-сумме.

`crc8` Результат вычислений для предыдущего байта или `0xFF` для инициализации.

Возвращает

Промежуточный результат вычисления `crc8`.

19.12.1.2. `crc8_compute()`

```
uint8_t crc8_compute (  
    const void * pData,  
    uint8_t size )
```

Расчитать чек-сумму `crc8`.

Аргументы

`pData` Указатель на данные, для которых ведется расчет чек-суммы.

`size` Размер данных.



Данные должны иметь размер не более 255 байт.

Возвращает

Чек-сумма `сгс8`.

19.13. Файл crt.h

Функции для работы с терминалом и клавиатурой, а также заглушки для обратной совместимости.

Коды цветов.

- #define *bgBrightBlack* 100
- #define *bgBrightBlue* 104
- #define *bgBrightCyan* 106
- #define *bgBrightGreen* 102
- #define *bgBrightMagenta* 105
- #define *bgBrightRed* 101
- #define *bgBrightWhite* 107
- #define *bgBrightYellow* 103
- #define *clBlack* 30
- #define *clBlue* 34
- #define *clBrightBlack* 90
- #define *clBrightBlue* 94
- #define *clBrightCyan* 96
- #define *clBrightGreen* 92
- #define *clBrightMagenta* 95
- #define *clBrightRed* 91
- #define *clBrightWhite* 97
- #define *clBrightYellow* 93
- #define *clCyan* 36
- #define *clGreen* 32
- #define *clMagenta* 35
- #define *clNone* 0
- #define *clRed* 31
- #define *clWhite* 37
- #define *clYellow* 33

Атрибуты текста.

- #define *taBgLight* 2
- #define *taInverse* 8
- #define *taLight* 1
- #define *taNormal* 0
- #define *taUnderlined* 4

Функции и макросы для работы с терминалом.



Терминал должен быть подключен к **stdout**.

- void *clrscr* (void)
- void *CSI* (const char *s)
- void *cursorMove* (unsigned char row, unsigned char column)
Установить курсор в терминале на заданную позицию.
- void *cursorOff* (void)
- void *cursorOn* (void)
- void *cursorRestore* (void)
- void *cursorStore* (void)
- void *textAttr* (char TA)
- void *textBackground* (char C)
- void *textColor* (char C)

Функции для работы с клавиатурой.

- int *keyPressed* (void)
- char *readKey* (void)
- char *readKey_Timeout* (int ticks)

Заглушки функций для работы с пищалкой.

- int *nosound* (void)
- int *sound* (int F)
- int *soundt* (int F, int t)

19.13.1. Макросы**19.13.1.1. bgBrightBlack**

```
#define bgBrightBlack 100
```

Серый фон.

19.13.1.2. bgBrightBlue

```
#define bgBrightBlue 104
```

Светло-синий фон.

19.13.1.3. bgBrightCyan

```
#define bgBrightCyan 106
```

Светлый циан (фон).

19.13.1.4. bgBrightGreen

```
#define bgBrightGreen 102
```

Светло-зелёный фон.

19.13.1.5. bgBrightMagenta

```
#define bgBrightMagenta 105
```

Светлый маджента (фон).

19.13.1.6. bgBrightRed

```
#define bgBrightRed 101
```

Светло-красный фон.

19.13.1.7. bgBrightWhite

```
#define bgBrightWhite 107
```

Яркий белый фон.

19.13.1.8. bgBrightYellow

```
#define bgBrightYellow 103
```

Светло-жёлтый фон.

19.13.1.9. clBlack

```
#define clBlack 30
```

Чёрный.

19.13.1.10. clBlue

```
#define clBlue 34
```

Синий.

19.13.1.11. clBrightBlack

```
#define clBrightBlack 90
```

Серый.

19.13.1.12. clBrightBlue

```
#define clBrightBlue 94
```

Светло-синий.

19.13.1.13. clBrightCyan

```
#define clBrightCyan 96
```

Светлый циан.

19.13.1.14. clBrightGreen

```
#define clBrightGreen 92
```

Светло-зелёный.

19.13.1.15. clBrightMagenta

```
#define clBrightMagenta 95
```

Светлый маджента.

19.13.1.16. clBrightRed

```
#define clBrightRed 91
```

Светло-красный.

19.13.1.17. clBrightWhite

```
#define clBrightWhite 97
```

Яркий белый.

19.13.1.18. clBrightYellow

```
#define clBrightYellow 93
```

Светло-жёлтый.

19.13.1.19. clCyan

```
#define clCyan 36
```

Циан.

19.13.1.20. clGreen

```
#define clGreen 32
```

Зеленый.

19.13.1.21. clMagenta

```
#define clMagenta 35
```

Маджента.

19.13.1.22. clNone

```
#define clNone 0
```

Синий.

19.13.1.23. clRed

```
#define clRed 31
```

Красный.

19.13.1.24. clWhite

```
#define clWhite 37
```

Белый.

19.13.1.25. clYellow

```
#define clYellow 33
```

Желтый.

19.13.1.26. taBgLight

```
#define taBgLight 2
```

Увеличение яркости фона.

19.13.1.27. taInverse

```
#define taInverse 8
```

Инвертирование цветов текста.

19.13.1.28. taLight

```
#define taLight 1
```

Увеличение яркости текста.

19.13.1.29. taNormal

```
#define taNormal 0
```

Выключение всех атрибутов.

19.13.1.30. taUnderlined

```
#define taUnderlined 4
```

Подчеркивание текста

19.13.2. Функции**19.13.2.1. clrscr()**

```
void clrscr (  
            void )
```

Очистить экран терминала.

19.13.2.2. CSI()

```
void CSI (  
         const char * s )
```

Отправить ESC-последовательность.

CSI будет отправлен перед управляющей строкой.

Аргументы	
s	Управляющая строка (без CSI).

19.13.2.3. cursorMove()

```
void cursorMove (
    unsigned char row,
    unsigned char column )
```

Устанавливает курсор в заданную строку на заданный символ. Начало отсчёта (позиция 1:1) — левый верхний угол.

Аргументы

<i>row</i>	Номер строки.
<i>column</i>	Номер символа (колонки).

19.13.2.4. cursorOff()

```
void cursorOff (
    void )
```

Скрыть курсор.

19.13.2.5. cursorOn()

```
void cursorOn (
    void )
```

Сделать курсор видимым.

19.13.2.6. cursorRestore()

```
void cursorRestore (
    void )
```

Восстановить положение курсора.

19.13.2.7. cursorStore()

```
void cursorStore (
    void )
```

Сохранить положение курсора.

19.13.2.8. keyPressed()

```
int keyPressed (
    void )
```

Проверить, есть ли зажатая клавиша на клавиатуре.

Возвращает

0, если клавиши или клавиатуры нет, не-0, если есть.

19.13.2.9. nosound()

```
int nosound (
    void )
```

ЗАГЛУШКА: Выключить пищалку.

Возвращает

Всегда возвращает 0.



Это функция-заглушка, она фактически не отработывает.

19.13.2.10. readKey()

```
char readKey (
    void )
```

Вернуть следующую нажатую клавишу.

Возвращает

Следующая нажатая клавиша, либо 0 при отсутствии клавиатуры или ошибке.



Функция блокирует исполнение до тех пор, пока не будет получена клавиша.

19.13.2.11. readKey_Timeout()

```
char readKey_Timeout (
    int ticks )
```

Вернуть следующую нажатую клавишу, при этом ждать не более установленного времени.

Аргументы	
<i>ticks</i>	Количество тиков, которое следует прождать (см.

См. также

[sysClkRateGet\(\)](#), либо *NO_WAIT* для отсутствия ожидания, либо *WAIT_FOREVER* для вечного ожидания.

Возвращает

Следующая нажатая клавиша, либо 0 при отсутствии клавиатуры или ошибке.



Функция блокирует исполнение до тех пор, пока не будет получена клавиша.

19.13.2.12. `sound()`

```
int sound (  
    int F)
```

ЗАГЛУШКА: Включить пищалку.

Аргументы	
<i>F</i>	Частота звука.

Возвращает

Всегда возвращает 0.



Это функция-заглушка, она фактически не обрабатывает.

19.13.2.13. `soundt()`

```
int soundt (  
    int F,  
    int t)
```

ЗАГЛУШКА: Включить пищалку на некоторое время.

Аргументы	
<i>F</i>	Частота звука.
<i>t</i>	Длительность писка в тиках (

См. также

[`sysClkRateGet\(\)`](#).

Возвращает

Всегда возвращает 0.



Это функция-заглушка, она фактически не обрабатывает.

19.13.2.14. textAttr()

```
void textAttr (  
    char TA )
```

Установить атрибут текста.

Аргументы

TA Атрибут текста.

19.13.2.15. textBackground()

```
void textBackground (  
    char C )
```

Установить цвет фона.

Аргументы

C Код цвета.

19.13.2.16. textColor()

```
void textColor (  
    char C )
```

Установить цвет текста.

Аргументы

C Код цвета.

19.14. Файл `csi.h`

Работа с интерфейсом CSI (Camera Sensor Interface)

Номер канала CSI

- `#define CSI_0 0`
Канал CSI 0.
- `#define CSI_1 1`
Канал CSI 1.

Настройка

Устройство CSI создаётся с помощью `csiDevCreate()`. Далее можно настроить созданное устройство с помощью функций из этого раздела.

- `void csiSetMode (void *dev, eCsiInputFormat inputFormat, eCsiOutputFormat outputFormat)`
Выбор форматов входного и выходного сигнала.
- `void csiSetPhase (void *dev, eCsiPolarity vRef, eCsiPolarity hRef, eCsiClockEdge clock)`
Выбор полярности сигнала.
- `void csiSetSeq (void *dev, eSciInputSequence seq)`
Установка последовательности компонент.
- `enum eCsiClockEdge { csiFallingEdge = 0, csiRisingEdge = 1 }`
Выбор активного фронта сигнала.
- `enum eCsiFieldSelection { csiField_odd, csiField_even, csiField_either }`
Выбор захватываемых строк.
- `enum eCsiInputFormat { csiInFmt_RAW = 0b000, csiInFmt_CCIR656_1ch = 0b010, csiInFmt_YUV422 = 0b011, csiInFmt_YUV422_16bit = 0b100, csiInFmt_CCIR656_2ch = 0b101, csiInFmt_CCIR656_4ch = 0b111 }`
Формат входных данных.
- `enum eCsiOutputFormat { csiOutFmt_PathThrough = 0b0000, csiOutFmt_YCbCr_422_fldp = 0b0000, csiOutFmt_YCbCr_420_fldp = 0b0001, csiOutFmt_YCbCr_420_frp = 0b0010, csiOutFmt_YCbCr_422_frp = 0b0011, csiOutFmt_YCbCr_422_fldp_UVc = 0b0100, csiOutFmt_YCbCr_420_fldp_UVc = 0b0101, csiOutFmt_YCbCr_420_frp_UVc = 0b0110, csiOutFmt_YCbCr_422_frp_UVc = 0b0111, csiOutFmt_YCbCr_422_inter = 0b1111, csiOutFmt_YCbCr_422_fldt = 0b1000, csiOutFmt_YCbCr_420_fldt = 0b1001, csiOutFmt_YCbCr_420_frt = 0b1010, csiOutFmt_YCbCr_422_frt = 0b1011, csiOutFmt_YUV_422_p = 0b0000, csiOutFmt_YUV_420_p = 0b0001, csiOutFmt_YUV_422_pc = 0b0100, csiOutFmt_YUV_420_pc = 0b0101, csiOutFmt_YUV_422_t = 0b1000, csiOutFmt_YUV_420_t = 0b1001 }`
Формат выходных данных.
- `enum eCsiPolarity { csiActiveLow = 0, csiActiveHigh = 1 }`
Выбор активного уровня сигнала.
- `enum eSciInputSequence { csiSequence_YUVV = 0, csiSequence_YVYU, csiSequence_UYVY, csiSequence_VYUY, sciSequence_RGRG = 0, sciSequence_GRGR, sciSequence_BGBG, sciSequence_GBGB }`
Последовательность компонент входных данных.

Создание устройства

- `void * csiDevCreate (unsigned int chNum, unsigned int width, unsigned int height)`
Создание устройства CSI для выбранного канала.
- `void csiDevDelete (void *dev)`
Удаление устройства CSI из системы.

Запуск драйвера и получение данных

- void *csiFlush* (void *dev)
Сброс ожидания выбранного устройства.
- void * *csiGetChroma* (void *dev)
Получение указателя на массив цветоразностного сигнала.
- void * *csiGetLuma* (void *dev)
Получение указателя на массив яркостного сигнала.
- STATUS *csiRun* (void *dev)
Запуск выбранного устройства.
- void *csiStop* (void *dev)
Остановка выбранного устройства.
- int *csiWaitFrame* (void *dev)
Ожидание приёма кадра.

19.14.1. Подробное описание

Файл содержит методы работы с аппаратным интерфейсом подключения цифровых камер.

Подключение:

```
#include <multimedia/csi.h>
```

Makefile:

```
LIBRARIES += -l_csi
```

См. также

Общее описание интерфейса в главе [Поддержка CSI \(Camera Sensor Interface\)](#).

19.14.2. Макросы

19.14.2.1. CSI_0

```
#define CSI_0 0
```

19.14.2.2. CSI_1

```
#define CSI_1 1
```

19.14.3. Перечисления

19.14.3.1. eCsiClockEdge

```
enum eCsiClockEdge
```

Элементы перечислений

csiFallingEdge Активный фронт сигнала - убывающий.

csiRisingEdge Активный фронт сигнала - возрастающий.

```
00197            {
00198        csiFallingEdge = 0,
00199        csiRisingEdge  = 1
00200 } eCsiClockEdge;
```

19.14.3.2. eCsiFieldSelection

enum *eCsiFieldSelection*

Выбор строк актуален только для входного формата CCIR656.

Элементы перечислений

csiField_odd Только нечётные строки.

csiField_even Только чётные строки.

csiField_either Как чётные так и нечётные строки.

```
00224            {
00225        csiField_odd,
00226        csiField_even,
00227        csiField_either
00228 } eCsiFieldSelection;
```

19.14.3.3. eCsiInputFormat

enum *eCsiInputFormat*

Формат входных данных модуля CSI.

Элементы перечислений

csiInFmt_RAW RAW stream.

csiInFmt_CCIR656_1ch CCIR656 (1 канал).

csiInFmt_YUV422 YUV422.

Продолжение на следующей странице

Элементы перечислений	
csiInFmt_YUV422_16bit	YUV422 шина данных 16-бит.
csiInFmt_CCIR656_2ch	CCIR656 (2 канала).
csiInFmt_CCIR656_4ch	CCIR656 (4 канала).

```

00136          {
00137      csiInFmt_RAW          = 0b000,
00138      csiInFmt_CCIR656_1ch = 0b010,
00139      csiInFmt_YUV422       = 0b011,
00140      csiInFmt_YUV422_16bit = 0b100,
00141      csiInFmt_CCIR656_2ch = 0b101,
00142      csiInFmt_CCIR656_4ch = 0b111
00143 } eCsiInputFormat;
    
```

19.14.3.4. eCsiOutputFormat

enum *eCsiOutputFormat*

Формат выходных данных зависит от выбранного входного формата. Описание значение взято из документации.

Элементы перечислений	
csiOutFmt_PathThrough	Pass-through for RAW.
csiOutFmt_YCbCr_422_fldp	For CCIR656 input. Field planar YCbCr 422.
csiOutFmt_YCbCr_420_fldp	For CCIR656 input. Field planar YCbCr 420.
csiOutFmt_YCbCr_420_frp	For CCIR656 input. Frame planar YCbCr 420.
csiOutFmt_YCbCr_422_frp	For CCIR656 input. Frame planar YCbCr 422.
csiOutFmt_YCbCr_422_fldp_UVc	For CCIR656 input. Field planar YCbCr 422 UV combined.
csiOutFmt_YCbCr_420_fldp_UVc	For CCIR656 input. Field planar YCbCr 420 UV combined.
csiOutFmt_YCbCr_420_frp_UVc	For CCIR656 input. Frame planar YCbCr 420 UV combined.
csiOutFmt_YCbCr_422_frp_UVc	For CCIR656 input. Frame planar YCbCr 422 UV combined.
csiOutFmt_YCbCr_422_inter	For CCIR656 input. Interlaced interleaved YCbCr422. In this mode, capturing interlaced input and output the interlaced fields from individual ports. Field 1 data will be wrote to FIFO0 output buffer and field 2 data will be wrote to FIFO1 output buffer.
csiOutFmt_YCbCr_422_fldt	For CCIR656 input. Field tiled YCbCr 422.
csiOutFmt_YCbCr_420_fldt	For CCIR656 input. Field tiled YCbCr 420.

Продолжение на следующей странице

Элементы перечислений	
csiOutFmt_YCbCr_420_frt	For CCIR656 input. Frame tiled YCbCr 420.
csiOutFmt_YCbCr_422_frt	For CCIR656 input. Frame tiled YCbCr 422.
csiOutFmt_YUV_422_p	For YUV422 input. Planar YUV 422.
csiOutFmt_YUV_420_p	For YUV422 input. Planar YUV 420.
csiOutFmt_YUV_422_pc	For YUV422 input. Planar YUV 422 UV combined.
csiOutFmt_YUV_420_pc	For YUV422 input. Planar YUV 420 UV combined.
csiOutFmt_YUV_422_t	For YUV422 input. Tiled YUV 422.
csiOutFmt_YUV_420_t	For YUV422 input. Tiled YUV 420.

```

00151         {
00152
00153 // When the input format is set RAW stream
00154     csiOutFmt_PathThrough = 0b0000,
00155
00156 // When the input format is set CCIR656 interface
00157     csiOutFmt_YCbCr_422_fldp = 0b0000,
00158     csiOutFmt_YCbCr_420_fldp = 0b0001,
00159     csiOutFmt_YCbCr_420_frp  = 0b0010,
00160     csiOutFmt_YCbCr_422_frp  = 0b0011,
00161     csiOutFmt_YCbCr_422_fldp_UVc = 0b0100,
00162     csiOutFmt_YCbCr_420_fldp_UVc = 0b0101,
00163     csiOutFmt_YCbCr_420_frp_UVc  = 0b0110,
00164     csiOutFmt_YCbCr_422_frp_UVc  = 0b0111,
00165     csiOutFmt_YCbCr_422_inter = 0b1111,
00166     csiOutFmt_YCbCr_422_fldt = 0b1000,
00167     csiOutFmt_YCbCr_420_fldt = 0b1001,
00168     csiOutFmt_YCbCr_420_frt  = 0b1010,
00169     csiOutFmt_YCbCr_422_frt  = 0b1011,
00170
00171 // When the input format is set YUV422
00172     csiOutFmt_YUV_422_p  = 0b0000,
00173     csiOutFmt_YUV_420_p  = 0b0001,
00174     csiOutFmt_YUV_422_pc = 0b0100,
00175     csiOutFmt_YUV_420_pc = 0b0101,
00176     csiOutFmt_YUV_422_t  = 0b1000,
00177     csiOutFmt_YUV_420_t  = 0b1001
00178 } eCsiOutputFormat;
    
```

19.14.3.5. eCsiPolarity

enum *eCsiPolarity*

Элементы перечислений

csiActiveLow	Активный уровень сигнала - низкий.
csiActiveHigh	Активный уровень сигнала - высокий.

```

00189         {
00190     csiActiveLow = 0,
00191     csiActiveHigh = 1
00192 } eCsiPolarity;
    
```

19.14.3.6. eSciInputSequence

enum *eSciInputSequence*

Элементы перечислений

csiSequence_YUYV	Последовательность компонент входных данных YUYV (только для входного формата YUV422).
csiSequence_YVYU	Последовательность компонент входных данных YVYU (только для входного формата YUV422).
csiSequence_UYVY	Последовательность компонент входных данных UYVY (только для входного формата YUV422).
csiSequence_VYUY	Последовательность компонент входных данных VYUY (только для входного формата YUV422).
sciSequence_RGRG	Входная последовательность RGRG (только для формата Byer).
sciSequence_GRGR	Входная последовательность GRGR (только для формата Byer).
sciSequence_BGBG	Входная последовательность BGBG (только для формата Byer).
sciSequence_GBGB	Входная последовательность GBGB (только для формата Byer).

```

00206         {
00207
00208     csiSequence_YUYV=0,
00209     csiSequence_YVYU,
00210     csiSequence_UYVY,
00211     csiSequence_VYUY,
00212
00213     sciSequence_RGRG=0,
00214     sciSequence_GRGR,
00215     sciSequence_BGBG,
00216     sciSequence_GBGB
00217 } eSciInputSequence;
    
```

19.14.4. Функции

19.14.4.1. `csiDevCreate()`

```
void* csiDevCreate (
    unsigned int chNum,
    unsigned int width,
    unsigned int height )
```

При создании устройства создаются структуры управления аппаратным драйвером и выделяется память под входные данные.

Аргументы

<i>chNum</i>	Номер канал. Рекомендуется брать значение из группы <i>макросов</i> .
<i>width</i>	Ширина принимаемого кадра для создаваемого устройства в пикселях.
<i>height</i>	Высота принимаемого кадра для создаваемого устройства в пикселях.

Возвращает

Идентификатор устройства - указатель на выделенную под устройство память.

19.14.4.2. `csiDevDelete()`

```
void csiDevDelete (
    void * dev )
```

Функция освобождает выделенную под устройство память. Перед удалением следует остановить устройство с помощью `csiStop()`.

Аргументы

<i>dev</i>	Идентификатор устройства, полученный при создании.
------------	--

19.14.4.3. `csiFlush()`

```
void csiFlush (
    void * dev )
```

Функция сбрасывает флаг ожидания кадра выбранного устройства. При этом задача ожидающая очередного кадра на `csiWaitFrame()` получит управление.

Аргументы

<i>dev</i>	Идентификатор устройства, полученный при создании.
------------	--

19.14.4.4. `csiGetChroma()`

```
void* csiGetChroma (  
    void * dev )
```

Функция возвращает указатель на на полученный массив цветоразностного сигнала (**CbCr**).

Аргументы

dev Идентификатор устройства, полученный при создании.

Возвращает

указатель на массив **CbCr**.

19.14.4.5. `csiGetLuma()`

```
void* csiGetLuma (  
    void * dev )
```

Функция возвращает указатель на на полученный массив яркостного сигнала (**Y'**).

Аргументы

dev Идентификатор устройства, полученный при создании.

Возвращает

указатель на массив **Y'**.

19.14.4.6. `csiRun()`

```
STATUS csiRun (  
    void * dev )
```

Запуск драйвера и начало получения данных от аппаратной части. Внутри драйвера ведётся двойная буферизация для каждого устройства.

Аргументы

dev Идентификатор устройства, полученный при создании.

Возвращает

OK при успешном запуске, иначе **ERROR**.

19.14.4.7. `csiSetMode()`

```
void csiSetMode (  
    void * dev,  
    eCsiInputFormat inputFormat,  
    eCsiOutputFormat outputFormat )
```

Функция позволяет указать формат входного сигнала и выбрать формат выходного. Преобразование осуществляется аппаратно.

Аргументы	
<i>dev</i>	Идентификатор устройства, полученный при создании.
<i>inputFormat</i>	Формат входного сигнала.
<i>outputFormat</i>	Формат выходного сигнала.

19.14.4.8. `csiSetPhase()`

```
void csiSetPhase (  
    void * dev,  
    eCsiPolarity vRef,  
    eCsiPolarity hRef,  
    eCsiClockEdge clock )
```

Функция задаёт полярность сигналов синхронизации и активный фронт сигнала CLK.

Аргументы	
<i>dev</i>	Идентификатор устройства, полученный при создании.
<i>vRef</i>	Полярность сигнала кадровой синхронизации.
<i>hRef</i>	Полярность сигнала строчной синхронизации.
<i>clock</i>	Активный фронт сигнала CLK.

19.14.4.9. `csiSetSeq()`

```
void csiSetSeq (  
    void * dev,  
    eSciInputSequence seq )
```

Установка последовательности компонент входных данных.

Продолжение на следующей странице

Аргументы (Продолжение.)

Аргументы

dev Идентификатор устройства, полученный при создании.

seq Задаваемая последовательность.

19.14.4.10. `csiStop()`

```
void csiStop (  
    void * dev )
```

Остановка драйвера для выбранного устройства.

Аргументы

dev Идентификатор устройства, полученный при создании.

19.14.4.11. `csiWaitFrame()`

```
int csiWaitFrame (  
    void * dev )
```

Функция ожидает приёма следующего кадра от выбранного устройства. Во время ожидания другие задачи не блокируются. Выбранной устройством должно быть запущено с помощью `csiRun()`.

Аргументы

dev Идентификатор устройства, полученный при создании.

Возвращает

Всегда *OK*.

19.15. Файл ctype.h

Классификация и преобразование отдельных символов.

Расширения BSD и SVID

- `#define _tolower(c) tolower(c)`
- `#define _toupper(c) toupper(c)`
- `int isascii (int c)`
- `int toascii (int c)`

Макросы для обратной совместимости

- `#define _IS_BLN 128`
- `#define _IS_CTL 32`
- `#define _IS_DIG 2`
- `#define _IS_HEX 16`
- `#define _IS_LOW 8`
- `#define _IS_PUN 64`
- `#define _IS_SP 1`
- `#define _IS_UPP 4`

Стандартные функции

- `int isalnum (int c)`
- `int isalpha (int c)`
- `int isblank (int c)`
- `int iscntrl (int c)`
- `int isdigit (int c)`
- `int isgraph (int c)`
- `int islower (int c)`
- `int isprint (int c)`
- `int ispunct (int c)`
- `int isspace (int c)`
- `int isupper (int c)`
- `int isxdigit (int c)`
- `int tolower (int c)`
- `int toupper (int c)`

19.15.1. Подробное описание

См. также

C11 standard 7.4.

19.15.2. Макросы

19.15.2.1. _IS_BLN

```
#define _IS_BLN 128
```

blank

19.15.2.2. _IS_CTL

```
#define _IS_CTL 32
```

Control

19.15.2.3. _IS_DIG

```
#define _IS_DIG 2  
is digit indicator
```

19.15.2.4. _IS_HEX

```
#define _IS_HEX 16  
[0..9] or [A-F] or [a-f]
```

19.15.2.5. _IS_LOW

```
#define _IS_LOW 8  
is lower case
```

19.15.2.6. _IS_PUN

```
#define _IS_PUN 64  
punctuation
```

19.15.2.7. _IS_SP

```
#define _IS_SP 1  
is space
```

19.15.2.8. _IS_UPP

```
#define _IS_UPP 4  
is upper case
```

19.15.2.9. _tolower

```
#define _tolower(  
    c ) tolower(c)
```

Псевдоним для `tolower` из SVID.

Псевдоним для `tolower` из SVID.

19.15.2.10. _toupper

```
#define _toupper(  
    c ) toupper(c)
```

Псевдоним для `toupper` из SVID.

Псевдоним для `toupper` из SVID.

19.15.3. Функции

19.15.3.1. isalnum()

```
int isalnum (  
           int c )
```

Проверить, является ли символ буквой или цифрой.

Аргументы

c Символ для проверки.

Возвращает

0, если символ не является буквой или цифрой, не-0 иначе.

19.15.3.2. isalpha()

```
int isalpha (  
           int c )
```

Проверить, является ли символ буквой.

Аргументы

c Символ для проверки.

Возвращает

0, если символ не является буквой, не-0 иначе.

19.15.3.3. isascii()

```
int isascii (  
           int c )
```

Проверить, является ли символ ascii-символом.

Аргументы

c Символ для проверки.

Возвращает

0, если символ не является ascii-символом, не-0 иначе.

19.15.3.4. `isblank()`

```
int isblank (
    int c )
```

Проверить, является ли символ пробелом или табуляцией.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является пробелом или табуляцией, не-0 иначе.

19.15.3.5. `isctrl()`

```
int isctrl (
    int c )
```

Проверить, является ли символ управляющим символом.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является управляющим символом, не-0 иначе.

19.15.3.6. `isdigit()`

```
int isdigit (
    int c )
```

Проверить, является ли символ десятичной цифрой.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является десятичной цифрой, не-0 иначе.

19.15.3.7. isgraph()

```
int isgraph (  
    int c )
```

Проверить, является ли символ печатаемым символом, отличным от пробела.

Аргументы

c Символ для проверки.

Возвращает

0, если символ не является печатаемым символом, отличным от пробела, не-0 иначе.

19.15.3.8. islower()

```
int islower (  
    int c )
```

Проверить, является ли символ символом нижнего регистра.

Аргументы

c Символ для проверки.

Возвращает

0, если символ не является символом нижнего регистра, не-0 иначе.

19.15.3.9. isprint()

```
int isprint (  
    int c )
```

Проверить, является ли символ печатаемым символом.

Аргументы

c Символ для проверки.

Возвращает

0, если символ не является печатаемым символом, не-0 иначе.

19.15.3.10. ispunct()

```
int ispunct (  
    int c )
```

Проверить, является ли символ знаком препинания.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является знаком препинания, не-0 иначе.

19.15.3.11. isspace()

```
int isspace (  
    int c )
```

Проверить, является ли символ пробельным символом.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является пробельным символом, не-0 иначе.

Пробельными символами являются: ' ', '\f', '\n', '\r', '\t', '\v'.

19.15.3.12. isupper()

```
int isupper (  
    int c )
```

Проверить, является ли символ символом верхнего регистра.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является символом верхнего регистра, не-0 иначе.

19.15.3.13. isxdigit()

```
int isxdigit (  
    int c )
```

Проверить, является ли символ шестнадцатеричной цифрой.

Аргументы

`c` Символ для проверки.

Возвращает

0, если символ не является шестнадцатеричной цифрой, не-0 иначе.

19.15.3.14. toascii()

```
int toascii (  
    int c )
```

Преобразовать символ в символ `ascii`, округлив его до 7 младших битов.

Аргументы

`c` Символ для преобразования.

Возвращает

Преобразованный символ.

19.15.3.15. tolower()

```
int tolower (  
    int c )
```

Преобразовать символ в нижний регистр.

Аргументы

`c` Символ для преобразования.

Возвращает

Преобразованный символ или исходный символ, если преобразование невозможно.

19.15.3.16. toupper()

```
int toupper (  
    int c )
```

Преобразовать символ в верхний регистр.

Аргументы

c Символ для преобразования.

Возвращает

Преобразованный символ или исходный символ, если преобразование невозможно.

19.16. Файл `datetime.h`

Дополнительные функции для работы с датой/временем.

Структуры данных

- struct `date_time`
- struct `dtcompact`
Структура формата Дата / Время (DOS-совместимая).

Определения типов

- typedef struct `date_time` `DATE_TIME`
- typedef struct `dtcompact` `DT_COMPACT`
Структура формата Дата / Время (DOS-совместимая).

Функции

- `time_t` `convertDateTimeTime` (const `DATE_TIME` *pDateTime)
- int `date` (const char *newDate)
- int `diffdate` (const `DATE_TIME` *pDateTime1, const `DATE_TIME` *pDateTime0)
- `STATUS` `increaseDateTimeBySecond` (`DATE_TIME` *pDateTime)
- int `setTime` (const char *newTime)

19.16.1. Типы

19.16.1.1. `DATE_TIME`

```
typedef struct date_time DATE_TIME
```

Структура формата Дата / Время.

19.16.1.2. `DT_COMPACT`

```
typedef struct dtcompact DT_COMPACT
```

Компактное представление даты / времени совместимое с DOS.

19.16.2. Функции

19.16.2.1. `convertDateTimeTime()`

```
time_t convertDateTimeTime (  
    const DATE_TIME * pDateTime )
```

Преобразовать структуру `DATE_TIME` в абсолютное время.

Аргументы	
<code>pDateTime</code>	Структура <code>DATE_TIME</code> .

Возвращает

Кол-во секунд, прошедших с 1 января 1970 года для представленного времени или -1 при ошибке.

19.16.2.2. date()

```
int date (
    const char * newDate )
```

Распечатать или/и установить дату.

Функция потенциально устанавливает новую дату, после чего распечатывает текущую/новую дату в stdout.

Аргументы

<i>newDate</i>	Указатель на строку с новой датой в формате "ДД/ММ/ГГГГ" или NULL, если установка даты не требуется.
----------------	--

Возвращает

Всегда возвращает 0.

19.16.2.3. diffdate()

```
int diffdate (
    const DATE_TIME * pDateTime1,
    const DATE_TIME * pDateTime0 )
```

Получить кол-во секунд, прошедших со времени pDateTime0 и до времени pDateTime1.

Аргументы

<i>pDateTime1</i>	Конечная временная точка.
-------------------	---------------------------

<i>pDateTime0</i>	Начальная временная точка.
-------------------	----------------------------

Возвращает

Количество секунд разницы.



Работает только для дат моложе 1970 года.

19.16.2.4. increaseDateTimeBySecond()

```
STATUS increaseDateTimeBySecond (  
    DATE_TIME * pDateTime )
```

Увеличить значение переменной *DATE_TIME* на одну секунду.

Аргументы

pDateTime Переменная, значение которой нужно увеличить.

Возвращает

ОК при успехе, ERROR иначе.

19.16.2.5. setTime()

```
int setTime (  
    const char * newTime )
```

Распечатать или/и установить время.

Функция потенциально устанавливает новое время, после чего распечатывает текущее/новое время в stdout.

Аргументы

newTime Указатель на строку с новой датой в формате "ЧЧ:ММ:СС" или NULL, если установка даты не требуется.

Возвращает

Всегда возвращает 0.

19.17. Файл de2.h

Allwinner DE2.

Структуры данных

- struct *tScreenDeviceMode*
Структура настройки подключения дисплея LCD.

Перечисления

- enum *eOverlayDataFormat* {
ovDataFormat_ARGB_8888 = 0x00 , *ovDataFormat_ABGR_8888* = 0x01 , *ovDataFormat_RGBA_8888* = 0x02 ,
ovDataFormat_BGRA_8888 = 0x03 ,
ovDataFormat_XRGB_8888 = 0x04 , *ovDataFormat_XBGR_8888* = 0x05 , *ovDataFormat_RGBX_8888* = 0x06 ,
ovDataFormat_BGRX_8888 = 0x07 ,
ovDataFormat_RGB_888 = 0x08 , *ovDataFormat_BGR_888* = 0x09 , *ovDataFormat_RGB_565* = 0x0A ,
ovDataFormat_BGR_565 = 0x0B ,
ovDataFormat_ARGB_4444 = 0x0C , *ovDataFormat_ABGR_4444* = 0x0D , *ovDataFormat_RGBA_4444* = 0x0E ,
ovDataFormat_BGRA_4444 = 0x0F ,
ovDataFormat_ARGB_1555 = 0x10 , *ovDataFormat_ABGR_1555* = 0x11 , *ovDataFormat_RGBA_5551* = 0x12 ,
ovDataFormat_BGRA_5551 = 0x13 }
Формат данных Overlay.
- enum *eScreenDeviceType* { *screenType_Lcd_480x272* , *screenType_Lcd_800x480* , *screenType_TM043NBH02* ,
screenType_TM070RxH10 }
Пресеты параметров LCD дисплеев

Инициализация и управление аппаратным модулем

- int * *de2FlipScreenAndConstr* ()
Смена поверхностей.
- void *de2GetDefaultScreenMode* (*tScreenDeviceMode* *mode, *eScreenDeviceType* screen)
Инициализация структуры описания дисплея.
- int * *de2Init* (const *tScreenDeviceMode* *mode, *eOverlayDataFormat* format)
Инициализация модуля.
- void *de2SetBacklight* (unsigned int brightness)
Управление подсветкой дисплея.
- int *de2WaitVerticalRetrace* ()
Ожидание обратного хода луча.

19.17.1. Подробное описание

Работа с аппаратным модулем "Display Engine 2" — вывод графики на дисплей LCD.

Подключение:

```
#include <multimedia/de2.h>
```

Makefile:

```
LIBRARIES += -l_de2
```

См. также

Общее описание работы с графической подсистемой в главе *Графическая подсистема*.

История **версий** библиотеки:

- **1.2** — Полная настройка аппаратного модуля с использованием одного канала *Overlay*.

19.17.2. Перечисления

19.17.2.1. eOverlayDataFormat

enum *eOverlayDataFormat*

Поддерживаемые аппаратным модулем форматы цветопередачи.

Элементы перечислений

ovDataFormat_ARGB_8888

ovDataFormat_ABGR_8888

ovDataFormat_RGBA_8888

ovDataFormat_BGRA_8888

ovDataFormat_XRGB_8888

ovDataFormat_XBGR_8888

ovDataFormat_RGBX_8888

ovDataFormat_BGRX_8888

ovDataFormat_RGB_888

ovDataFormat_BGR_888

ovDataFormat_RGB_565

ovDataFormat_BGR_565

ovDataFormat_ARGB_4444

ovDataFormat_ABGR_4444

ovDataFormat_RGBA_4444

ovDataFormat_BGRA_4444

ovDataFormat_ARGB_1555

ovDataFormat_ABGR_1555

ovDataFormat_RGBA_5551

ovDataFormat_BGRA_5551

00044

{

```

00045     ovDataFormat_ARGB_8888 = 0x00,
00046     ovDataFormat_ABGR_8888 = 0x01,
00047     ovDataFormat_RGBA_8888 = 0x02,
00048     ovDataFormat_BGRA_8888 = 0x03,
00049     ovDataFormat_XRGB_8888 = 0x04,
00050     ovDataFormat_XBGR_8888 = 0x05,
00051     ovDataFormat_RGBX_8888 = 0x06,
00052     ovDataFormat_BGRX_8888 = 0x07,
00053     ovDataFormat_RGB_888  = 0x08,
00054     ovDataFormat_BGR_888  = 0x09,
00055     ovDataFormat_RGB_565  = 0x0A,
00056     ovDataFormat_BGR_565  = 0x0B,
00057     ovDataFormat_ARGB_4444 = 0x0C,
00058     ovDataFormat_ABGR_4444 = 0x0D,
00059     ovDataFormat_RGBA_4444 = 0x0E,
00060     ovDataFormat_BGRA_4444 = 0x0F,
00061     ovDataFormat_ARGB_1555 = 0x10,
00062     ovDataFormat_ABGR_1555 = 0x11,
00063     ovDataFormat_RGBA_5551 = 0x12,
00064     ovDataFormat_BGRA_5551 = 0x13
00065 } eOverlayDataFormat;
    
```

19.17.2.2. eScreenDeviceType

enum *eScreenDeviceType*

Элементы перечислений

screenType_Lcd_480x272	Описание стандартного LCD дисплея 480x272
screenType_Lcd_800x480	Описание стандартного LCD дисплея 800x480
screenType_TM043NBH02	LCD дисплей TIANMA 4,3" 480x272
screenType_TM070RxH10	LCD дисплей TIANMA 7" 800x480

```

00032     {
00033     screenType_Lcd_480x272,
00034     screenType_Lcd_800x480,
00035     screenType_TM043NBH02,
00036     screenType_TM070RxH10,
00037 } eScreenDeviceType;
    
```

19.17.3. Функции

19.17.3.1. de2FlipScreenAndConstr()

int* de2FlipScreenAndConstr ()

Функция меняет местами видимую (*SCREEN*) и конструируемую (*CONSTR*) поверхности.

Возвращает

Указатель на область видеопамати конструируемой поверхности. Изменяется каждый раз при смене поверхностей.

19.17.3.2. de2GetDefaultScreenMode()

```
void de2GetDefaultScreenMode (
    tScreenDeviceMode * mode,
    eScreenDeviceType screen )
```

Функция заполняет структуру описания дисплея значениями по умолчанию — используются настройки, подходящие для большинства дисплеев. Заполненную структуру следует использовать при инициализации модуля `de2Init()`. При необходимости некоторые параметры можно изменить до инициализации модуля.

Аргументы

mode Заполняемая структура описания дисплея.

screen Выбранный тип дисплея.

19.17.3.3. de2Init()

```
int* de2Init (
    const tScreenDeviceMode * mode,
    eOverlayDataFormat format )
```

Подготовка памяти экранной области, инициализация переменных.

Аргументы

mode Параметры дисплея (можно воспользоваться `de2GetDefaultScreenMode()` для получения значений по умолчанию).

format Формат данных изображения, используемый в аппаратном модуле **Display-Engine**.

Возвращает

Указатель на область видеопамати конструируемой поверхности.

19.17.3.4. de2SetBacklight()

```
void de2SetBacklight (
    unsigned int brightness )
```

Для управления подсветкой в *Конфигурации проекта* должен быть указан используемый для этого канал ШИМ. Настройка канала ШИМ производится в функции инициализации `de2Init()`. Управление подсветкой возможно только после успешной инициализации канала.

Аргументы

brightness Значение яркости от **0** до **100**.

19.17.3.5. de2WaitVerticalRetrace()

int de2WaitVerticalRetrace ()

Функция ожидает начала обратного хода луча на мониторе для избежания появления строба.

Возвращает

OK, либо код ошибки, возвращаемый *semTake()*.

19.18. Файл `drivers.dox`

19.19. Файл `env_vars.h`

Дополнительные функции для работы с переменными окружения.

Структуры данных

- struct `env_var`

Функции

- struct `env_var` * `getenv_var` (const char *name)
- int `printenv` (const char *name)

19.19.1. Подробное описание

Некоторые функции, дополняющие стандартный функционал `stdlib.h`.

19.19.2. Функции

19.19.2.1. `getenv_var()`

```
struct env_var* getenv_var (  
    const char * name )
```

Получить структуру `env_var` переменной окружения.

Аргументы

<code>name</code>	Имя переменной окружения.
-------------------	---------------------------

Возвращает

Указатель на структуру `env_var`, либо `NULL`, если переменная окружения не была найдена.

19.19.2.2. `printenv()`

```
int printenv (  
    const char * name )
```

Распечатать в `stdout` значение переменной окружения в формате 'имя = значение'.

Аргументы

<code>name</code>	Имя переменной окружения, которая должна быть распечатана, или <code>NULL</code> , для печати всех переменных.
-------------------	--

Возвращает

Общее количество имеющихся переменных окружения.

19.20. Файл `errno-base.h`

Заголовочный файл для обратной совместимости.

19.20.1. Подробное описание

Раньше содержал в себе часть [errno.h](#).

19.21. Файл `errno.h`

Обработка причин ошибок в библиотечных функциях.

Переменные

- `int errno`

Коды ошибок

- `#define E2BIG 7`
- `#define EACCES 13`
- `#define EADDRINUSE 98`
- `#define EADDRNOTAVAIL 99`
- `#define EADV 68`
- `#define EAFNOSUPPORT 97`
- `#define EAGAIN 11`
- `#define EALREADY 114`
- `#define EBADE 52`
- `#define EBADF 9`
- `#define EBADFD 77`
- `#define EBADMSG 74`
- `#define EBADR 53`
- `#define EBADRQC 56`
- `#define EBADSLT 57`
- `#define EBFONT 59`
- `#define EBUSY 16`
- `#define ECANCELED 125`
- `#define ECHILD 10`
- `#define ECHRNG 44`
- `#define ECOMM 70`
- `#define ECONNABORTED 103`
- `#define ECONNREFUSED 111`
- `#define ECONNRESET 104`
- `#define EDEADLK 35`
- `#define EDEADLOCK EDEADLK`
- `#define EDESTADDRREQ 89`
- `#define EDOM 33`
- `#define EDOTDOT 73`
- `#define EDQUOT 122`
- `#define EEXIST 17`
- `#define EFAULT 14`
- `#define EFBIG 27`
- `#define EHOSTDOWN 112`
- `#define EHOSTUNREACH 113`
- `#define EHWPOISON 133`
- `#define EIDRM 43`
- `#define EILSEQ 84`
- `#define EINPROGRESS 115`
- `#define EINTR 4`
- `#define EINVAL 22`
- `#define EIO 5`
- `#define EISCONN 106`
- `#define EISDIR 21`
- `#define EISNAM 120`
- `#define EKEYEXPIRED 127`
- `#define EKEYREJECTED 129`
- `#define EKEYREVOKED 128`
- `#define EL2HLT 51`
- `#define EL2NSYNC 45`
- `#define EL3HLT 46`
- `#define EL3RST 47`

- #define *ELIBACC* 79
- #define *ELIBBAD* 80
- #define *ELIBEXEC* 83
- #define *ELIBMAX* 82
- #define *ELIBSCN* 81
- #define *ELNRNG* 48
- #define *ELOOP* 40
- #define *EMEDIUMTYPE* 124
- #define *EMFILE* 24
- #define *EMLINK* 31
- #define *EMSGSIZE* 90
- #define *EMULTIHOP* 72
- #define *ENAMETOOLONG* 36
- #define *ENAVAIL* 119
- #define *ENETDOWN* 100
- #define *ENETRESET* 102
- #define *ENETUNREACH* 101
- #define *ENFILE* 23
- #define *ENOANO* 55
- #define *ENOBUFS* 105
- #define *ENOCSS* 50
- #define *ENODATA* 61
- #define *ENODEV* 19
- #define *ENOENT* 2
- #define *ENOEXEC* 8
- #define *ENOKEY* 126
- #define *ENOLCK* 37
- #define *ENOLINK* 67
- #define *ENOMEDIUM* 123
- #define *ENOMEM* 12
- #define *ENOMSG* 42
- #define *ENONET* 64
- #define *ENOPKG* 65
- #define *ENOPROTOOPT* 92
- #define *ENOSPC* 28
- #define *ENOSR* 63
- #define *ENOSTR* 60
- #define *ENOSYS* 38
- #define *ENOTBLK* 15
- #define *ENOTCONN* 107
- #define *ENOTDIR* 20
- #define *ENOTEMPTY* 39
- #define *ENOTNAM* 118
- #define *ENOTRECOVERABLE* 131
- #define *ENOTSOCK* 88
- #define *ENOTSUP* 200
- #define *ENOTTY* 25
- #define *ENOTUNIQ* 76
- #define *ENXIO* 6
- #define *EOPNOTSUPP* 95
- #define *E_OVERFLOW* 75
- #define *EOWNERDEAD* 130
- #define *EPERM* 1
- #define *EPFNOSUPPORT* 96
- #define *EPIPE* 32
- #define *EPROTO* 71
- #define *EPROTONOSUPPORT* 93
- #define *EPROTOTYPE* 91
- #define *ERANGE* 34
- #define *EREMCHG* 78
- #define *EREMOTE* 66
- #define *EREMOTEIO* 121
- #define *ERESTART* 85

- #define *ERFKILL* 132
- #define *EROFS* 30
- #define *ESHUTDOWN* 108
- #define *ESOCKNOSUPPORT* 94
- #define *ESPIPE* 29
- #define *ESRCH* 3
- #define *ESRMNT* 69
- #define *ESTALE* 116
- #define *ESTRPIPE* 86
- #define *ETIME* 62
- #define *ETIMEDOUT* 110
- #define *ETOOMANYREFS* 109
- #define *ETXTBSY* 26
- #define *EUCLEAN* 117
- #define *EUNATCH* 49
- #define *EUSERS* 87
- #define *EWOULDLOCK AGAIN*
- #define *EXDEV* 18
- #define *EXFULL* 54

19.21.1. Подробное описание

См. стандарт C11 7.5.

См. также

[C11 standard 7.5.](#)



В основном используется 'внешними' библиотеками, например, FFmpeg или LIBZ.

19.21.2. Макросы

19.21.2.1. E2BIG

```
#define E2BIG 7
```

Argument list too long

19.21.2.2. EACCES

```
#define EACCES 13
```

Permission denied

19.21.2.3. EADDRINUSE

```
#define EADDRINUSE 98
```

Address already in use

19.21.2.4. EADDRNOTAVAIL

```
#define EADDRNOTAVAIL 99
```

Cannot assign requested address

19.21.2.5. EADV

```
#define EADV 68
```

Advertise error

19.21.2.6. EAFNOSUPPORT

```
#define EAFNOSUPPORT 97
```

Address family not supported by protocol

19.21.2.7. EAGAIN

```
#define EAGAIN 11
```

Try again

19.21.2.8. EALREADY

```
#define EALREADY 114
```

Operation already in progress

19.21.2.9. EBADE

```
#define EBADE 52
```

Invalid exchange

19.21.2.10. EBADF

```
#define EBADF 9
```

Bad file number

19.21.2.11. EBADFD

```
#define EBADFD 77
```

File descriptor in bad state

19.21.2.12. EBADMSG

```
#define EBADMSG 74
```

Not a data message

19.21.2.13. EBADR

```
#define EBADR 53
```

Invalid request descriptor

19.21.2.14. EBADRQC

```
#define EBADRQC 56  
Invalid request code
```

19.21.2.15. EBADSLT

```
#define EBADSLT 57  
Invalid slot
```

19.21.2.16. EBFONT

```
#define EBFONT 59  
Bad font file format
```

19.21.2.17. EBUSY

```
#define EBUSY 16  
Device or resource busy
```

19.21.2.18. ECANCELED

```
#define ECANCELED 125  
Operation Canceled
```

19.21.2.19. ECHILD

```
#define ECHILD 10  
No child processes
```

19.21.2.20. ECHRNG

```
#define ECHRNG 44  
Channel number out of range
```

19.21.2.21. ECOMM

```
#define ECOMM 70  
Communication error on send
```

19.21.2.22. ECONNABORTED

```
#define ECONNABORTED 103  
Software caused connection abort
```

19.21.2.23. ECONNREFUSED

```
#define ECONNREFUSED 111  
Connection refused
```

19.21.2.24. ECONNRESET

```
#define ECONNRESET 104  
Connection reset by peer
```

19.21.2.25. EDEADLK

```
#define EDEADLK 35  
Resource deadlock would occur
```

19.21.2.26. EDEADLOCK

```
#define EDEADLOCK EDEADLK
```

19.21.2.27. EDESTADDRREQ

```
#define EDESTADDRREQ 89  
Destination address required
```

19.21.2.28. EDOM

```
#define EDOM 33  
Math argument out of domain of func
```

19.21.2.29. EDOTDOT

```
#define EDOTDOT 73  
RFS specific error
```

19.21.2.30. EDQUOT

```
#define EDQUOT 122  
Quota exceeded
```

19.21.2.31. EEXIST

```
#define EEXIST 17  
File exists
```

19.21.2.32. EFAULT

```
#define EFAULT 14  
Bad address
```

19.21.2.33. EFBIG

```
#define EFBIG 27  
File too large
```

19.21.2.34. EHOSTDOWN

```
#define EHOSTDOWN 112  
Host is down
```

19.21.2.35. EHOSTUNREACH

```
#define EHOSTUNREACH 113  
No route to host
```

19.21.2.36. EHWPOISON

```
#define EHWPOISON 133  
Memory page has hardware error
```

19.21.2.37. EIDRM

```
#define EIDRM 43  
Identifier removed
```

19.21.2.38. EILSEQ

```
#define EILSEQ 84  
Illegal byte sequence
```

19.21.2.39. EINPROGRESS

```
#define EINPROGRESS 115  
Operation now in progress
```

19.21.2.40. EINTR

```
#define EINTR 4  
Interrupted system call
```

19.21.2.41. EINVAL

```
#define EINVAL 22
```

Invalid argument

19.21.2.42. EIO

```
#define EIO 5
```

I/O error

19.21.2.43. EISCONN

```
#define EISCONN 106
```

Transport endpoint is already connected

19.21.2.44. EISDIR

```
#define EISDIR 21
```

Is a directory

19.21.2.45. EISNAM

```
#define EISNAM 120
```

Is a named type file

19.21.2.46. EKEYEXPIRED

```
#define EKEYEXPIRED 127
```

Key has expired

19.21.2.47. EKEYREJECTED

```
#define EKEYREJECTED 129
```

Key was rejected by service

19.21.2.48. EKEYREVOKED

```
#define EKEYREVOKED 128
```

Key has been revoked

19.21.2.49. EL2HLT

```
#define EL2HLT 51
```

Level 2 halted

19.21.2.50. EL2NSYNC

```
#define EL2NSYNC 45  
Level 2 not synchronized
```

19.21.2.51. EL3HLT

```
#define EL3HLT 46  
Level 3 halted
```

19.21.2.52. EL3RST

```
#define EL3RST 47  
Level 3 reset
```

19.21.2.53. ELIBACC

```
#define ELIBACC 79  
Can not access a needed shared library
```

19.21.2.54. ELIBBAD

```
#define ELIBBAD 80  
Accessing a corrupted shared library
```

19.21.2.55. ELIBEXEC

```
#define ELIBEXEC 83  
Cannot exec a shared library directly
```

19.21.2.56. ELIBMAX

```
#define ELIBMAX 82  
Attempting to link in too many shared libraries
```

19.21.2.57. ELIBSCN

```
#define ELIBSCN 81  
.lib section in a.out corrupted
```

19.21.2.58. ELNRNG

```
#define ELNRNG 48  
Link number out of range
```

19.21.2.59. ELOOP

```
#define ELOOP 40
```

Too many symbolic links encountered

19.21.2.60. EMEDIUMTYPE

```
#define EMEDIUMTYPE 124
```

Wrong medium type

19.21.2.61. EMFILE

```
#define EMFILE 24
```

Too many open files

19.21.2.62. EMLINK

```
#define EMLINK 31
```

Too many links

19.21.2.63. EMSGSIZE

```
#define EMSGSIZE 90
```

Message too long

19.21.2.64. EMULTIHOP

```
#define EMULTIHOP 72
```

Multihop attempted

19.21.2.65. ENAMETOOLONG

```
#define ENAMETOOLONG 36
```

File name too long

19.21.2.66. ENAVAIL

```
#define ENAVAIL 119
```

No XENIX semaphores available

19.21.2.67. ENETDOWN

```
#define ENETDOWN 100
```

Network is down

19.21.2.68. ENETRESET

```
#define ENETRESET 102
```

Network dropped connection because of reset

19.21.2.69. ENETUNREACH

```
#define ENETUNREACH 101
```

Network is unreachable

19.21.2.70. ENFILE

```
#define ENFILE 23
```

File table overflow

19.21.2.71. ENOANO

```
#define ENOANO 55
```

No anode

19.21.2.72. ENOBUFS

```
#define ENOBUFS 105
```

No buffer space available

19.21.2.73. ENOCSI

```
#define ENOCSI 50
```

No CSI structure available

19.21.2.74. ENODATA

```
#define ENODATA 61
```

No data available

19.21.2.75. ENODEV

```
#define ENODEV 19
```

No such device

19.21.2.76. ENOENT

```
#define ENOENT 2
```

No such file or directory

19.21.2.77. ENOEXEC

```
#define ENOEXEC 8
```

Exec format error

19.21.2.78. ENOKEY

```
#define ENOKEY 126
```

Required key not available

19.21.2.79. ENOLCK

```
#define ENOLCK 37
```

No record locks available

19.21.2.80. ENOLINK

```
#define ENOLINK 67
```

Link has been severed

19.21.2.81. ENOMEDIUM

```
#define ENOMEDIUM 123
```

No medium found

19.21.2.82. ENOMEM

```
#define ENOMEM 12
```

Out of memory

19.21.2.83. ENOMSG

```
#define ENOMSG 42
```

No message of desired type

19.21.2.84. ENONET

```
#define ENONET 64
```

Machine is not on the network

19.21.2.85. ENOPKG

```
#define ENOPKG 65
```

Package not installed

19.21.2.86. ENOPROTOOPT

```
#define ENOPROTOOPT 92  
Protocol not available
```

19.21.2.87. ENOSPC

```
#define ENOSPC 28  
No space left on device
```

19.21.2.88. ENOSR

```
#define ENOSR 63  
Out of streams resources
```

19.21.2.89. ENOSTR

```
#define ENOSTR 60  
Device not a stream
```

19.21.2.90. ENOSYS

```
#define ENOSYS 38  
Function not implemented
```

19.21.2.91. ENOTBLK

```
#define ENOTBLK 15  
Block device required
```

19.21.2.92. ENOTCONN

```
#define ENOTCONN 107  
Transport endpoint is not connected
```

19.21.2.93. ENOTDIR

```
#define ENOTDIR 20  
Not a directory
```

19.21.2.94. ENOTEMPTY

```
#define ENOTEMPTY 39  
Directory not empty
```

19.21.2.95. ENOTNAM

```
#define ENOTNAM 118
```

Not a XENIX named type file

19.21.2.96. ENOTRECOVERABLE

```
#define ENOTRECOVERABLE 131
```

Robust mutex: State not recoverable

19.21.2.97. ENOTSOCK

```
#define ENOTSOCK 88
```

Socket operation on non-socket

19.21.2.98. ENOTSUP

```
#define ENOTSUP 200
```

19.21.2.99. ENOTTY

```
#define ENOTTY 25
```

Not a typewriter

19.21.2.100. ENOTUNIQ

```
#define ENOTUNIQ 76
```

Name not unique on network

19.21.2.101. ENXIO

```
#define ENXIO 6
```

No such device or address

19.21.2.102. EOPNOTSUPP

```
#define EOPNOTSUPP 95
```

Operation not supported on transport endpoint

19.21.2.103. EOVERFLOW

```
#define EOVERFLOW 75
```

Value too large for defined data type

19.21.2.104. EOWNERDEAD

```
#define EOWNERDEAD 130
```

Robust mutex: Owner died

19.21.2.105. EPERM

```
#define EPERM 1
```

Operation not permitted

19.21.2.106. EPNOSUPPORT

```
#define EPNOSUPPORT 96
```

Protocol family not supported

19.21.2.107. EPIPE

```
#define EPIPE 32
```

Broken pipe

19.21.2.108. EPROTO

```
#define EPROTO 71
```

Protocol error

19.21.2.109. EPROTONOSUPPORT

```
#define EPROTONOSUPPORT 93
```

Protocol not supported

19.21.2.110. EPROTOTYPE

```
#define EPROTOTYPE 91
```

Protocol wrong type for socket

19.21.2.111. ERANGE

```
#define ERANGE 34
```

Math result not representable

19.21.2.112. EREMCHG

```
#define EREMCHG 78
```

Remote address changed

19.21.2.113. EREMOTE

```
#define EREMOTE 66
```

Object is remote

19.21.2.114. EREMOTEIO

```
#define EREMOTEIO 121
```

Remote I/O error

19.21.2.115. ERESTART

```
#define ERESTART 85
```

Interrupted system call should be restarted

19.21.2.116. ERFKILL

```
#define ERFKILL 132
```

Operation not possible due to RF-kill

19.21.2.117. EROFS

```
#define EROFS 30
```

Read-only file system

19.21.2.118. ESHUTDOWN

```
#define ESHUTDOWN 108
```

Cannot send after transport endpoint shutdown

19.21.2.119. ESOCKTNOSUPPORT

```
#define ESOCKTNOSUPPORT 94
```

Socket type not supported

19.21.2.120. ESPIPE

```
#define ESPIPE 29
```

Illegal seek

19.21.2.121. ESRCH

```
#define ESRCH 3
```

No such process

19.21.2.122. ESRMNT

```
#define ESRMNT 69  
Srmount error
```

19.21.2.123. ESTALE

```
#define ESTALE 116  
Stale NFS file handle
```

19.21.2.124. ESTRPIPE

```
#define ESTRPIPE 86  
Streams pipe error
```

19.21.2.125. ETIME

```
#define ETIME 62  
Timer expired
```

19.21.2.126. ETIMEDOUT

```
#define ETIMEDOUT 110  
Connection timed out
```

19.21.2.127. ETOOMANYREFS

```
#define ETOOMANYREFS 109  
Too many references: cannot splice
```

19.21.2.128. ETXTBSY

```
#define ETXTBSY 26  
Text file busy
```

19.21.2.129. EUCLEAN

```
#define EUCLEAN 117  
Structure needs cleaning
```

19.21.2.130. EUNATCH

```
#define EUNATCH 49  
Protocol driver not attached
```

19.21.2.131. EUSERS

```
#define EUSERS 87
```

Too many users

19.21.2.132. EWOULDBLOCK

```
#define EWOULDBLOCK EAGAIN
```

Operation would block

19.21.2.133. EXDEV

```
#define EXDEV 18
```

Cross-device link

19.21.2.134. EXFULL

```
#define EXFULL 54
```

Exchange full

19.21.3. Переменные**19.21.3.1. errno**

```
int errno [extern]
```

Переменная, хранящая в себе код ошибки.

19.22. Файл `filesyst.h`

Дополнительные функции для работы с файловой системой.

Функции

- *STATUS copy* (const char *srcMask, const char *dstMask, *bool* overwriteDst)
- *STATUS copyFile* (const char *src, const char *dst, *bool* overwriteDst)
- int *del* (const char *fileMask)
- *STATUS dir* (const char *dirPath, *bool* usePageMode)
- *STATUS dirCopy* (const char *src, const char *dst)
- *STATUS dirCopy_Delay* (const char *src, const char *dst, int delay)
- *STATUS dirIsEmpty* (const char *dir)
- char ** *findFilesInDirAndSubdirs* (const char *fileMask, const char *searchPath, unsigned int maxNestingDepth, int *arrayLength)
- int *getFileSize* (const char *filePath)
- void * *loadFile* (const char *filePath)
- void * *loadFileSz* (const char *filePath, int *pSizeVar)
- *STATUS move* (const char *srcMask, const char *dstMask, *bool* overwriteDst)
- *STATUS removeContentFromDir* (const char *dir)
- const char * *setWorkDevice* (const char *devName)
- *STATUS silentDirCopy* (const char *src, const char *dst)
- *STATUS type* (const char *filePath)

19.22.1. Подробное описание

См. также

Общее описание системы ввода / вывода см. в главе [Базовая система ввода / вывода](#).

19.22.2. Функции

19.22.2.1. `copy()`

```
STATUS copy (
    const char * srcMask,
    const char * dstMask,
    bool overwriteDst )
```

Скопировать файл(ы) по маске.

Аргументы	
<i>srcMask</i>	Маска копируемых файлов.
<i>dstMask</i>	Маска файлов-копий.
<i>overwriteDst</i>	Нужно ли перезаписывать файлы, если они существуют.

Возвращает

OK, если при копировании не произошло ошибок (отсутствие файлов для копирования не считается ошибкой), *ERROR* иначе.

19.22.2.2. copyFile()

```
STATUS copyFile (  
    const char * src,  
    const char * dst,  
    bool overwriteDst )
```

Скопировать файл.

Аргументы	
<i>src</i>	Путь к копируемому файлу.
<i>dst</i>	Путь к копии.
<i>overwriteDst</i>	Нужно ли перезаписывать файл с именем <i>dst</i> , если он существует.

Возвращает

OK при успехе, *ERROR* иначе.

19.22.2.3. del()

```
int del (  
    const char * fileMask )
```

Удалить файл(ы) по маске.

Удаляемые файлы будут перечислены в stdout.

Аргументы	
<i>fileMask</i>	Имя файла или маска файла.

Возвращает

OK, если удаление прошло успешно или файлы не были обнаружены, *ERROR* иначе.

19.22.2.4. dir()

```
STATUS dir (  
    const char * dirPath,  
    bool usePageMode )
```

Распечатать содержимое каталога в stdout.

Аргументы	
<i>dirPath</i>	Путь к каталогу.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>usePageMode</i>	Использовать ли страничный режим.
--------------------	-----------------------------------

Возвращает

OK при успехе, *ERROR* иначе.

При использовании страничного режима будет печататься по 24 файла, после чего ожидается символ в stdin, для продолжения печати. В не-страничном режиме будут печататься всё содержимое сразу.

19.22.2.5. dircopy()

```
STATUS dircopy (  
    const char * src,  
    const char * dst )
```

Скопировать содержимое каталога src в каталог dst.

Функция будет печатать в stdout созданные каталоги и скопированные файлы. При конфликтах файлы будут заменены.

Аргументы

<i>src</i>	Путь к каталогу, из которого будет производится копирование.
------------	--

<i>dst</i>	Путь к каталогу, в который должно производится копирование. Каталог не должен существовать.
------------	---

Возвращает

OK, если при копировании не произошло ошибок (отсутствие файлов для копирования не считается ошибкой), *ERROR* иначе.

19.22.2.6. dirCopy_Delay()

```
STATUS dirCopy_Delay (  
    const char * src,  
    const char * dst,  
    int delay )
```

Скопировать содержимое каталога src в каталог dst с задержкой в ходе копирования.

Функция не будет ничего печатать в stdout. При конфликтах файлы будут заменены.

Аргументы

<i>src</i>	Путь к каталогу, из которого будет производится копирование.
------------	--

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>dst</i>	Путь к каталогу, в который должно производиться копирование.
<i>delay</i>	Задержка в ходе копирования, -1 для отключения задержки, 0 для принудительного переключения задач в ходе копирования, положительное число для задержки (чем выше, тем выше задержка итогового копирования).

Возвращает

OK, если при копировании не произошло ошибок (отсутствие файлов для копирования не считается ошибкой), *ERROR* иначе.

19.22.2.7. dirIsEmpty()

```
STATUS dirIsEmpty (
    const char * dir )
```

Проверить, пуст ли каталог (есть ли там файлы и/или каталоги).

Аргументы

<i>dir</i>	Путь к каталогу
------------	-----------------

Возвращает

OK, если каталог пуст, *ERROR*, если не пуст.

19.22.2.8. findFilesInDirAndSubdirs()

```
char** findFilesInDirAndSubdirs (
    const char * fileMask,
    const char * searchPath,
    unsigned int maxNestingDepth,
    int * arrayLength )
```

Найти все файлы соответствующие маске в каталоге и его подкаталогах.

Для директории со следующей структурой:

```
dir
|  iniFile1.ini
|
+--- folder1
|   +--- folder1_1
|   |       iniFile1_1_1.ini
|   +--- folder1_2
|           pngFile1_2_1.png
+--- folder2
```

```
|      iniFile2_1.ini  
|      pngFile2_1.png  
+---folder3  
+---folder4  
      iniFile4_1.ini
```

Вызов функции

```
findFilesInDirAndSubdirs("{*.ini"}", dirPath, 1, &buf);
```

вернёт массив длиной 3 со следующим содержимым:

```
["folder2/iniFile2_1.ini", "folder4/iniFile4_1.ini", "iniFile1.ini"]
```



Полученный массив необходимо будет освободить: сначала каждый элемент, затем сам массив.

Аргументы

<i>fileMask</i>	Маска файла.
<i>searchPath</i>	Путь, откуда нужно начинать поиск.
<i>maxNestingDepth</i>	Максимальная глубина вложенности, -1 для поиска без ограничений.
out <i>arrayLength</i>	Длина возвращаемого массива.

Возвращает

Массив с путями найденных файлов относительно пути для поиска или *NULL*, если произошла ошибка или файлы не были обнаружены.

19.22.2.9. getFileSize()

```
int getFileSize (  
    const char * filePath )
```

Получить размер файла.

Аргументы

filePath Путь к файлу.

Возвращает

Размер файла ≥ 0 или отрицательное значение при ошибке.

19.22.2.10. loadFile()

```
void* loadFile (  
    const char * filePath )
```

Загрузить файл в динамическую память.

Аргументы

filePath Путь к файлу.

Возвращает

Выделенный блок памяти с содержимым файла или *NULL* при ошибке.



По истечении надобности возвращенную память следует освободить при помощи функции *free()*.

19.22.2.11. loadFileSz()

```
void* loadFileSz (  
    const char * filePath,  
    int * pSizeVar )
```

Загрузить файл в динамическую память и вернуть его размер.

Аргументы

filePath Путь к файлу.

pSizeVar Указатель на переменную, в которую будет записан размер загруженного файла.

Возвращает

Выделенный блок памяти с содержимым файла или *NULL* при ошибке.



По истечении надобности возвращенную память следует освободить при помощи функции *free()*.

19.22.2.12. move()

```
STATUS move (  
    const char * srcMask,  
    const char * dstMask,  
    bool overwriteDst )
```

Перенести файл(ы) по маске.

Аргументы	
<i>srcMask</i>	Маска переносимых файлов.
<i>dstMask</i>	Маска перенесенных файлов.
<i>overwriteDst</i>	Нужно ли перезаписывать файлы, если они существует.

Возвращает

OK, если при перемещении не произошло ошибок (отсутствие файлов для перемещения не считается ошибкой), *ERROR* иначе.

19.22.2.13. `removeContentFromDir()`

```
STATUS removeContentFromDir (  
    const char * dir )
```

Удалить содержимое каталога.

Не удаляет сам каталог.

Аргументы	
<i>dir</i>	Путь к каталогу.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.22.2.14. `setWorkDevice()`

```
const char* setWorkDevice (  
    const char * devName )
```

Установить устройство в качестве рабочего.

Аргументы	
<i>devName</i>	Имя устройства (например, имя тома).

Возвращает

Установленное рабочее устройство.

19.22.2.15. silentDirCopy()

```
STATUS silentDirCopy (  
    const char * src,  
    const char * dst )
```

Скопировать содержимое каталога *src* в каталог *dst*.

Функция не будет ничего печатать в `stdout`. При конфликтах файлы будут заменены.

Аргументы

src Путь к каталогу, из которого будет производится копирование.

dst Путь к каталогу, в который должно производится копирование.

Возвращает

OK, если при копировании не произошло ошибок (отсутствие файлов для копирования не считается ошибкой), *ERROR* иначе.

19.22.2.16. type()

```
STATUS type (  
    const char * filePath )
```

Распечатать файл в `stdout` в текстовом режиме.

Аргументы

filePath Путь к файлу.

Возвращает

OK, если файл был распечатан, *ERROR* иначе.



Бинарные файлы будут распечатаны некорректно.

19.23. Файл `fnames.h`

Функционал для работы с именами файлов.

Функции

- void `buildFileName` (const char **fileName*, const char **pathMask*, char **dstNameBuffer*)
- `STATUS compareNames` (const char **mask*, const char **name*)
- `STATUS expandFileName` (const char **fileName*, char **fullPathBuf*)
- const char * `extractDevice` (const char **path*, char **deviceBuf*)
- void `extractFileDevPath` (const char **filePath*, const char **rootPath*, char **dstBuffer*)
- void `extractFileDrive` (const char **filePath*, char **driveBuf*)
- void `extractFilePath` (const char **fileName*, char **pathBuf*)
- const char * `extractNextDir` (const char **path*, char **dirBuf*)
- const char * `fileNamePreprocess` (const char **path*, char **devBuf*, `DEV_HDR` ***dh*)
- const char * `getFileName` (const char **path*)
- char * `getFileNameNonConst` (char **path*)
- const char * `getWorkDevice` (void)
- const char * `getWorkDirectory` (const char **dev*)
- const char * `setWorkDevice` (const char **path*)

19.23.1. Функции

19.23.1.1. `buildFileName()`

```
void buildFileName (
    const char * fileName,
    const char * pathMask,
    char * dstNameBuffer )
```

Создать имя файла на основе имени файла и маски пути.

Аргументы	
<i>fileName</i>	Имя файла.
<i>pathMask</i>	Маска пути.
<i>dstNameBuffer</i>	Буфер под итоговое полное имя.

Функция берет маску пути и, с учетом маски, присоединяет к ней имя файла, создав в итоге полное имя (например, "text.txt" + "./dir/?*.*" = "C:/currentDir/dir/text.txt").

19.23.1.2. `compareNames()`

```
STATUS compareNames (
    const char * mask,
    const char * name )
```

Сравнить маску поиска и имя файла.

Аргументы	
<i>mask</i>	Маска поиска.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>name</i>	Имя файла.
-------------	------------

Возвращает

OK, если имя соответствует маске, ERROR иначе.

19.23.1.3. expandFileName()

```
STATUS expandFileName (  
    const char * fileName,  
    char * fullPathBuf )
```

Снабдить имя файла полным путем к нему.

Аргументы

<i>fileName</i>	Имя файла или путь к нему.
-----------------	----------------------------

<i>fullPathBuf</i>	Буфер под итоговый полный путь.
--------------------	---------------------------------

Возвращает

OK при успехе, ERROR иначе.

Если путь к файлу изначально не полный, то он будет дополнен рабочим устройством и/или рабочим каталогом.

19.23.1.4. extractDevice()

```
const char* extractDevice (  
    const char * path,  
    char * deviceBuf )
```

Извлечь имя устройства из пути.

Аргументы

<i>path</i>	Путь.
-------------	-------

<i>deviceBuf</i>	Буфер под имя устройства.
------------------	---------------------------

Возвращает

Смещенный указатель, указывающий на часть буфера path, с которой начинается часть пути без имени устройства.

19.23.1.5. extractFileDevPath()

```
void extractFileDevPath (  
    const char * filePath,  
    const char * rootPath,  
    char * dstBuffer )
```

Извлечь из пути к файлу его каталог и присоединить его к корневому каталогу.

Аргументы

<i>filePath</i>	Путь к файлу.
<i>rootPath</i>	Корневой каталог.
<i>dstBuffer</i>	Итоговый буфер.

Примеры вызова: `extractFileDevPath("dir/file.txt", "C:/root/", buf)`,
`extractFileDevPath("/home/dir/file.txt", "none", buf)`. Результат в `buf`: "C:/root/dir/" и "/home/dir/" соответственно.

19.23.1.6. extractFileDrive()

```
void extractFileDrive (  
    const char * filePath,  
    char * driveBuf )
```

Извлечь из пути к файлу имя устройства.

Аргументы

<i>filePath</i>	Путь к файлу.
<i>driveBuf</i>	Буфер под имя устройства.

Выделяет часть до символа двоеточия из имени файла. Если имя устройства не удалось извлечь, то в буфер будет записано устройство по-умолчанию.

19.23.1.7. extractFilePath()

```
void extractFilePath (  
    const char * fileName,  
    char * pathBuf )
```

Извлечь из пути к/имени файла полный путь.

Аргументы

<i>fileName</i>	Путь к файлу или имя файла.
<i>pathBuf</i>	Буфер под путь.

Если путь к файлу изначально не полный, то он будет дополнен рабочим устройством и/или

рабочим каталогом.

19.23.1.8. extractNextDir()

```
const char* extractNextDir (  
    const char * path,  
    char * dirBuf )
```

Считать имя первого каталога из пути.

Аргументы

path Путь.

dirBuf Буфер под каталог.

Возвращает

Смещенный указатель, указывающий на часть буфера *path*, с которой начинается следующий каталог, после записанного в буфер.

Если в пути нет каталогов, то будет возвращено значение переменной *path*.

19.23.1.9. fileNamePreprocess()

```
const char* fileNamePreprocess (  
    const char * path,  
    char * devBuf,  
    DEV_HDR ** dh )
```

Предобработка имени файла/устройства.

Аргументы

path Путь к файлу.

devBuf Буфер, в который будет записано устройство.

dh Указатель на указатель на буфер, в который будет записан общий заголовок устройства.

Возвращает

Смещенный указатель, указывающий на часть буфера *path*, с которой начинается часть пути без имени устройства или *NULL*, если устройство не было найдено.

19.23.1.10. getFileName()

```
const char* getFileName (  
    const char * path )
```

Получить короткое имя файла (имя + расширение) из пути к нему.

Аргументы

path Путь к файлу.

Возвращает

Смещенный указатель, указывающий на часть буфера *path*, с которой начинается короткое имя файла.

19.23.1.11. getFileNameNonConst()

```
char* getFileNameNonConst (  
    char * path )
```

Получить короткое имя файла (имя + расширение) из пути к нему для не-константного указателя.

Аргументы

path Путь к файлу.

Возвращает

Смещенный указатель, указывающий на часть буфера *path*, с которой начинается короткое имя файла.

Аналогично функции `getFileName`, но возвращает не-константный указатель.

19.23.1.12. getWorkDevice()

```
const char* getWorkDevice (  
    void )
```

Получить имя текущего рабочего устройства.

Возвращает

Имя текущего рабочего устройства.

19.23.1.13. getWorkDirectory()

```
const char* getWorkDirectory (  
    const char * dev )
```

Получить рабочий каталог устройства.

Аргументы

dev Имя устройства.

Возвращает

Рабочий каталог устройства или NULL, при ошибке или если устройство не может иметь рабочего каталога.

19.23.1.14. setWorkDevice()

```
const char* setWorkDevice (  
    const char * path )
```

Установить устройство в качестве рабочего.

Аргументы

path Имя устройства (например, имя тома).

Возвращает

Установленное рабочее устройство.

19.24. Файл `fonts.h`

Высокоуровневые интерфейсы для работы с TTF-шрифтами.

Структуры данных

- struct *sTtfFont*

Определения типов

- typedef *sTtfFont* * *pTtfFont*
- typedef struct *ttfPrivateFontStruct* * *pTtfPrivateFontStruct*
- typedef struct *ttfPrivateFontStruct* *sTtfPrivateFontStruct*

Перечисления

- enum *eTtfFontOptions* {
ttf_NoGlyphSaving = 0x00 , *ttf_SaveGlyphs* = 0x01 , *ttf_NoPrerender* = 0x00 , *ttf_PrerenderASCII* = 0x10 ,
ttf_PrerenderDecNumbers = 0x20 , *ttf_NormalOrientation* = 0x000 , *ttf_RotatedOrientation* = 0x100 ,
ttf_KeepFontOpen = 0x0000 ,
ttf_CloseFontAfterPrerender = 0x1000 }

Прочие функции

- enum *eTtfTextOutputType* { *TTOT_None* = 0 , *TTOT_FontLoadingTime* = 0x1 , *TTOT_Errors* = 0x2 ,
TTOT_Debug = 0x8 }
- int *ttf_ConvertFontSize* (int originalSize, int originalDpi, int newDpi)
- void *ttf_SetTextOutputType* (*eTtfTextOutputType* outputType)

Инициализация библиотеки

Настройка режимов. Выделение и освобождение ресурсов.

- *STATUS* *ttf_FreeFont* (*pTtfFont* font)
- *pTtfFont* *ttf_LoadFont* (const char *path, unsigned int size, unsigned int color, *eTtfFontOptions* options)
- *STATUS* *ttf_SetDpi* (int dpiHorizontal, int dpiVertical)

Вывод текста

- *STATUS* *ttf_Print* (void *outputSurface, int x, int y, *pTtfFont* font, const char *text)
- *STATUS* *ttf_PrintRect* (void *outputSurface, const *pTextRect* dstRect, *eAlignType* align, *pTtfFont* font, const char *text)
- *STATUS* *ttf_PrintRectNoCheckBorder* (void *outputSurface, const *pTextRect* dstRect, *eAlignType* align, *pTtfFont* font, const char *text)
- *STATUS* *ttf_PrintRectUft16* (void *outputSurface, const *pTextRect* dstRect, *eAlignType* align, *pTtfFont* font, const unsigned short *text, unsigned int len)
- *STATUS* *ttf_PrintRectUft16NoCheckBorder* (void *outputSurface, const *pTextRect* dstRect, *eAlignType* align, *pTtfFont* font, const unsigned short *text, unsigned int len)
- *STATUS* *ttf_PrintUtf16* (void *outputSurface, int x, int y, *pTtfFont* font, const unsigned short *text, unsigned int len)

Получение характеристик текста

- int *ttf_GetTextPixelLength* (*pTtfFont* font, const char *text)
- int *ttf_GetTextPixelLengthUtf16* (*pTtfFont* font, const unsigned short *text, unsigned int len)
- *STATUS* *ttf_GetTextRect* (int *width, int *height, *pTtfFont* font, const char *text)
- *STATUS* *ttf_GetTextRectUtf16* (int *width, int *height, *pTtfFont* font, const unsigned short *text, unsigned int len)

19.24.1. Подробное описание

Библиотека функций поддержки **TrueType**-шрифтов.

Подключение:

```
#include <multimedia/fonts.h>
```

Makefile:

```
LIBRARIES += -l_font -l_freetype
```

См. также

Общее описание функций поддержки **ТТf** в главе *Поддержка шрифтов FreeType*.

19.24.2. Типы

19.24.2.1. pTtfFont

```
typedef sTtfFont* pTtfFont
```

Указатель на ttf-шрифт.

19.24.2.2. pTtfPrivateFontStruct

```
typedef struct ttfPrivateFontStruct* pTtfPrivateFontStruct
```

Указатель на инкапсулированные поля шрифта.

19.24.2.3. sTtfPrivateFontStruct

```
typedef struct ttfPrivateFontStruct sTtfPrivateFontStruct
```

Инкапсулированные поля шрифта.

19.24.3. Перечисления

19.24.3.1. eTtfFontOptions

```
enum eTtfFontOptions
```

Опции функционирования ttf-шрифта.

Наборы опций {NoGlyphSaving, SaveGlyphs}, {NoPrerender, PrerenderASCII, ttf_PrerenderDecNumbers}, {NormalOrientation, RotatedOrientation}, {KeepFontOpen, CloseFontAfterPrerender} допускают выбор только одной опции из каждого набора. Если из набора не будет выбрана ни одна из опций, то по-умолчанию будет использоваться одна из них (см. детальное описание опций). Опции из разных наборов можно совмещать через оператор '|'.

Элементы перечислений	
<code>ttf_NoGlyphSaving</code>	Не кешировать глифы в памяти, опция по-умолчанию.
<code>ttf_SaveGlyphs</code>	Каждый раз при рендере нового глифа он будет кеширован в память (пока только ASCII-глифы).
<code>ttf_NoPrerender</code>	Не пререндерить глифы при инициализации шрифта, опция по-умолчанию.
<code>ttf_PrerenderASCII</code>	Пререндерить все ASCII-глифы при инициализации шрифта.
<code>ttf_PrerenderDecNumbers</code>	Пререндерить все глифы десятичных цифр (0-9).
<code>ttf_NormalOrientation</code>	Глифы будут иметь обычную ориентацию, опция по-умолчанию.
<code>ttf_RotatedOrientation</code>	Глифы будут развернуты на 270 градусов (про координатные перерасчеты см. описание интерфейса).
<code>ttf_KeepFontOpen</code>	Держать шрифт открытым, это позволит рендерить символы "на лету", опция по-умолчанию.
<code>ttf_CloseFontAfterPrerender</code>	Закрывать шрифт после инициализации, при этом можно будет использовать только пререндеренные символы.

```

00039     {
00040     // Настройки сохранения глифов "на ходу"
00041     ttf_NoGlyphSaving = 0x00,
00042     ttf_SaveGlyphs = 0x01,
00043
00044     // Настройки пререндера глифов
00045     ttf_NoPrerender = 0x00,
00046     ttf_PrerenderASCII = 0x10,
00047     ttf_PrerenderDecNumbers = 0x20,
00048
00049     // Ориентация глифов
00050     ttf_NormalOrientation = 0x000,
00051     ttf_RotatedOrientation = 0x100,
00052
00053     // Нужно ли держать шрифт "открытым", то есть нужна ли возможность
00054     // выводить любые символы на лету, или пререндера достаточно
00054     ttf_KeepFontOpen = 0x0000,
00055     ttf_CloseFontAfterPrerender = 0x1000,
00056
00057 } eTtfFontOptions;

```

19.24.3.2. `eTtfTextOutputType`

enum `eTtfTextOutputType`

Варианты текстового вывода от библиотеки. Варианты могут комбинироваться.

Элементы перечислений	
TTOT_None	Текстовый вывод отсутствует, опция по-умолчанию.
TTOT_FontLoadingTime	Выводить сообщения вида "Font "Font.ttf" (color = 0xFFFFFFFF, size = 90) was loaded in 1313 ms".
TTOT_Errors	Выводить сообщения при ошибках.
TTOT_Debug	Выводить вспомогательный вывод для отладки библиотеки.

```

00282         {
00283     TTOT_None = 0,
00284     TTOT_FontLoadingTime = 0x1,
00285     TTOT_Errors = 0x2,
00286     TTOT_Debug = 0x8,
00287 } eTtfTextOutputType;
    
```

19.24.4. Функции

19.24.4.1. ttf_ConvertFontSize()

```

int ttf_ConvertFontSize (
    int originalSize,
    int originalDpi,
    int newDpi )
    
```

Преобразовать размер шрифта (в типографских пунктах) для различных dpi.

Аргументы	
<i>originalSize</i>	Исходный размер шрифта в типографских пунктах.
<i>originalDpi</i>	Исходный DPI .
<i>newDpi</i>	Новый DPI .

Возвращает

Размер шрифта (в типографских пунктах) для нового **DPI** или -1, в случае ошибки.

19.24.4.2. ttf_FreeFont()

```

STATUS ttf_FreeFont (
    pTtfFont font )
    
```

Освободить ресурсы шрифта.

Аргументы

font Шрифт для освобождения.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

19.24.4.3. ttf_GetTextPixelLength()

```
int ttf_GetTextPixelLength (  
    pTtfFont font,  
    const char * text )
```

Получить ширину выводимой ASCII-надписи для шрифта нормальной ориентации или высоту надписи для шрифта повернутой ориентации.

Аргументы

font Указатель на шрифт.

text Указатель на ASCII-буфер.

Возвращает

Ширина или высота (в зависимости от ориентации шрифта) в пикселях или -1 при ошибке.

19.24.4.4. ttf_GetTextPixelLengthUtf16()

```
int ttf_GetTextPixelLengthUtf16 (  
    pTtfFont font,  
    const unsigned short * text,  
    unsigned int len )
```

Получить ширину выводимой UTF-16-надписи для шрифта нормальной ориентации или высоту надписи для шрифта повернутой ориентации.

Аргументы

font Указатель на шрифт.

text Указатель на UTF-16 буфер.

len Длина UTF-16 буфера.

Возвращает

Ширина или высота (в зависимости от ориентации шрифта) в пикселях или -1 при ошибке.

19.24.4.5. ttf_GetTextRect()

```
STATUS ttf_GetTextRect (  
    int * width,  
    int * height,  
    pTtfFont font,  
    const char * text )
```

Получить высоту и ширину ASCII-надписи.

Аргументы		
out	<i>width</i>	Указатель на буфер, куда будет записана ширина надписи.
out	<i>height</i>	Указатель на буфер, куда будет записана высота шрифта (не надписи).
	<i>font</i>	Указатель на шрифт.
	<i>text</i>	Указатель на ASCII-буфер.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

19.24.4.6. ttf_GetTextRectUtf16()

```
STATUS ttf_GetTextRectUtf16 (  
    int * width,  
    int * height,  
    pTtfFont font,  
    const unsigned short * text,  
    unsigned int len )
```

Получить высоту и ширину UTF-16-надписи.

Аргументы		
out	<i>width</i>	Указатель на буфер, куда будет записана ширина надписи.
out	<i>height</i>	Указатель на буфер, куда будет записана высота шрифта (не надписи).
	<i>font</i>	Указатель на шрифт.
	<i>text</i>	Указатель на UTF-16 буфер.
	<i>len</i>	Длина UTF-16 буфера.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

19.24.4.7. ttf_LoadFont()

```
pTtfFont ttf_LoadFont (
    const char * path,
    unsigned int size,
    unsigned int color,
    eTtfFontOptions options )
```

Загрузить ttf-шрифт.

Аргументы	
<i>path</i>	Путь к ttf-файлу (настоящему или виртуальному).
<i>size</i>	Высота шрифта в пунктах.
<i>color</i>	Цвет шрифта в HTML-формате.
<i>options</i>	Комбинация перечислений <i>eTtfFontOptions</i> , можно использовать несколько перечислений через оператор ' '.

Возвращает

Указатель на загруженный шрифт или *NULL* при ошибке.

19.24.4.8. ttf_Print()

```
STATUS ttf_Print (
    void * outputSurface,
    int x,
    int y,
    pTtfFont font,
    const char * text )
```

Функция вывода ASCII-текста, аналог textOut.

Аргументы	
<i>outputSurface</i>	Поверхность, на которую осуществляется вывод. Может отличаться в зависимости от используемой графической библиотеки. Поверхность будет использована при копировании символа с помощью <i>fntGlyphCopy()</i> и должна иметь тот же тип данных.
<i>x,y</i>	Позиция для вывода на сурфейсе.
<i>font</i>	Указатель на шрифт для вывода.

Продолжение на следующей странице

Аргументы (Продолжение.)	
<i>text</i>	Указатель на выводимый текстовый буфер.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

Границы вывода не проверяются, выход за границы экрана/буфера будет предотвращен функциями графической подсистемы.

19.24.4.9. `ttf_PrintRect()`

```
STATUS ttf_PrintRect (
    void * outputSurface,
    const pTextRect dstRect,
    eAlignType align,
    pTtfFont font,
    const char * text )
```

Вывести ASCII-текст в прямоугольник с проверкой границ, аналог `drawUtf16Text`.

Аргументы	
<i>outputSurface</i>	Поверхность, на которую осуществляется вывод. Может отличаться в зависимости от используемой графической библиотеки. Поверхность будет использована при копировании символа с помощью <code>fontGlyphCopy()</code> и должна иметь тот же тип данных.
<i>dstRect</i>	Координатный прямоугольник для вывода.
<i>align</i>	Выравнивание текста в прямоугольнике.
<i>font</i>	Указатель на шрифт для вывода.
<i>text</i>	Указатель на выводимый текстовый буфер.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

Границы вывода проверяются, символы, выходящие за границы сурфейса или прямоугольника вывода НЕ будут выведены.

19.24.4.10. `ttf_PrintRectNoCheckBorder()`

```
STATUS ttf_PrintRectNoCheckBorder (
    void * outputSurface,
    const pTextRect dstRect,
    eAlignType align,
    pTtfFont font,
    const char * text )
```

Вывести ASCII-текст в прямоугольник без проверки границ, аналог `drawUtf16TextNoCheckBorder`.

Аргументы	
<i>outputSurface</i>	Поверхность, на которую осуществляется вывод. Может отличаться в зависимости от используемой графической библиотеки. Поверхность будет использована при копировании символа с помощью <code>fntGlyphCopy()</code> и должна иметь тот же тип данных.
<i>dstRect</i>	Координатный прямоугольник для вывода.
<i>align</i>	Выравнивание текста в прямоугольнике.
<i>font</i>	Указатель на шрифт для вывода.
<i>text</i>	Указатель на выводимый текстовый буфер.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

Границы вывода не проверяются, символы могут выйти за границы прямоугольника, однако выход за границы сурфейса будет предотвращен функциями графической подсистемы.

19.24.4.11. `tft_PrintRectUft16()`

```
STATUS tft_PrintRectUft16 (
    void * outputSurface,
    const pTextRect dstRect,
    eAlignType align,
    pTtfFont font,
    const unsigned short * text,
    unsigned int len )
```

Вывести UTF-16-текст в прямоугольник с проверкой границ, аналог `drawUtf16Text`.

Аргументы	
<i>outputSurface</i>	Поверхность, на которую осуществляется вывод. Может отличаться в зависимости от используемой графической библиотеки. Поверхность будет использована при копировании символа с помощью <code>fntGlyphCopy()</code> и должна иметь тот же тип данных.
<i>dstRect</i>	Координатный прямоугольник для вывода.
<i>align</i>	Выравнивание текста в прямоугольнике.
<i>font</i>	Указатель на шрифт для вывода.
<i>text</i>	Указатель на выводимый UTF-16 буфер.
<i>len</i>	Длина выводимого UTF-16 буфера.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

Границы вывода проверяются, символы, выходящие за границы сурфейса или прямоугольника вывода НЕ будут выведены.

19.24.4.12. `ttf_PrintRectUft16NoCheckBorder()`

```
STATUS ttf_PrintRectUft16NoCheckBorder (
    void * outputSurface,
    const pTextRect dstRect,
    eAlignType align,
    pTtfFont font,
    const unsigned short * text,
    unsigned int len )
```

Вывести UTF-16-текст в прямоугольник без проверки границ, аналог `drawUtf16TextNoCheckBorder`.

Аргументы	
<i>outputSurface</i>	Поверхность, на которую осуществляется вывод. Может отличаться в зависимости от используемой графической библиотеки. Поверхность будет использована при копировании символа с помощью <code>fntGlyphCopy()</code> и должна иметь тот же тип данных.
<i>dstRect</i>	Координатный прямоугольник для вывода.
<i>align</i>	Выравнивание текста в прямоугольнике.
<i>font</i>	Указатель на шрифт для вывода.
<i>text</i>	Указатель на выводимый UTF-16 буфер.
<i>len</i>	Длина выводимого UTF-16 буфера.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

Границы вывода не проверяются, символы могут выйти за границы прямоугольника, однако выход за границы сурфейса будет предотвращен функциями графической подсистемы.

19.24.4.13. `ttf_PrintUtf16()`

```
STATUS ttf_PrintUtf16 (
    void * outputSurface,
    int x,
    int y,
    pTtfFont font,
    const unsigned short * text,
    unsigned int len )
```

Вывести UTF-16-текст, аналог `textOut`.

Аргументы	
<i>outputSurface</i>	Поверхность, на которую осуществляется вывод. Может отличаться в зависимости от используемой графической библиотеки. Поверхность будет использована при копировании символа с помощью <code>fntGlyphCopy()</code> и должна иметь тот же тип данных.
<i>x,y</i>	Позиция для вывода на сурфейсе.
<i>font</i>	Указатель на шрифт для вывода.
<i>text</i>	Указатель на выводимый UTF-16 буфер.
<i>len</i>	Длина выводимого UTF-16 буфера.

Возвращает

OK в случае успеха, *ERROR* в противном случае.

Границы вывода не проверяются, выход за границы экрана/буфера будет предотвращен функциями графической подсистемы.

19.24.4.14. `ttf_SetDpi()`

```
STATUS ttf_SetDpi (  
    int dpiHorizontal,  
    int dpiVerical )
```

Установить **DPI**, который будет использоваться при выводе шрифтов.

Значение **DPI** по умолчанию - **96**.

Аргументы	
<i>dpiHorizontal,dpiVerical</i>	Значения DPI по вертикальной и горизонтальной оси.

Возвращает

OK при успехе, *ERROR*, если выполнение функции не разрешено.



Данную функцию разрешено использовать только перед началом взаимодействия со шрифтами; после загрузки какого-либо шрифта данная функция не будет выполняться.

19.24.4.15. `ttf_SetTextOutputType()`

```
void ttf_SetTextOutputType (  
    eTtfTextOutputType outputType )
```

Установить текстовый вывод библиотеки.

Аргументы

outputType Параметры текстового вывода.

19.25. Файл `fontsdefines.h`

Различные общие структуры, перечисления и дефайны для шрифтов.

Структуры данных

- struct `textRect`
Структура прямоугольника.

Макросы

- #define `ARRAY_INDEX_TO_SYMBOL(x)` `((x) + '')`
- #define `ASCII_PRINTED_SYMBOLS_AMOUNT` `('~' - ' ' + 1)`
- #define `SYMBOL_CODE_TO_ARRAY_INDEX(x)` `((x) - '')`
- #define `SYMBOL_IS_PRINTED_ASCII(x)` `((x) >= ' ' && (x) <= '~')`

Определения типов

- typedef `alignType` `eAlignType`
- typedef `sTextRect` * `pTextRect`
@details Указатель на прямоугольник, добавлен для совместимости и улучшения читаемости.
- typedef `textRect` `sTextRect`

Перечисления

- enum `alignType` {
`noAlign` = 0 , `alignHLeft` = 0x00000001 , `alignHCenter` = 0x00000002 , `alignHRight` = 0x00000004 ,
`alignVTop` = 0x00000008 , `alignVMiddle` = 0x00000010 , `alignVBottom` = 0x00000020 }
Опции выравнивания шрифта при выводе в прямоугольник.

19.25.1. Подробное описание

Предполагается, что данный файл будет подключаться во все шрифтовые интерфейсы (ttf шрифты, спрайтовые шрифты, может еще куда-нибудь).

19.25.2. Макросы

19.25.2.1. `ARRAY_INDEX_TO_SYMBOL`

```
#define ARRAY_INDEX_TO_SYMBOL(  
    x) ((x) + '')
```

Перевод индекса массива (например 0) в символ (' ').

19.25.2.2. `ASCII_PRINTED_SYMBOLS_AMOUNT`

```
#define ASCII_PRINTED_SYMBOLS_AMOUNT ('~' - ' ' + 1)
```

Количество ASCII-символов, которое можно вывести на экран.

19.25.2.3. `SYMBOL_CODE_TO_ARRAY_INDEX`

```
#define SYMBOL_CODE_TO_ARRAY_INDEX(  
    x) ((x) - '')
```

Перевод символа (например ' ') в индекс массива (0).

19.25.2.4. SYMBOL_IS_PRINTED_ASCII

```
#define SYMBOL_IS_PRINTED_ASCII(  
    x) ((x) >= ' ' && (x) <= '~')
```

Проверка на то, что символ - выводимый и из ascii.

19.25.3. Типы

19.25.3.1. eAlignType

```
typedef eAlignType eAlignType
```

Тип, добавленный для совместимости и улучшения читаемости.

19.25.3.2. pTextRect

```
typedef sTextRect* pTextRect
```

19.25.3.3. sTextRect

```
typedef textRect sTextRect
```

Переименование типа, добавлено для совместимости и улучшения читаемости.

19.25.4. Перечисления

19.25.4.1. alignType

```
enum alignType
```

Наборы опций {alignHLeft, alignHCenter, alignHRight} и {alignVTop, alignVMiddle, alignVBottom} допускают выбор только одной опции из каждого набора. Если из набора не будет выбрана ни одна из опций, то по-умолчанию будет использоваться опция noAlign. Опции из разных наборов можно совмещать через оператор '|'.

Элементы перечислений

noAlign	Не использовать выравнивание, текст будет выводиться в левый нижний угол прямоугольника, опция по-умолчанию.
alignHLeft	Выравнивать горизонтальное положение текста по левой стороне прямоугольника.
alignHCenter	Выравнивать горизонтальное положение текста по центру прямоугольника.
alignHRight	Выравнивать горизонтальное положение текста по правой стороне прямоугольника.

Продолжение на следующей странице

Элементы перечислений

alignVTop	Выравнивать вертикальное положение текста по верхней стороне прямоугольника.
alignVMiddle	Выравнивать вертикальное положение текста по центру прямоугольника.
alignVBottom	Выравнивать вертикальное положение текста по нижней стороне прямоугольника.

```
00053      {
00054      noAlign = 0,
00055      alignHLeft = 0x00000001,
00056      alignHCenter = 0x00000002,
00057      alignHRight = 0x00000004,
00058      alignVTop = 0x00000008,
00059      alignVMiddle = 0x00000010,
00060      alignVBottom = 0x00000020
00061 } alignType;
```

19.26. Файл gpio.h

Порты ввода/вывода (GPIO).

Макросы определения имён портов

Для выбора одного из пинов порта следует использовать сумму имени порта и номера пина. Например **P_B+4**.

- #define **P_A** (0)
- #define **P_B** (32 * 1)
- #define **P_C** (32 * 2)
- #define **P_D** (32 * 3)
- #define **P_E** (32 * 4)
- #define **P_F** (32 * 5)
- #define **P_G** (32 * 6)
- #define **P_H** (32 * 7)
- #define **P_I** (32 * 8)

Настройка внешних прерываний

- enum **eGpioInterruptMode** {
eintPositiveEdge = 0x0 , *eintNegativeEdge* = 0x1 , *eintHighLevel* = 0x2 , *eintLowLevel* = 0x3 ,
eintDoubleEdge = 0x4 }
- **STATUS gpioInterruptConnect** (unsigned int gpio, **eGpioInterruptMode** mode, int group, int priority, *usr_int_proc* routine, int parameter)
Подключение обработчика прерывания к линии GPIO.
- **STATUS gpioInterruptDisconnect** (unsigned int gpio)
Отключение обработки прерывания для заданной линии.

Управление выходным мультиплексором

- int **gpio_direction_input** (unsigned gpio)
Сконфигурировать пин порта как вход.
- int **gpio_direction_output** (unsigned gpio, int value)
Сконфигурировать пин порта как выход и выставить заданное значение.
- int **gpio_set_mux** (unsigned int gpio, int mux)
Сконфигурировать пин порта.

Управление линией ввода/вывода

- int **gpio_get_value** (unsigned gpio)
Получить значение на линии ввода.
- int **gpio_set_value** (unsigned gpio, int value)
Выставить значение на линии вывода.

Настройка подтягивающих резисторов

- int **gpio_pull_disable** (unsigned gpio)
Отпустить линию.
- int **gpio_pull_down** (unsigned gpio)
Подтянуть линию к минусу питания.
- int **gpio_pull_up** (unsigned gpio)
Подтянуть линию к плюсу питания.

Прочее

- unsigned **gpio_from_string** (const char *str, bool *ok)
Получить номер GPIO из строки.

19.26.1. Подробное описание

Настройка портов **GPIO**, подключение линий ввода/вывода к аппаратным модулям процессора.

Подключение:

```
#include <gpio.h>
```

См. также

Общее описание работы с портами ввода/вывода в главе *GPIO – Порты ввода/вывода*.

19.26.2. Макросы

19.26.2.1. P_A

```
#define P_A (0)
```

Порт ввода/вывода **A**.

19.26.2.2. P_B

```
#define P_B (32 * 1)
```

Порт ввода/вывода **B**.

19.26.2.3. P_C

```
#define P_C (32 * 2)
```

Порт ввода/вывода **C**.

19.26.2.4. P_D

```
#define P_D (32 * 3)
```

Порт ввода/вывода **D**.

19.26.2.5. P_E

```
#define P_E (32 * 4)
```

Порт ввода/вывода **E**.

19.26.2.6. P_F

```
#define P_F (32 * 5)
```

Порт ввода/вывода **F**.

19.26.2.7. P_G

```
#define P_G (32 * 6)
```

Порт ввода/вывода **G**.

19.26.2.8. P_H

```
#define P_H (32 * 7)
```

Порт ввода/вывода **H**.

19.26.2.9. P_I

```
#define P_I (32 * 8)
```

Порт ввода/вывода **I**.

19.26.3. Перечисления

19.26.3.1. eGpioInterruptMode

enum *eGpioInterruptMode*

Элементы перечислений

eintPositiveEdge	Внешнее прерывание сработает по переднему фронту импульса.
eintNegativeEdge	Внешнее прерывание сработает по заднему фронту импульса.
eintHighLevel	Внешнее прерывание сработает по высокому уровню сигнала.
eintLowLevel	Внешнее прерывание сработает по низкому уровню сигнала.
eintDoubleEdge	Внешнее прерывание сработает по обоим фронтам сигнала.

```
00227     {
00228     eintPositiveEdge = 0x0,
00229     eintNegativeEdge = 0x1,
00230     eintHighLevel   = 0x2,
00231     eintLowLevel    = 0x3,
00232     eintDoubleEdge  = 0x4
00233 } eGpioInterruptMode;
```

19.26.4. Функции

19.26.4.1. gpio_direction_input()

```
int gpio_direction_input (
```

```
unsigned gpio )
```

Функция конфигурирует выбранный пин указанного порта как вход. Функция является обёрткой функции `gpio_set_mux()` и записывает **0** в мультиплексор пина.

Аргументы

`gpio` Имя и номер порта из группы *макросов* описания имён портов.

Возвращает

Всегда ОК.

19.26.4.2. `gpio_direction_output()`

```
int gpio_direction_output (  
    unsigned gpio,  
    int value )
```

Функция конфигурирует выбранный пин указанного порта как выход. Функция является обёрткой функции `gpio_set_mux()` и записывает **1** в мультиплексор пина. После чего сразу устанавливает указанное значение на выходе пина с помощью `gpio_set_value()`.

Аргументы

`gpio` Имя и номер порта из группы *макросов* описания имён портов.

`value` **0** устанавливает низкий уровень на выходе, **1** – высокий.

Возвращает

Всегда ОК.

19.26.4.3. `gpio_from_string()`

```
unsigned gpio_from_string (  
    const char * str,  
    bool * ok )
```

Функция получает номер **GPIO** из строк, содержащих имя порта и номер пина. Стока может быть записана без пробелов, как в документации на процессор, либо в стиле *MULTEX-ARM*:

```
bool ok;  
gpio = gpio_from_string ( "{PG1}{}", \&ok);  
gpio = gpio_from_string ( "{P_G + 1}{}", \&ok);
```

Аргументы

str Строка содержащая имя линии **GPIO**.

ok Проверка успешного преобразования. Если в строке не обнаружено название линии – возвращает *false*.

Возвращает

Номер **GPIO**.

19.26.4.4. gpio_get_value()

```
int gpio_get_value (  
    unsigned gpio )
```

Функция возвращает значение, соответствующее уровню сигнала на входе выбранного пина указанного порта. Выбранная линия должна быть сконфигурирована как вход с помощью *gpio_direction_input()*.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

Возвращает

0 если на входе присутствует низкий уровень, **1** – если высокий.

19.26.4.5. gpio_pull_disable()

```
int gpio_pull_disable (  
    unsigned gpio )
```

Функция отключает *подтягивающие* резисторы от выбранной линии заданного порта. Выбранная линия должна быть сконфигурирована как вход с помощью *gpio_direction_input()*.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

Возвращает

Всегда ОК.

19.26.4.6. gpio_pull_down()


```
int gpio_pull_down (
    unsigned gpio )
```

Функция *подтягивает* выбранную линию заданного порта к минусу питания. Выбранная линия должна быть сконфигурирована как вход с помощью *gpio_direction_input()*.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

Возвращает

Всегда ОК.

19.26.4.7. gpio_pull_up()

```
int gpio_pull_up (
    unsigned gpio )
```

Функция *подтягивает* выбранную линию заданного порта к плюсу питания. Выбранная линия должна быть сконфигурирована как вход с помощью *gpio_direction_input()*.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

Возвращает

Всегда ОК.

19.26.4.8. gpio_set_mux()

```
int gpio_set_mux (
    unsigned int gpio,
    int mux )
```

Установить значение мультиплексора указанного пина выбранного порта для подключения к аппаратным модулям процессора. Функция может использоваться так же для настройки пина как обычного **GPIO**.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

mux Записываемое значение мультиплексора — одно из значений, описанных в документации на используемый процессор.

Возвращает

Всегда ОК.

19.26.4.9. `gpio_set_value()`

```
int gpio_set_value (
    unsigned gpio,
    int value )
```

Функция позволяет установить заданное значение на выходе выбранного пина указанного порта. Выбранная линия должна быть сконфигурирована как выход с помощью `gpio_direction_output()`.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

value **0** устанавливает низкий уровень на выходе линии, **1** — высокий уровень.

Возвращает

Всегда ОК.

19.26.4.10. `gpioInterruptConnect()`

```
STATUS gpioInterruptConnect (
    unsigned int gpio,
    eGpioInterruptMode mode,
    int group,
    int priority,
    usr_int_proc routine,
    int parameter )
```

Функция подключает процедуру пользователя в качестве обработчика аппаратного прерывания линии ввода/вывода процессора с помощью функции `interruptConnect()`. В драйвере реализовано отдельное подключение процедур пользователя на каждую линию (аппаратно для всех линий вызывается один и тот же обработчик драйвера).

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

mode Режим генерации прерывания — выбор фронта сигнала, по которому будет сгенерировано прерывание.

group Группа, к которой относится прерывание. Рекомендуется выбирать значение из макросов соответствующей *группы*.

priority Приоритет прерывания внутри группы. Рекомендуется выбирать значение из макросов соответствующей *группы*.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>routine</i>	Процедура, которую требуется подключить в качестве обработчика.
<i>parameter</i>	Параметр пользовательской процедуры обработчика прерывания.

Возвращает

OK при успешном выполнении, иначе *ERROR*.

19.26.4.11. `gpioInterruptDisconnect()`

STATUS `gpioInterruptDisconnect (`
 `unsigned int gpio)`

Отключается только обработка данной линии в контроллере внешних прерываний. Само прерывание **IRQ** остаётся активным, так как в контроллере к нему могут быть подключены другие линии.

Аргументы

gpio Имя и номер порта из группы *макросов* описания имён портов.

Возвращает

OK при успешном выполнении, иначе *ERROR*.

19.27. Файл i2c.h

Интерфейс I2C.

Функции

- `bool i2cInit` (unsigned int n, unsigned int set, unsigned int clk)
Инициализация канала I2C.
- `STATUS i2cRead` (unsigned int n, unsigned int addr, void *data, unsigned int len)
Чтение данных по шине I2C в режиме ведущего устройства.
- `STATUS i2cRegisterRead` (unsigned int n, unsigned int devAddr, unsigned int regAddr, unsigned int regLen, void *data, unsigned int dataLen)
Чтение регистра адресуемого устройства на шине I2C в режиме ведущего устройства.
- `STATUS i2cRegisterWrite` (unsigned int n, unsigned int devAddr, unsigned int regAddr, unsigned int regLen, const void *data, unsigned int dataLen)
Запись в регистр адресуемого устройства на шине I2C в режиме ведущего устройства.
- `STATUS i2cWrite` (unsigned int n, unsigned int addr, const void *data, unsigned int len)
Отправка данных по шине I2C в режиме ведущего устройства.

Макросы выбора каналов I2C

- `#define I2C_0 0`
- `#define I2C_1 1`
- `#define I2C_2 2`
- `#define I2C_3 3`
- `#define I2C_4 4`

Макросы выбора набора выходных линий I2C

- `#define I2C_GPIO_ADDITIONAL 1`
- `#define I2C_GPIO_DEFAULT 0`

Макросы выбора частоты сигнала CLK модуля I2C

- `#define I2C_CLK_100_kHz 100000`
- `#define I2C_CLK_400_kHz 400000`
- `#define I2C_CLK_FAST I2C_CLK_400_kHz`
- `#define I2C_CLK_NORMAL I2C_CLK_100_kHz`

19.27.1. Подробное описание

Настройка аппаратного модуля I2C.

Подключение:

```
#include <i2c.h>
```

См. также

Общее описание работы с интерфейсом I2C в разделе [I2C](#).

19.27.2. Макросы

19.27.2.1. I2C_0

```
#define I2C_0 0
```

Индекс для **I2C0**.

19.27.2.2. I2C_1

```
#define I2C_1 1
```

Индекс для **I2C1**.

19.27.2.3. I2C_2

```
#define I2C_2 2
```

Индекс для **I2C2**.

19.27.2.4. I2C_3

```
#define I2C_3 3
```

Индекс для **I2C3**.

19.27.2.5. I2C_4

```
#define I2C_4 4
```

Индекс для **I2C4**.

19.27.2.6. I2C_CLK_100_kHz

```
#define I2C_CLK_100_kHz 100000
```

Частота сигнала CLK модуля I2C 100 кГц.

19.27.2.7. I2C_CLK_400_kHz

```
#define I2C_CLK_400_kHz 400000
```

Частота сигнала CLK модуля I2C 400 кГц.

19.27.2.8. I2C_CLK_FAST

```
#define I2C_CLK_FAST I2C_CLK_400_kHz
```

Высокая частота сигнала CLK.

19.27.2.9. I2C_CLK_NORMAL

```
#define I2C_CLK_NORMAL I2C_CLK_100_kHz
```

Нормальная частота сигнала CLK (значение по умолчанию).

19.27.2.10. I2C_GPIO_ADDITIONAL

```
#define I2C_GPIO_ADDITIONAL 1
```

Выбор дополнительного набора линий модуля I2C.

19.27.2.11. I2C_GPIO_DEFAULT

```
#define I2C_GPIO_DEFAULT 0
```

Выбор основного набора линий модуля I2C. В большинстве случаев это единственный набор линий.

19.27.3. Функции

19.27.3.1. i2cInit()

```
bool i2cInit (  
    unsigned int n,  
    unsigned int set,  
    unsigned int clk )
```

Настройка и запуск выбранного канала **I2C**.

Аргументы

<i>n</i>	Номер канала I2C . Для определения номера канала рекомендуется использовать один из <i>макросов</i> .
<i>set</i>	Набор линий ввода/вывода, подключаемых к аппаратному модулю I2C . Рекомендуется выбирать один из наборов, описанных в группе <i>макросов</i> . Название выбранного порта и номера линий, подключенных к модулю можно увидеть в отладочном выводе функции в консоли. Какие именно выходы используются на плате, следует выяснить, изучив схему этой платы.
<i>clk</i>	Частота работы шины I2C в герцах. Рекомендуется выбирать одну из частот, определённых в группе <i>макросов</i> , либо задать значение частоты самостоятельно. Реальная частота работы будет равна либо меньше заданной, если точное значение частоты выставить не возможно.

Возвращает

true если инициализация прошла успешно, модуль запущен, иначе *false*.

19.27.3.2. i2cRead()

```
STATUS i2cRead (  
    unsigned int n,  
    unsigned int addr,  
    void * data,  
    unsigned int len )
```

Чтение данных в буфер **data** в режиме ведущего устройства **MASTER**. Функция дожидается окончания приёма данных. Перед началом чтения нового массива данных функция ожидает окончания предыдущей транзакции.

Аргументы

<i>n</i>	Номер канала I2C . Для определения номера канала рекомендуется использовать один из <i>макросов</i> .
<i>addr</i>	Адрес устройства на шине данных I2C .
<i>data</i>	Получаемые данные.
<i>len</i>	Длина получаемых данных (количество байт).

Возвращает

STATUS

19.27.3.3. i2cRegisterRead()

```
STATUS i2cRegisterRead (  
    unsigned int n,  
    unsigned int devAddr,  
    unsigned int regAddr,  
    unsigned int regLen,  
    void * data,  
    unsigned int dataLen )
```

Копирование адреса во внутренний буфер драйвера и запуск обмена с ведомым в режиме ведущего устройства **MASTER**. Функция является надстройкой над *i2cWrite()* с изменением направления передачи данных после отправки адреса регистра ведомому устройству. Номер регистра передаётся как первый байт(ы) данных.

Аргументы

<i>n</i>	Номер канала I2C . Для определения номера канала рекомендуется использовать один из <i>макросов</i> .
<i>devAddr</i>	Адрес устройства на шине данных I2C .
<i>regAddr</i>	Номер регистра, в который записываются данные.
<i>regLen</i>	Длина адреса регистра.
<i>data</i>	Получаемые данные.
<i>dataLen</i>	Длина принимаемых данных (количество байт) без учёта адреса регистра.

Возвращает

OK Если транзакция прошла без ошибок, иначе **ERROR**.
STATUS

19.27.3.4. i2cRegisterWrite()

```
STATUS i2cRegisterWrite (  
    unsigned int n,  
    unsigned int devAddr,  
    unsigned int regAddr,  
    unsigned int regLen,  
    const void * data,  
    unsigned int dataLen )
```

Копирование данных во внутренний буфер драйвера и запуск отправки буфера в режиме ведущего устройства **MASTER**. Функция является надстройкой над *i2cWrite()*. Номер регистра передаётся как первый байт(ы) данных.

Аргументы	
<i>n</i>	Номер канала I2C . Для определения номера канала рекомендуется использовать один из <i>макросов</i> .
<i>devAddr</i>	Адрес устройства на шине данных I2C .
<i>regAddr</i>	Номер регистра, в который записываются данные.
<i>regLen</i>	Длина адреса регистра.
<i>data</i>	Передаваемые данные.
<i>dataLen</i>	Длина передаваемых данных (количество байт) без учёта адреса регистра.

Возвращает

OK Если пакет отправлен без ошибок, иначе *ERROR*.

19.27.3.5. i2cWrite()

```
STATUS i2cWrite (  
    unsigned int n,  
    unsigned int addr,  
    const void * data,  
    unsigned int len )
```

Копирование данных во внутренний буфер драйвера и запуск отправки буфера в режиме ведущего устройства **MASTER**. Функция не дожидается окончания отправки. Перед началом отправки нового массива данных функция ожидает окончания предыдущей транзакции.

Аргументы	
<i>n</i>	Номер канала I2C . Для определения номера канала рекомендуется использовать один из <i>макросов</i> .
<i>addr</i>	Адрес устройства на шине данных I2C .

Продолжение на следующей странице

Аргументы (Продолжение.)

data Передаваемые данные.

len Длина передаваемых данных (количество байт).

Возвращает

OK Если пакет отправлен без ошибок, иначе *ERROR*.

19.28. Файл inifiles.h

Методы работы с ini-файлами.

Структуры данных

- struct *iniBinaryArray*
Структура для сохранения массива бинарных данных в ini-файле.
- struct *iniCoords*
- struct *iniIntArray*
Структура для сохранения массива целых чисел в ini-файле.
- struct *iniRect*

Определения типов

- typedef T_INI_FILE * *INI_FILE*

Работа с ini-файлом

- *STATUS iniFileChangeName (INI_FILE IF, const char *newName)*
Изменить имя ini-файла.
- *STATUS iniFileClose (INI_FILE IF)*
Закреть ini-файл с сохранением изменений.
- *INI_FILE iniFileCreate (const char *ifName)*
Создать новый идентификатор ini-файла.
- *STATUS iniFileFlush (INI_FILE IF)*
Сохранить все изменения в ini-файле.
- void *iniFileFree (INI_FILE IF)*
Закреть ini-файл без сохранения изменений.
- *INI_FILE iniFileOpen (const char *ifName)*
Открыть и считать ini-файл.
- *STATUS iniFilePrint (INI_FILE IF)*
Печать файла.

Общие функции работы с содержимым файла

- *bool iniFileCheckIfItemExists (INI_FILE IF, const char *Section, const char *Name)*
Проверка наличия параметра.
- *bool iniFileCheckIfSectionExists (INI_FILE iniFileDesc, const char *sectionName)*
Проверка наличия секции в ini-файле.
- *STATUS iniFileDeleteItem (INI_FILE iniFileDesc, const char *sectionName, const char *itemName)*
- *STATUS iniFileDeleteSection (INI_FILE iniFileDesc, const char *sectionName)*
- *char * iniFileItemName (INI_FILE IF, const char *Section, int ItemNum)*
Поиск параметра по порядковому номеру.
- *STATUS iniFileRenameItem (INI_FILE iniFileDesc, const char *sectionName, const char *oldItemName, const char *newItemName)*
- *STATUS iniFileRenameSection (INI_FILE iniFileDesc, const char *oldSectionName, const char *newSectionName)*
- *char * iniFileSectionName (INI_FILE IF, int SectionNum)*
Поиск секции по порядковому номеру.

Целые числа

- int *iniFileReadInteger (INI_FILE IF, const char *Section, const char *Name, int defVal)*
Считать целое число из ini-файла.
- void *iniFileWriteInteger (INI_FILE IF, const char *Section, const char *Name, int Val)*
Записать целое число в ini-файл.

Целые числа в hex-формате

- int *iniFileReadHex* (*INI_FILE* IF, const char *Section, const char *Name, int defVal)
Считать целое число в формате HEX из ini-файла.
- void *iniFileWriteHex* (*INI_FILE* IF, const char *Section, const char *Name, int Val)
Записать целое число в формате HEX в ini-файл.

Логическое значение

- int *iniFileReadBoolean* (*INI_FILE* IF, const char *Section, const char *Name, int defVal)
Считать логическое значение из ini-файла.
- void *iniFileWriteBoolean* (*INI_FILE* IF, const char *Section, const char *Name, int Val)
Записать логическое значение в ini-файл.

Строки

- const char * *iniFileReadConstString* (*INI_FILE* IF, const char *Section, const char *Name, const char *defVal)
Считать текстовую строку из ini-файла.
- char * *iniFileReadString* (*INI_FILE* IF, const char *Section, const char *Name, const char *defVal)
Считать текстовую строку из ini-файла.
- void *iniFileWriteString* (*INI_FILE* IF, const char *Section, const char *Name, const char *Val)
Записать строку в ini-файл.

Прямоугольные области

- *iniRect* * *iniFileReadTextRect* (*INI_FILE* iniFile, const char *section, const char *name, const *iniRect* *defVal)
Считать данные о прямоугольной области.
- void *iniFileWriteTextRect* (*INI_FILE* iniFile, const char *section, const char *name, const *iniRect* *val)
Записать данные о прямоугольной области.

Пары координат

- *iniCoords* * *iniFileReadCoords* (*INI_FILE* iniFile, const char *section, const char *name, const *iniCoords* *defVal)
Считать пару координат.
- void *iniFileWriteCoords* (*INI_FILE* iniFile, const char *section, const char *name, const *iniCoords* *val)
Записать пару координат.

Массивы целых чисел

- *iniIntArray* * *iniFileReadIntArray* (*INI_FILE* iniFile, const char *section, const char *name, const *iniIntArray* *defVal)
Считать массив целых чисел.
- void *iniFileWriteIntArray* (*INI_FILE* iniFile, const char *section, const char *name, const *iniIntArray* *val)
Записать массив целых чисел.

Массивы бинарных данных.

Формат данных в ini-файле: #5:AB78223109

- *iniBinaryArray* * *iniFileReadBinaryArray* (*INI_FILE* iniFile, const char *section, const char *name, const *iniBinaryArray* *defVal)
Считать массив бинарных данных.
- void *iniFileWriteBinaryArray* (*INI_FILE* iniFile, const char *section, const char *name, const *iniBinaryArray* *val)
Записать массив бинарных данных.

19.28.1. Подробное описание

Файл содержит описание методов работы с **ini**-файлами в ОС *MULTEX-ARM*.

См. также

Общее описание работы с **ini**-файлами в главе *Работа с ini-файлами*.

19.28.2. Типы

19.28.2.1. INI_FILE

```
typedef T_INI_FILE* INI_FILE
```

Указатель на структуру **ini**-файла.

19.28.3. Функции

19.28.3.1. iniFileChangeName()

```
STATUS iniFileChangeName (  
    INI_FILE IF,  
    const char * newName )
```

Функция изменяет имя **ini**-файла. При закрытии он будет сохранен с новым именем. Старый файл при этом не будет удален и останется со своим изначальным именем и содержимым.

Аргументы

<i>IF</i>	Указатель на открытый ini -файла.
<i>newName</i>	Указатель на строку нового имени файла.

Возвращает

OK при успехе.
ERROR в случае ошибки.

19.28.3.2. iniFileCheckIfItemExists()

```
bool iniFileCheckIfItemExists (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name )
```

Функция проверять наличие параметра в **ini**-файле.

Аргументы

<i>IF</i>	Идентификатор ini -файла.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.

Возвращает

true, если параметр содержится в файле, иначе *false*.

19.28.3.3. iniFileCheckIfSectionExists()

```
bool iniFileCheckIfSectionExists (  
    INI_FILE iniFileDesc,  
    const char * sectionName )
```

Аргументы

<i>iniFileDesc</i>	Идентификатор ini -файла.
<i>sectionName</i>	Название секции, в которой расположен параметр.

Возвращает

true если секция существует, иначе *false*.

19.28.3.4. iniFileClose()

```
STATUS iniFileClose (  
    INI_FILE IF )
```

Функция закрывает **ini**-файл, освобождает все занятые ресурсы и сохраняет все изменения.

Аргументы

<i>IF</i>	Указатель на открытый ini -файл.
-----------	---

Возвращает

OK при успехе.
ERROR в случае ошибки.

19.28.3.5. iniFileCreate()

```
INI_FILE iniFileCreate (  
    const char * ifName )
```

Создать идентификатор **ini**-файла, но не открывать файл и не заполнять идентификатор данными.

Аргументы

ifName Указатель на строку имени **ini**-файла.

Возвращает

Указатель, который следует использовать как идентификатор **ini**-файла.



Даже при отсутствии файла лучше использовать функцию *iniFileOpen()*.

19.28.3.6. iniFileDeleteItem()

```
STATUS iniFileDeleteItem (  
    INI_FILE iniFileDesc,  
    const char * sectionName,  
    const char * itemName )
```

Удалить параметр из ini-файла.

Аргументы

iniFileDesc Идентификатор **ini**-файла.

sectionName Название секции, в которой расположен параметр.

itemName Название параметра.

Возвращает

OK, если параметр был успешно удален, *ERROR* иначе.

19.28.3.7. iniFileDeleteSection()

```
STATUS iniFileDeleteSection (  
    INI_FILE iniFileDesc,  
    const char * sectionName )
```

Удалить секцию и все её параметры из ini-файла.

Аргументы

<i>iniFileDesc</i>	Идентификатор ini -файла.
<i>sectionName</i>	Название секции.

Возвращает

OK, если секция была успешно удалена, *ERROR* иначе.

19.28.3.8. iniFileFlush()

```
STATUS iniFileFlush (  
    INI_FILE IF)
```

Сохранить все изменения в **ini**-файле. Файл будет сохранён в том же режиме обфускации, в котором он был открыт.

Аргументы

<i>IF</i>	Идентификатор ini -файла.
-----------	----------------------------------

Возвращает

OK при успехе.
ERROR при ошибках записи и т.п.

19.28.3.9. iniFileFree()

```
void iniFileFree (  
    INI_FILE IF)
```

Функция закрывает файл, освобождает все ресурсы, при этом ничего не сохраняет на диск.

Аргументы

<i>IF</i>	Указатель на открытый ini -файл.
-----------	---

19.28.3.10. iniFileName()

```
char* iniFileName (  
    INI_FILE IF,  
    const char * Section,  
    int ItemNum)
```

Функция позволяет получить имя параметра с номером **ItemNum** из секции **Section**. Нумерация

начинается с **0**.

Аргументы	
<i>IF</i>	Идентификатор ini -файла.
<i>Section</i>	Указатель на имя секции.
<i>ItemNum</i>	Порядковый номер параметра.

Возвращает

Указатель на строку, содержащую имя параметра.

19.28.3.11. iniFileOpen()

```
INI_FILE iniFileOpen (  
    const char * ifName )
```

Функция открывает имеющийся (либо создает новый) **ini**-файл.

Аргументы	
<i>ifName</i>	Указатель на строку имени ini -файла.

Возвращает

Указатель, который следует использовать как идентификатор **ini**-файла.



Сам файл не держится открытым, он закрывается после прочтения.

19.28.3.12. iniFilePrint()

```
STATUS iniFilePrint (  
    INI_FILE IF )
```

Распечатать **ini**-файл в **stdout**.

Аргументы	
<i>IF</i>	Идентификатор ini -файла.

Возвращает

ОК при успехе, ERROR иначе.

19.28.3.13. iniFileReadBinaryArray()

```
iniBinaryArray* iniFileReadBinaryArray (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniBinaryArray * defVal )
```

Функция считывает массив бинарных данных из **ini**-файла.

Аргументы

<i>iniFile</i>	Указатель на открытый ini -файл.
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>defVal</i>	Указатель на структуру, возвращаемую при неудачном поиске.

Возвращает

Указатель на структуру полученных данных *iniBinaryArray*.



Необходимо освободить полученную структуру после использования!

19.28.3.14. iniFileReadBoolean()

```
int iniFileReadBoolean (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    int defVal )
```

Функция считывает из **ini**-файла значение переменной как логическое значение.

Аргументы

<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>defVal</i>	Значение, возвращаемое при неудачном поиске.

Возвращает

Значение указанной переменной, либо **defVal**, если переменную найти не удалось.

19.28.3.15. iniFileReadConstString()

```
const char* iniFileReadConstString (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    const char * defVal )
```

Функция аналогична iniFileReadString ().

Возвращает

Константный указатель на строку символов, либо на **defVal**, если переменную найти не удалось.

19.28.3.16. iniFileReadCoords()

```
iniCoords* iniFileReadCoords (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniCoords * defVal )
```

Функция считывает пару координат из **ini**-файла.

Аргументы

<i>iniFile</i>	Указатель на открытый ini -файл.
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>defVal</i>	Указатель на структуру, возвращаемую при неудачном поиске.

Возвращает

Указатель на структуру полученных данных *iniCoords*.



Необходимо освободить полученную структуру после использования!

19.28.3.17. iniFileReadHex()

```
int iniFileReadHex (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    int defVal )
```

Функция считывает из **ini**-файла значение переменной как целое число в формате **HEX**.

Аргументы	
<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>defVal</i>	Значение, возвращаемое при неудачном поиске.

Возвращает

Значение указанной переменной, либо **defVal**, если переменную найти не удалось.

19.28.3.18. iniFileReadIntArray()

```
iniIntArray* iniFileReadIntArray (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniIntArray * defVal )
```

Функция считывает массив целых чисел из **ini**-файла.

Аргументы	
<i>iniFile</i>	Указатель на открытый ini -файл.
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>defVal</i>	Указатель на структуру, возвращаемую при неудачном поиске.

Возвращает

Указатель на структуру полученных данных *iniIntArray*.



Необходимо освободить полученную структуру после использования!

19.28.3.19. iniFileReadInteger()

```
int iniFileReadInteger (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    int defVal )
```

Функция считывает из **ini**-файла значение переменной как целое число.

Аргументы

<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>defVal</i>	Значение, возвращаемое при неудачном поиске.

Возвращает

Значение указанной переменной, либо **defVal**, если переменную найти не удалось.

19.28.3.20. iniFileReadString()

```
char* iniFileReadString (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    const char * defVal )
```

Функция считывает из **ini**-файла значение переменной как текстовую строку. Значение по указателю должно быть использовано/скопировано ДО закрытия **ini**-файла.



Необходимо освободить полученную строку после использования.

Аргументы

<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>defVal</i>	Указатель на строку, возвращаемую при неудачном поиске.

Возвращает

Указатель на строку символов, либо на **defVal**, если переменную найти не удалось.

19.28.3.21. iniFileReadTextRect()

```
iniRect* iniFileReadTextRect (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniRect * defVal )
```

Функция получает данные о прямоугольной области из **ini**-файла.

Аргументы

<i>iniFile</i>	Указатель на открытый ini -файл.
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>defVal</i>	Указатель на структуру, возвращаемую при неудачном поиске.

Возвращает

Указатель на структуру полученных данных *iniRect*.



Необходимо освободить полученную структуру после использования!

19.28.3.22. iniFileRenameItem()

```
STATUS iniFileRenameItem (  
    INI_FILE iniFileDesc,  
    const char * sectionName,  
    const char * oldItemName,  
    const char * newItemName )
```

Переименовать параметр в ини-файле.

Аргументы

<i>iniFileDesc</i>	Идентификатор ini -файла.
<i>sectionName</i>	Название секции, в которой расположен параметр.
<i>oldItemName</i>	Старое название параметра.
<i>newItemName</i>	Новое название параметра.

Возвращает

OK, если параметр был переименована, *ERROR* если параметр не был найден, если секция с *newSectionName* уже существует или в случае других ошибок.

19.28.3.23. iniFileRenameSection()

```
STATUS iniFileRenameSection (  
    INI_FILE iniFileDesc,  
    const char * oldSectionName,  
    const char * newSectionName )
```

Переименовать секцию в ини-файле.

Аргументы	
<i>iniFileDesc</i>	Идентификатор ini -файла.
<i>oldSectionName</i>	Старое название секции.
<i>newSectionName</i>	Новое название секции.

Возвращает

OK, если секция была переименована, *ERROR* если секция не была найдена, если секция с *newSectionName* уже существует или в случае других ошибок.

19.28.3.24. iniFileSectionName()

```
char* iniFileSectionName (  
    INI_FILE IF,  
    int SectionNum )
```

Функция позволяет получить имя секции с номером **SectionNum** в структуре *INI_FILE*. Нумерация секций начинается с **0**.

Аргументы	
<i>IF</i>	Идентификатор ini -файла.
<i>SectionNum</i>	Порядковый номер секции.

Возвращает

Указатель на строку, содержащую имя параметра.

19.28.3.25. iniFileWriteBinaryArray()

```
void iniFileWriteBinaryArray (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniBinaryArray * val )
```

Функция записывает массив бинарных данных в **ini**-файл.

Аргументы

<i>iniFile</i>	Указатель на открытый ini -файл.
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>val</i>	Указатель на записываемые данные.

19.28.3.26. iniFileWriteBoolean()

```
void iniFileWriteBoolean (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    int Val )
```

Функция записывает в **ini**-файл значение переменной как логическое.

Аргументы

<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>Val</i>	Значение, которое требуется записать.

19.28.3.27. iniFileWriteCoords()

```
void iniFileWriteCoords (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniCoords * val )
```

Функция записывает пару координат в **ini**-файл.

Аргументы

<i>iniFile</i>	Указатель на открытый ini -файл.
----------------	---

Продолжение на следующей странице

Аргументы (Продолжение.)	
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>val</i>	Указатель на записываемые данные.

19.28.3.28. iniFileWriteHex()

```
void iniFileWriteHex (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    int Val )
```

Функция записывает в **ini**-файл значение переменной как целое число в формате **HEX**.

Аргументы	
<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>Val</i>	Значение, которое требуется записать.

19.28.3.29. iniFileWriteIntArray()

```
void iniFileWriteIntArray (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniIntArray * val )
```

Функция записывает массив целых чисел в **ini**-файл.

Аргументы	
<i>iniFile</i>	Указатель на открытый ini -файл.
<i>section</i>	Указатель на имя секции.
<i>name</i>	Указатель на имя переменной.
<i>val</i>	Указатель на записываемые данные.

19.28.3.30. iniFileWriteInteger()


```
void iniFileWriteInteger (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    int Val )
```

Функция записывает в **ini**-файл значение переменной как целое число.

Аргументы	
<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>Val</i>	Значение, которое требуется записать.

19.28.3.31. iniFileWriteString()

```
void iniFileWriteString (  
    INI_FILE IF,  
    const char * Section,  
    const char * Name,  
    const char * Val )
```

Функция записывает в **ini**-файл значение переменной как строку символов.

Аргументы	
<i>IF</i>	Указатель на открытый ini -файл.
<i>Section</i>	Указатель на имя секции.
<i>Name</i>	Указатель на имя переменной.
<i>Val</i>	Указатель на строку, которую требуется записать.

19.28.3.32. iniFileWriteTextRect()

```
void iniFileWriteTextRect (  
    INI_FILE iniFile,  
    const char * section,  
    const char * name,  
    const iniRect * val )
```

Функция записывает данные о прямоугольной области в **ini**-файл.

Аргументы	
<i>iniFile</i>	Указатель на открытый ini -файл.

Продолжение на следующей странице

Аргументы (Продолжение.)

section Указатель на имя секции.

name Указатель на имя переменной.

val Указатель на записываемые данные.

19.29. Файл `inputstr.h`

Функции для обработки введенных строк и работы с управляющими клавишами.

Перечисления

- enum `CTL_KEY` {
`CTL_NONE = 0`, `CTL_RIGHT`, `CTL_LEFT`, `CTL_UP`,
`CTL_DWN`, `CTL_INS`, `CTL_DEL`, `CTL_HOME`,
`CTL_END`, `CTL_PGUP`, `CTL_PGDWN`, `ALT_B`,
`ALT_D`, `ALT_F`}

Функции

- `CTL_KEY` `getCtlKey` (void)

19.29.1. Перечисления

19.29.1.1. `CTL_KEY`

enum `CTL_KEY`

Управляющая клавиша.

Элементы перечислений

<code>CTL_NONE</code>	Клавиша не найдена.
<code>CTL_RIGHT</code>	Стрелка вправо.
<code>CTL_LEFT</code>	Стрелка влево.
<code>CTL_UP</code>	Стрелка вверх.
<code>CTL_DWN</code>	Стрелка вниз.
<code>CTL_INS</code>	Insert.
<code>CTL_DEL</code>	Delete.
<code>CTL_HOME</code>	Home.
<code>CTL_END</code>	End.
<code>CTL_PGUP</code>	Page up.
<code>CTL_PGDWN</code>	Page down.
<code>ALT_B</code>	Alt-B.
<code>ALT_D</code>	Alt-D.
<code>ALT_F</code>	Alt-F.

```
00021 {
00022     CTL_NONE = 0,
```

```
00023 CTL_RIGHT ,
00024 CTL_LEFT ,
00025 CTL_UP ,
00026 CTL_DWN ,
00027 CTL_INS ,
00028 CTL_DEL ,
00029 CTL_HOME ,
00030 CTL_END ,
00031 CTL_PGUP ,
00032 CTL_PGDOWN ,
00033 ALT_B ,
00034 ALT_D ,
00035 ALT_F ,
00036 } CTL_KEY ;
```

19.29.2. Функции

19.29.2.1. getCtlKey()

```
CTL_KEY getCtlKey (
    void )
```

Считать следующую управляющую клавишу.

Функцию следует вызывать после считывания ESC-клавиши.

Возвращает

Значение перечисления CTL_KEY.

19.30. Файл `intlib.h`

Методы управления прерываниями.

Макросы

- `#define IRQ_HANDLED` (1)
- `#define IRQ_NONE` (0)

Определения типов

- `typedef int(* usr_int_proc)` (int)
Процедурный тип обработчиков прерываний.

Функции

- `int initIntLib` (void)
Инициализация прерываний.
- `int intConnect` (int vector, `usr_int_proc` routine, int parameter)
Подключение обработчика прерывания (устаревший вариант).
- `int interruptConnect` (int vector, int group, int priority, `usr_int_proc` routine, int parameter)
Подключение обработчика прерывания с указанием приоритета.
- `void irq` ()
Просмотр подключенных прерываний.

Макросы распределения приоритетов прерываний

Подробнее о приоритетах прерываний см. *Приоритеты прерываний и задач.*

- `#define INTERRUPT_GROUP_MAJOR` 2
- `#define INTERRUPT_GROUP_MINOR` 3
- `#define INTERRUPT_GROUP_SYSTEM` 1
- `#define INTERRUPT_GROUP_TIME_CRITICAL` 0
- `#define INTERRUPT_PRIORITY_BASE` 3
Базовый приоритет прерываний.
- `#define INTERRUPT_PRIORITY_HIGHEST` 0
- `#define INTERRUPT_PRIORITY_LOWEST` 7

19.30.1. Подробное описание

Файл содержит объявления методов управления прерываниями.

См. также

Общее описание работы с прерываниями в разделе *Управление прерываниями.*

19.30.2. Макросы

19.30.2.1. `INTERRUPT_GROUP_MAJOR`

```
#define INTERRUPT_GROUP_MAJOR 2
```

Основная группа прерываний. Исходно все прерывания находятся в этой группе.

19.30.2.2. INTERRUPT_GROUP_MINOR

```
#define INTERRUPT_GROUP_MINOR 3
```

Группа для прерываний с большим временем выполнения обработчика.

19.30.2.3. INTERRUPT_GROUP_SYSTEM

```
#define INTERRUPT_GROUP_SYSTEM 1
```

Группа системных прерываний. Обычно в этой группе одно прерывание — системный таймер.

19.30.2.4. INTERRUPT_GROUP_TIME_CRITICAL

```
#define INTERRUPT_GROUP_TIME_CRITICAL 0
```

Группа прерываний, для которых критично время выполнения.



Не рекомендуется помещать в эту группу более, чем одно прерывание в проекте.

19.30.2.5. INTERRUPT_PRIORITY_BASE

```
#define INTERRUPT_PRIORITY_BASE 3
```

19.30.2.6. INTERRUPT_PRIORITY_HIGHEST

```
#define INTERRUPT_PRIORITY_HIGHEST 0
```

Наивысший приоритет прерываний.

19.30.2.7. INTERRUPT_PRIORITY_LOWEST

```
#define INTERRUPT_PRIORITY_LOWEST 7
```

Для лучшей читаемости кода рекомендуется задавать приоритет, как смещение относительно базового. Например, `INTERRUPT_PRIORITY_BASE + 1`.

Самый низкий приоритет прерываний.

19.30.2.8. IRQ_HANDLED

```
#define IRQ_HANDLED (1)
```

19.30.2.9. IRQ_NONE

```
#define IRQ_NONE (0)
```

19.30.3. Типы

19.30.3.1. `usr_int_proc`

```
typedef int(* usr_int_proc) (int)
```

Процедурный тип обработчиков прерываний.

19.30.4. Функции

19.30.4.1. `initIntLib()`

```
int initIntLib (  
    void )
```

Процедура инициализации таблицы дескрипторов прерываний.

19.30.4.2. `intConnect()`

```
int intConnect (  
    int vector,  
    usr_int_proc routine,  
    int parameter )
```

Усм. Функция устарела, поскольку не поддерживает приоритеты прерываний. Подключаемый прерывания будут иметь значения приоритета по умолчанию. В новых проектах следует использовать функцию `interruptConnect()`.

19.30.4.3. `interruptConnect()`

```
int interruptConnect (  
    int vector,  
    int group,  
    int priority,  
    usr_int_proc routine,  
    int parameter )
```

Функция подключает процедуру пользователя в качестве обработчика аппаратного прерывания. Подключаемая процедура должна иметь формат:

```
int routine(int parameter);
```

См. также

Подробнее о приоритетах прерываний в разделе [Приоритеты прерываний и задач](#).

Аргументы

<i>vector</i>	Номер аппаратного прерывания, к которому требуется подключить обработчик.
---------------	---

Продолжение на следующей странице

Аргументы (Продолжение.)	
<i>group</i>	Группа, к которой относится прерывание. Рекомендуется выбирать значение из макросов соответствующей <i>группы</i> .
<i>priority</i>	Приоритет прерывания внутри группы. Рекомендуется выбирать значение из макросов соответствующей <i>группы</i> .
<i>routine</i>	Процедура, которую требуется подключить в качестве обработчика.
<i>parameter</i>	Параметр, который будет передан обработчику.

Возвращает

OK при успешном выполнении.

ERROR при неверном значении вектора прерывания.

19.30.4.4. irq()

void irq ()

Вывод в консоль таблицы подключенных прерываний.

19.31. Файл `inttypes.h`

Расширения для работы с типами заданного размера.

Структуры данных

- struct `imaxdiv_t`
Возврат функции `imaxdiv`, результат деления.

Макросы форматных строк `printf`

Макросы, определяющие форматные строки для **printf-подобных** функций для типов с заданными размерами.

- #define `PRId16` "d"
- #define `PRId32` "d"
- #define `PRId64` "lld"
- #define `PRId8` "d"
- #define `PRIdFAST16` "d"
- #define `PRIdFAST32` "d"
- #define `PRIdFAST64` "lld"
- #define `PRIdFAST8` "d"
- #define `PRIdLEAST16` "d"
- #define `PRIdLEAST32` "d"
- #define `PRIdLEAST64` "lld"
- #define `PRIdLEAST8` "d"
- #define `PRIdMAX` "lld"
- #define `PRIdPTR` "d"
- #define `PRi16` "i"
- #define `PRi32` "i"
- #define `PRi64` "lli"
- #define `PRi8` "i"
- #define `PRiFAST16` "i"
- #define `PRiFAST32` "i"
- #define `PRiFAST64` "lli"
- #define `PRiFAST8` "i"
- #define `PRiLEAST16` "i"
- #define `PRiLEAST32` "i"
- #define `PRiLEAST64` "lli"
- #define `PRiLEAST8` "i"
- #define `PRiMAX` "lli"
- #define `PRiPTR` "i"
- #define `PRIo16` "o"
- #define `PRIo32` "o"
- #define `PRIo64` "llo"
- #define `PRIo8` "o"
- #define `PRIoFAST16` "o"
- #define `PRIoFAST32` "o"
- #define `PRIoFAST64` "llo"
- #define `PRIoFAST8` "o"
- #define `PRIoLEAST16` "o"
- #define `PRIoLEAST32` "o"
- #define `PRIoLEAST64` "llo"
- #define `PRIoLEAST8` "o"
- #define `PRIoMAX` "llo"
- #define `PRIoPTR` "o"
- #define `PRiu16` "u"
- #define `PRiu32` "u"
- #define `PRiu64` "llu"
- #define `PRiu8` "u"
- #define `PRiuFAST16` "u"

- #define *PRIuFAST32* "u"
- #define *PRIuFAST64* "llu"
- #define *PRIuFAST8* "u"
- #define *PRIuLEAST16* "u"
- #define *PRIuLEAST32* "u"
- #define *PRIuLEAST64* "llu"
- #define *PRIuLEAST8* "u"
- #define *PRIuMAX* "llu"
- #define *PRIuPTR* "u"
- #define *PRIx16* "x"
- #define *PRIx16* "X"
- #define *PRIx32* "x"
- #define *PRIx32* "X"
- #define *PRIx64* "llx"
- #define *PRIx64* "llX"
- #define *PRIx8* "x"
- #define *PRIx8* "X"
- #define *PRIxFAST16* "x"
- #define *PRIxFAST16* "X"
- #define *PRIxFAST32* "x"
- #define *PRIxFAST32* "X"
- #define *PRIxFAST64* "llx"
- #define *PRIxFAST64* "llX"
- #define *PRIxFAST8* "x"
- #define *PRIxFAST8* "X"
- #define *PRIxLEAST16* "x"
- #define *PRIxLEAST16* "X"
- #define *PRIxLEAST32* "x"
- #define *PRIxLEAST32* "X"
- #define *PRIxLEAST64* "llx"
- #define *PRIxLEAST64* "llX"
- #define *PRIxLEAST8* "x"
- #define *PRIxLEAST8* "X"
- #define *PRIxMAX* "llx"
- #define *PRIxMAX* "llX"
- #define *PRIxPTR* "x"
- #define *PRIxPTR* "X"

Макросы форматных строк scanf

Макросы, определяющие форматные строки для **scanf-подобных** функций для типов с заданными размерами.

- #define *SCNd16* "hd"
- #define *SCNd32* "d"
- #define *SCNd64* "lld"
- #define *SCNd8* "hhd"
- #define *SCNdFAST16* "d"
- #define *SCNdFAST32* "d"
- #define *SCNdFAST64* "lld"
- #define *SCNdFAST8* "hhd"
- #define *SCNdLEAST16* "hd"
- #define *SCNdLEAST32* "d"
- #define *SCNdLEAST64* "lld"
- #define *SCNdLEAST8* "hhd"
- #define *SCNdMAX* "lld"
- #define *SCNdPTR* "d"
- #define *SCNi16* "hi"
- #define *SCNi32* "i"
- #define *SCNi64* "lli"
- #define *SCNi8* "hhi"
- #define *SCNiFAST16* "i"

- #define SCNiFAST32 "i"
- #define SCNiFAST64 "lli"
- #define SCNiFAST8 "hhi"
- #define SCNiLEAST16 "hi"
- #define SCNiLEAST32 "i"
- #define SCNiLEAST64 "lli"
- #define SCNiLEAST8 "hhi"
- #define SCNiMAX "lli"
- #define SCNiPTR "i"
- #define SCNo16 "ho"
- #define SCNo32 "o"
- #define SCNo64 "llo"
- #define SCNo8 "hho"
- #define SCNoFAST16 "o"
- #define SCNoFAST32 "o"
- #define SCNoFAST64 "llo"
- #define SCNoFAST8 "hho"
- #define SCNoLEAST16 "ho"
- #define SCNoLEAST32 "o"
- #define SCNoLEAST64 "llo"
- #define SCNoLEAST8 "hho"
- #define SCNoMAX "llo"
- #define SCNoPTR "o"
- #define SCNu16 "hu"
- #define SCNu32 "u"
- #define SCNu64 "llu"
- #define SCNu8 "hhu"
- #define SCNuFAST16 "u"
- #define SCNuFAST32 "u"
- #define SCNuFAST64 "llu"
- #define SCNuFAST8 "hhu"
- #define SCNuLEAST16 "hu"
- #define SCNuLEAST32 "u"
- #define SCNuLEAST64 "llu"
- #define SCNuLEAST8 "hhu"
- #define SCNuMAX "llu"
- #define SCNuPTR "u"
- #define SCNx16 "hx"
- #define SCNx32 "x"
- #define SCNx64 "llx"
- #define SCNx8 "hxx"
- #define SCNxFAST16 "x"
- #define SCNxFAST32 "x"
- #define SCNxFAST64 "llx"
- #define SCNxFAST8 "hxx"
- #define SCNxLEAST16 "hx"
- #define SCNxLEAST32 "x"
- #define SCNxLEAST64 "llx"
- #define SCNxLEAST8 "hxx"
- #define SCNxMAX "llx"
- #define SCNxPTR "x"

Функции для работы с типами максимального размера

- *intmax_t imaxabs* (*intmax_t* i)
- *imaxdiv_t imaxdiv* (*intmax_t* numer, *intmax_t* denom)
- *intmax_t strtoumax* (const char *restrict nptr, char **restrict endptr, int base)
- *uintmax_t strtoumax* (const char *restrict nptr, char **restrict endptr, int base)

19.31.1. Подробное описание

См. стандарт C11 7.8.

См. также

[C11 standard 7.8.](#)

19.31.2. Макросы

19.31.2.1. PRId16

```
#define PRId16 "d"
```

19.31.2.2. PRId32

```
#define PRId32 "d"
```

19.31.2.3. PRId64

```
#define PRId64 "lld"
```

19.31.2.4. PRId8

```
#define PRId8 "d"
```

19.31.2.5. PRIdFAST16

```
#define PRIdFAST16 "d"
```

19.31.2.6. PRIdFAST32

```
#define PRIdFAST32 "d"
```

19.31.2.7. PRIdFAST64

```
#define PRIdFAST64 "lld"
```

19.31.2.8. PRIdFAST8

```
#define PRIdFAST8 "d"
```

19.31.2.9. PRIdLEAST16

```
#define PRIdLEAST16 "d"
```

19.31.2.10. PRIdLEAST32

```
#define PRIdLEAST32 "d"
```

19.31.2.11. PRIdLEAST64

```
#define PRIdLEAST64 "lld"
```

19.31.2.12. PRIdLEAST8

```
#define PRIdLEAST8 "d"
```

19.31.2.13. PRIdMAX

```
#define PRIdMAX "lld"
```

19.31.2.14. PRIdPTR

```
#define PRIdPTR "d"
```

19.31.2.15. PRIi16

```
#define PRIi16 "i"
```

19.31.2.16. PRIi32

```
#define PRIi32 "i"
```

19.31.2.17. PRIi64

```
#define PRIi64 "lli"
```

19.31.2.18. PRIi8

```
#define PRIi8 "i"
```

19.31.2.19. PRIiFAST16

```
#define PRIiFAST16 "i"
```

19.31.2.20. PRIiFAST32

```
#define PRIiFAST32 "i"
```

19.31.2.21. PRIiFAST64

```
#define PRIiFAST64 "li"
```

19.31.2.22. PRIiFAST8

```
#define PRIiFAST8 "i"
```

19.31.2.23. PRIiLEAST16

```
#define PRIiLEAST16 "i"
```

19.31.2.24. PRIiLEAST32

```
#define PRIiLEAST32 "i"
```

19.31.2.25. PRIiLEAST64

```
#define PRIiLEAST64 "li"
```

19.31.2.26. PRIiLEAST8

```
#define PRIiLEAST8 "i"
```

19.31.2.27. PRIiMAX

```
#define PRIiMAX "li"
```

19.31.2.28. PRIiPTR

```
#define PRIiPTR "i"
```

19.31.2.29. PRIo16

```
#define PRIo16 "o"
```

19.31.2.30. PRIo32

```
#define PRIo32 "o"
```

19.31.2.31. PRIo64

```
#define PRIo64 "llo"
```

19.31.2.32. PRIo8

```
#define PRIo8 "o"
```

19.31.2.33. PRIoFAST16

```
#define PRIoFAST16 "o"
```

19.31.2.34. PRIoFAST32

```
#define PRIoFAST32 "o"
```

19.31.2.35. PRIoFAST64

```
#define PRIoFAST64 "llo"
```

19.31.2.36. PRIoFAST8

```
#define PRIoFAST8 "o"
```

19.31.2.37. PRIoLEAST16

```
#define PRIoLEAST16 "o"
```

19.31.2.38. PRIoLEAST32

```
#define PRIoLEAST32 "o"
```

19.31.2.39. PRIoLEAST64

```
#define PRIoLEAST64 "llo"
```

19.31.2.40. PRIoLEAST8

```
#define PRIoLEAST8 "o"
```

19.31.2.41. PRIoMAX

```
#define PRIoMAX "llo"
```

19.31.2.42. PRIoPTR

```
#define PRIoPTR "o"
```

19.31.2.43. PRIu16

```
#define PRIu16 "u"
```

19.31.2.44. PRIu32

```
#define PRIu32 "u"
```

19.31.2.45. PRIu64

```
#define PRIu64 "llu"
```


19.31.2.46. PRIu8

```
#define PRIu8 "u"
```

19.31.2.47. PRIuFAST16

```
#define PRIuFAST16 "u"
```

19.31.2.48. PRIuFAST32

```
#define PRIuFAST32 "u"
```

19.31.2.49. PRIuFAST64

```
#define PRIuFAST64 "llu"
```

19.31.2.50. PRIuFAST8

```
#define PRIuFAST8 "u"
```

19.31.2.51. PRIuLEAST16

```
#define PRIuLEAST16 "u"
```

19.31.2.52. PRIuLEAST32

```
#define PRIuLEAST32 "u"
```

19.31.2.53. PRIuLEAST64

```
#define PRIuLEAST64 "llu"
```

19.31.2.54. PRIuLEAST8

```
#define PRIuLEAST8 "u"
```

19.31.2.55. PRIuMAX

```
#define PRIuMAX "llu"
```

19.31.2.56. PRIuPTR

```
#define PRIuPTR "u"
```

19.31.2.57. PRIx16

```
#define PRIx16 "x"
```

19.31.2.58. PRIX16

```
#define PRIX16 "X"
```

19.31.2.59. PRIx32

```
#define PRIx32 "x"
```

19.31.2.60. PRIX32

```
#define PRIX32 "X"
```

19.31.2.61. PRIx64

```
#define PRIx64 "llx"
```

19.31.2.62. PRIX64

```
#define PRIX64 "lX"
```

19.31.2.63. PRIx8

```
#define PRIx8 "x"
```

19.31.2.64. PRIX8

```
#define PRIX8 "X"
```

19.31.2.65. PRiXFAST16

```
#define PRiXFAST16 "x"
```

19.31.2.66. PRIXFAST16

```
#define PRIXFAST16 "X"
```

19.31.2.67. PRiXFAST32

```
#define PRiXFAST32 "x"
```

19.31.2.68. PRIXFAST32

```
#define PRIXFAST32 "X"
```

19.31.2.69. PRiXFAST64

```
#define PRiXFAST64 "llx"
```

19.31.2.70. PRIXFAST64

```
#define PRIXFAST64 "llX"
```

19.31.2.71. PRiXFAST8

```
#define PRiXFAST8 "x"
```

19.31.2.72. PRIXFAST8

```
#define PRIXFAST8 "X"
```

19.31.2.73. PRiXLEAST16

```
#define PRiXLEAST16 "x"
```

19.31.2.74. PRiXLEAST16

```
#define PRiXLEAST16 "X"
```

19.31.2.75. PRiXLEAST32

```
#define PRiXLEAST32 "x"
```

19.31.2.76. PRiXLEAST32

```
#define PRiXLEAST32 "X"
```

19.31.2.77. PRiXLEAST64

```
#define PRiXLEAST64 "llx"
```

19.31.2.78. PRiXLEAST64

```
#define PRiXLEAST64 "llX"
```

19.31.2.79. PRiXLEAST8

```
#define PRiXLEAST8 "x"
```

19.31.2.80. PRiXLEAST8

```
#define PRiXLEAST8 "X"
```

19.31.2.81. PRiXMAX

```
#define PRiXMAX "llx"
```

19.31.2.82. PRIXMAX

```
#define PRIXMAX "IIX"
```

19.31.2.83. PRIxPTR

```
#define PRIxPTR "x"
```

19.31.2.84. PRIXPTR

```
#define PRIXPTR "X"
```

19.31.2.85. SCNd16

```
#define SCNd16 "hd"
```

19.31.2.86. SCNd32

```
#define SCNd32 "d"
```

19.31.2.87. SCNd64

```
#define SCNd64 "lld"
```

19.31.2.88. SCNd8

```
#define SCNd8 "hhd"
```

19.31.2.89. SCNdFAST16

```
#define SCNdFAST16 "d"
```

19.31.2.90. SCNdFAST32

```
#define SCNdFAST32 "d"
```

19.31.2.91. SCNdFAST64

```
#define SCNdFAST64 "lld"
```

19.31.2.92. SCNdFAST8

```
#define SCNdFAST8 "hhd"
```

19.31.2.93. SCNdLEAST16

```
#define SCNdLEAST16 "hd"
```

19.31.2.94. SCNdLEAST32

```
#define SCNdLEAST32 "d"
```

19.31.2.95. SCNdLEAST64

```
#define SCNdLEAST64 "lld"
```

19.31.2.96. SCNdLEAST8

```
#define SCNdLEAST8 "hhd"
```

19.31.2.97. SCNdMAX

```
#define SCNdMAX "lld"
```

19.31.2.98. SCNdPTR

```
#define SCNdPTR "d"
```

19.31.2.99. SCNi16

```
#define SCNi16 "hi"
```

19.31.2.100. SCNi32

```
#define SCNi32 "i"
```

19.31.2.101. SCNi64

```
#define SCNi64 "li"
```

19.31.2.102. SCNi8

```
#define SCNi8 "hi"
```

19.31.2.103. SCNiFAST16

```
#define SCNiFAST16 "i"
```

19.31.2.104. SCNiFAST32

```
#define SCNiFAST32 "i"
```

19.31.2.105. SCNiFAST64

```
#define SCNiFAST64 "li"
```

19.31.2.106. SCNiFAST8

```
#define SCNiFAST8 "hi"
```

19.31.2.107. SCNiLEAST16

```
#define SCNiLEAST16 "hi"
```

19.31.2.108. SCNiLEAST32

```
#define SCNiLEAST32 "i"
```

19.31.2.109. SCNiLEAST64

```
#define SCNiLEAST64 "lli"
```

19.31.2.110. SCNiLEAST8

```
#define SCNiLEAST8 "hhi"
```

19.31.2.111. SCNiMAX

```
#define SCNiMAX "lli"
```

19.31.2.112. SCNiPTR

```
#define SCNiPTR "i"
```

19.31.2.113. SCNo16

```
#define SCNo16 "ho"
```

19.31.2.114. SCNo32

```
#define SCNo32 "o"
```

19.31.2.115. SCNo64

```
#define SCNo64 "llo"
```

19.31.2.116. SCNo8

```
#define SCNo8 "hho"
```

19.31.2.117. SCNoFAST16

```
#define SCNoFAST16 "o"
```


19.31.2.118. SCNoFAST32

```
#define SCNoFAST32 "o"
```

19.31.2.119. SCNoFAST64

```
#define SCNoFAST64 "llo"
```

19.31.2.120. SCNoFAST8

```
#define SCNoFAST8 "hho"
```

19.31.2.121. SCNoLEAST16

```
#define SCNoLEAST16 "ho"
```

19.31.2.122. SCNoLEAST32

```
#define SCNoLEAST32 "o"
```

19.31.2.123. SCNoLEAST64

```
#define SCNoLEAST64 "llo"
```

19.31.2.124. SCNoLEAST8

```
#define SCNoLEAST8 "hho"
```

19.31.2.125. SCNoMAX

```
#define SCNoMAX "llo"
```

19.31.2.126. SCNoPTR

```
#define SCNoPTR "o"
```

19.31.2.127. SCNu16

```
#define SCNu16 "hu"
```

19.31.2.128. SCNu32

```
#define SCNu32 "u"
```

19.31.2.129. SCNu64

```
#define SCNu64 "llu"
```

19.31.2.130. SCNu8

```
#define SCNu8 "hhu"
```

19.31.2.131. SCNuFAST16

```
#define SCNuFAST16 "u"
```

19.31.2.132. SCNuFAST32

```
#define SCNuFAST32 "u"
```

19.31.2.133. SCNuFAST64

```
#define SCNuFAST64 "llu"
```

19.31.2.134. SCNuFAST8

```
#define SCNuFAST8 "hhu"
```

19.31.2.135. SCNuLEAST16

```
#define SCNuLEAST16 "hu"
```

19.31.2.136. SCNuLEAST32

```
#define SCNuLEAST32 "u"
```

19.31.2.137. SCNuLEAST64

```
#define SCNuLEAST64 "llu"
```

19.31.2.138. SCNuLEAST8

```
#define SCNuLEAST8 "hhu"
```

19.31.2.139. SCNuMAX

```
#define SCNuMAX "llu"
```

19.31.2.140. SCNuPTR

```
#define SCNuPTR "u"
```

19.31.2.141. SCNx16

```
#define SCNx16 "hx"
```

19.31.2.142. SCNx32

```
#define SCNx32 "x"
```

19.31.2.143. SCNx64

```
#define SCNx64 "llx"
```

19.31.2.144. SCNx8

```
#define SCNx8 "hhx"
```

19.31.2.145. SCNxFAST16

```
#define SCNxFAST16 "x"
```

19.31.2.146. SCNxFAST32

```
#define SCNxFAST32 "x"
```

19.31.2.147. SCNxFAST64

```
#define SCNxFAST64 "lx"
```

19.31.2.148. SCNxFAST8

```
#define SCNxFAST8 "hhx"
```

19.31.2.149. SCNxLEAST16

```
#define SCNxLEAST16 "hx"
```

19.31.2.150. SCNxLEAST32

```
#define SCNxLEAST32 "x"
```

19.31.2.151. SCNxLEAST64

```
#define SCNxLEAST64 "lx"
```

19.31.2.152. SCNxLEAST8

```
#define SCNxLEAST8 "hhx"
```

19.31.2.153. SCNxMAX

```
#define SCNxMAX "lx"
```

19.31.2.154. SCNxPTR

```
#define SCNxPTR "x"
```

19.31.3. Функции

19.31.3.1. `imaxabs()`

```
intmax_t imaxabs (  
    intmax_t i )
```

Вычислить абсолютное значение целого.

Аргументы

i Целое.

Возвращает

Абсолютное значение целого.

19.31.3.2. `imaxdiv()`

```
imaxdiv_t imaxdiv (  
    intmax_t numer,  
    intmax_t denom )
```

Разделить два целых числа и вычислить частное и остаток.

Аргументы

numer Делимое.

denom Делитель.

Возвращает

Структура из частного и остатка.

19.31.3.3. `strtoimax()`

```
intmax_t strtoimax (  
    const char *restrict nptr,  
    char **restrict endptr,  
    int base )
```

Преобразовать начальную часть строки в `intmax_t`-представление.

Аргументы

	<i>nptr</i>	Указатель на строку.
out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
	<i>base</i>	Система счисления числа в строке.

Возвращает

Преобразованное значение.

19.31.3.4. strtoumax()

```
uintmax_t strtoumax (  
    const char *restrict nptr,  
    char **restrict endptr,  
    int base )
```

Преобразовать начальную часть строки в *uintmax_t*-представление.

Аргументы

	<i>nptr</i>	Указатель на строку.
out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
	<i>base</i>	Система счисления числа в строке.

Возвращает

Преобразованное значение.

19.32. Файл `iolib.h`

Работа с базовой системой ввода / вывода.

Структуры данных

- struct `blk_dev`
- struct `device_header`
Общий заголовок устройства.
- struct `ffblk`
Блок информации для поиска файлов в каталоге.
- struct `file_fcb`
Блок управления файла.
- struct `seekblk`
Блок информации для функции `seek`.

Определения типов

- typedef void `DCB`
Блок управления устройства (специфичен для у-ва).
- typedef struct `device_header DEV_HDR`
Общий заголовок устройства.
- typedef struct `file_fcb FCB`
Блок управления файла.
- typedef struct `ffblk FFBLK`
Блок информации для поиска файлов в каталоге.
- typedef `FCB * P_FCB`
Указатель на блок управления файла.
- typedef struct `seekblk SEEKBLK`
Блок информации для функции `seek`.

Функции

- `STATUS cd` (const char *newWorkDir)
- `STATUS chkDsk` (const char *devName)
- `DEV_HDR * findDev` (const char *devName)
- `FCB * findFCB` (int fd)
- int `findFd` (const char *name)
- void `freeFCB` (`FCB *fcb`)
- int `getLongName` (int fd)
ioctl-запрос к файлу `FIOGETLNAME`.
- `STATUS getWorkDir` (char *pwd)
- `STATUS getWorkDir_s` (char *pwd, size_t bufSize)
- `STATUS ioGlobalStdSet` (int ioe, int fd)
Перенаправление глобального (стандартного) ввода/вывода.
- size_t `iosDevShow` (void)
Вывести в stdout список подключенных устройств.
- void `iosFdShow` (void)
Вывести в stdout список открытых файлов.
- `STATUS ioTaskStdSet` (`TASK_ID` task, int ioe, int fd)
Перенаправление ввода/вывода для отдельной задачи.
- `STATUS mkdir` (const char *name)
- int `reset` (const char *devName)
Сброс контроллера `IDE`.
- `STATUS rmdir` (const char *name)
- int `unloadVolume` (const char *devName)
Разгрузка съемного носителя.
- `STATUS upd` (void)

Переменные

- `DEV_HDR * sysDeviceList`
Список устройств.

БЛОЧНЫЕ УСТРОЙСТВА

Устройства прямого доступа.

- `#define BD_STD_SIGNATURE 0x4535FAEB`
- `typedef struct blk_dev BLK_DEV`
- `typedef void DEVICE`
- `STATUS mountVolume` (int fsDrvNum, const char *devName, int volume, int partNr, `BLK_DEV *driver`)
Монтирование тома.

БАЗОВЫЙ ВВОД/ВЫВОД

Флаги базовой системы ввода / вывода.

- `#define FA_ARCH 0x20`
- `#define FA_CAT 0x10`
- `#define FA_HIDE 0x02`
- `#define FA_LABEL 0x08`
- `#define FA_RO 0x01`
- `#define FA_SYSTEM 0x04`
- `#define FIOCD 37`
- `#define FIOCHKDSK 36`
- `#define FIODISKFORMAT 100`
- `#define FIOEOF 28`
- `#define FIOFILESIZE 27`
- `#define FIOFINDFIRST 30`
- `#define FIOFINDNEXT 31`
- `#define FIOFLUSH 33`
- `#define FIOFREESIZE 35`
- `#define FIOGETDT 42`
- `#define FIOGETLABEL 40`
- `#define FIOGETLNAME 105`
- `#define FIOGETTO 201`
- `#define FIOMKD 43`
- `#define FIORENAME 34`
- `#define FIORESETDRV 104`
- `#define FIORMD 44`
- `#define FIOSEEK 29`
- `#define FIOSETDT 41`
- `#define FIOSETTO 200`
- `#define FIOTELL 32`
- `#define FIOUNMOUNT 107`
- `#define FIOUPD 39`
- `#define FIOWRKDIR 38`
- `#define O_BINARY 8`
- `#define O_CAT 4`
- `#define O_CREAT 0x200`
- `#define O_FIXED 0x10`
- `#define O_RDONLY 1`
- `#define O_RDWR 0`
- `#define O_TRUNC 0x400`
- `#define O_WRONLY 2`

Функциональные типы базовых функций

Функциональные типы для 7 базовых функций ввода / вывода.

- typedef *STATUS*(* *iosDevCloseProc*) (*FCB* *fp)
- typedef *FCB* *(* *iosDevCreateProc*) (*DEV_HDR* *iosDev, const char *filename, int mode)
- typedef int(* *iosDevIoctlProc*) (*FCB* *fp, int requestCode, int arg)
- typedef *FCB* *(* *iosDevOpenProc*) (*DEV_HDR* *iosDev, const char *filename, int mode)
- typedef int(* *iosDevRdProc*) (*FCB* *fp, char *buffer, int nBytes)
- typedef *STATUS*(* *iosDevRemoveProc*) (*DEV_HDR* *iosDev, const char *filename)
- typedef int(* *iosDevWrProc*) (*FCB* *fp, const char *buffer, int nBytes)

Прототипы внутренних функций IOS

- *STATUS iosDevAdd* (*DCB* *iosDev, const char *name, int iosDrvNum)
Регистрация драйвера в системе.
- *STATUS iosDevRemove* (*DCB* *iosDev)
Удаление устройства по блоку управления.
- int *iosDrvInstall* (*iosDevCreateProc* createProc, *iosDevRemoveProc* removeProc, *iosDevOpenProc* openProc, *iosDevCloseProc* closeProc, *iosDevRdProc* readProc, *iosDevWrProc* writeProc, *iosDevIoctlProc* ioctlProc)
Инсталляция драйвера в системе.
- *STATUS removeDevice* (const char *devName)
Удаление устройства по имени.

Процедуры работы с устройствами / файлами

- int *close* (int fd)
Закрыть файл или устройство.
- int *creat* (const char *path, int flags)
Создать файл по имени.
- int *creat_empty* (const char *devn, int flags)
Создать пустой файл.
- int *ioctl* (int fd, int function, int arg)
ioctl-запрос к файлу.
- int *open* (const char *path, int flags)
Открыть файл или устройство.
- int *read* (int fd, void *buffer, int maxBytes)
Прочитать буфер из файла.
- *STATUS remove* (const char *path)
Удалить файл по имени.
- int *write* (int fd, const void *buffer, int maxBytes)
Записать буфер в файл.

Стандартные надстройки над ioctl

- int *flush* (int fd)
Сброс всех изменений файла.
- int *lseek* (int fd, int shift, int seektype)
Переместить указатель чтения / записи (алиас seek).
- int *seek* (int fd, int shift, int seektype)
Переместить указатель чтения / записи.
- int *tell* (int fd)
Положение указателя в файле.

Процедуры работы с каталогами

- *bool dirExists* (const char *dirName)
- *bool fileExists* (const char *fileName)
- *int fileSize* (const char *fileName)
- *STATUS findFirst* (const char *pathMask, *FFBLK* *ff)
Поиск первого файла в каталоге.
- *STATUS findNext* (*FFBLK* *ff)
Поиск следующего файла в каталоге.
- *STATUS formatVolume* (const char *devName)
- *int freeSpace* (const char *devName)
- *char ** getFilesInDir* (const char *dirName, int *filesAmount)
Получить двухмерный массив названий файлов.
- *char * getFullFileName* (int fd)
- *STATUS getVolumeLabel* (const char *devName, char buf[static 12])

19.32.1. Подробное описание

Файл содержит описания методов базовой системы ввода / вывода.

См. также

Общее описание системы ввода / вывода см. в главе *Базовая система ввода / вывода*.

19.32.2. Макросы

19.32.2.1. BD_STD_SIGNATURE

```
#define BD_STD_SIGNATURE 0x4535FAEB
```

Стандартный маркер структуры *BLK_DEV*.

19.32.2.2. FA_ARCH

```
#define FA_ARCH 0x20
```

19.32.2.3. FA_CAT

```
#define FA_CAT 0x10
```

19.32.2.4. FA_HIDE

```
#define FA_HIDE 0x02
```

19.32.2.5. FA_LABEL

```
#define FA_LABEL 0x08
```

19.32.2.6. FA_RO

```
#define FA_RO 0x01
```

19.32.2.7. FA_SYSTEM

```
#define FA_SYSTEM 0x04
```

19.32.2.8. FIOCD

```
#define FIOCD 37
```

19.32.2.9. FIOCHKDSK

```
#define FIOCHKDSK 36
```

19.32.2.10. FIODISKFORMAT

```
#define FIODISKFORMAT 100
```

19.32.2.11. FIOEOF

```
#define FIOEOF 28
```

19.32.2.12. FIOFILESIZE

```
#define FIOFILESIZE 27
```

19.32.2.13. FIOFINDFIRST

```
#define FIOFINDFIRST 30
```

19.32.2.14. FIOFINDNEXT

```
#define FIOFINDNEXT 31
```

19.32.2.15. FIOFLUSH

```
#define FIOFLUSH 33
```

19.32.2.16. FIOFREESIZE

```
#define FIOFREESIZE 35
```

19.32.2.17. FIOGETDT

```
#define FIOGETDT 42
```

19.32.2.18. FIOGETLABEL

```
#define FIOGETLABEL 40
```

19.32.2.19. FIOGETLNAME

```
#define FIOGETLNAME 105
```

19.32.2.20. FIOGETTO

```
#define FIOGETTO 201
```

19.32.2.21. FIOMKD

```
#define FIOMKD 43
```

19.32.2.22. FIORENAME

```
#define FIORENAME 34
```

19.32.2.23. FIORESETDRV

```
#define FIORESETDRV 104
```

19.32.2.24. FIORMD

```
#define FIORMD 44
```

19.32.2.25. FIOSEEK

```
#define FIOSEEK 29
```

19.32.2.26. FIOSETDT

```
#define FIOSETDT 41
```

19.32.2.27. FIOSETTO

```
#define FIOSETTO 200
```

19.32.2.28. FIOTELL

```
#define FIOTELL 32
```

19.32.2.29. FIOUNMOUNT

```
#define FIOUNMOUNT 107
```

19.32.2.30. FIOUPD

```
#define FIOUPD 39
```

19.32.2.31. FIOWRKDIR

```
#define FIOWRKDIR 38
```

19.32.2.32. O_BINARY

```
#define O_BINARY 8
```

19.32.2.33. O_CAT

```
#define O_CAT 4
```

19.32.2.34. O_CREAT

```
#define O_CREAT 0x200
```

19.32.2.35. O_FIXED

```
#define O_FIXED 0x10
```

19.32.2.36. O_RDONLY

```
#define O_RDONLY 1
```

19.32.2.37. O_RDWR

```
#define O_RDWR 0
```

19.32.2.38. O_TRUNC

```
#define O_TRUNC 0x400
```

19.32.2.39. O_WRONLY

```
#define O_WRONLY 2
```

19.32.3. Типы**19.32.3.1. BLK_DEV**

```
typedef struct blk_dev BLK_DEV
```

19.32.3.2. DCB

```
typedef void DCB
```

19.32.3.3. DEV_HDR

```
typedef struct device_header DEV_HDR
```

Структура общего заголовка устройства.

19.32.3.4. DEVICE

```
typedef void DEVICE
```

19.32.3.5. FCB

```
typedef struct file_fcb FCB
```

Структура блока управления файла.

19.32.3.6. FFBLK

```
typedef struct ffblk FFBLK
```

Структура данных блока поиска файлов в каталоге.

19.32.3.7. iosDevCloseProc

```
typedef STATUS(* iosDevCloseProc) (FCB *fp)
```

19.32.3.8. iosDevCreateProc

```
typedef FCB(* iosDevCreateProc) (DEV_HDR *iosDev, const char *filename, int mode)
```

19.32.3.9. iosDevIoctlProc

```
typedef int(* iosDevIoctlProc) (FCB *fp, int requestCode, int arg)
```

19.32.3.10. iosDevOpenProc

```
typedef FCB(* iosDevOpenProc) (DEV_HDR *iosDev, const char *filename, int mode)
```

19.32.3.11. iosDevRdProc

```
typedef int(* iosDevRdProc) (FCB *fp, char *buffer, int nBytes)
```

19.32.3.12. iosDevRemoveProc

```
typedef STATUS(* iosDevRemoveProc) (DEV_HDR *iosDev, const char *filename)
```

19.32.3.13. iosDevWrProc

```
typedef int(* iosDevWrProc) (FCB *fp, const char *buffer, int nBytes)
```

19.32.3.14. P_FCB

```
typedef FCB* P_FCB
```

19.32.3.15. SEEKBLK

```
typedef struct seekblk SEEKBLK
```

Структура данных блока информации функции **seek**.

19.32.4. Функции

19.32.4.1. cd()

```
STATUS cd (
    const char * newWorkDir )
```

Изменить рабочий каталог.

Аргументы	
<i>newWorkDir</i>	Путь к новому рабочему каталогу.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.2. chkDsk()

```
STATUS chkDsk (
    const char * devName )
```

Провести проверку устройства и вывести результаты проверки в stdout.

Фактически проверка будет корректно проводится только для подключенных файловых томов.

Аргументы

devName Идентификатор устройства.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.3. close()

```
int close (  
    int fd )
```

Функция закрывает ранее открытый файл или устройство по дескриптору.

Аргументы

fd Дескриптор открытого файла.

19.32.4.4. creat()

```
int creat (  
    const char * path,  
    int flags )
```

Функция создает файл по его имени.

Аргументы

path Имя создаваемого файла.

flags Флаги для создания, такие как *O_RDONLY*, *O_WRONLY*, *O_RDWR*, *O_CREAT*.

См. также

Полный перечень флагов см. в разделе [Флаги базовой системы ввода / вывода](#).

Возвращает

Дескриптор создаваемого файла, небольшое целое число. При неудаче вернет отрицательное значение.

19.32.4.5. creat_empty()

```
int creat_empty (  
    const char * devn,
```

```
int flags )
```

Функция создаёт дескриптор для пустого файла на устройстве.

Аргументы

devn Идентификатор устройства.

flags Флаги для создания, такие как *O_RDONLY*, *O_WRONLY*, *O_RDWR*, *O_CREAT*.

См. также

Полный перечень флагов см. в разделе [Флаги базовой системы ввода / вывода](#).

Возвращает

Дескриптор создаваемого файла, небольшое целое число. При неудаче вернет отрицательное значение.

19.32.4.6. dirExists()

```
bool dirExists (  
    const char * dirName )
```

Проверить наличие каталога на диске.

Аргументы

dirName Путь к каталогу.

Возвращает

true, если каталог существует, *false*, если нет.

19.32.4.7. fileExists()

```
bool fileExists (  
    const char * fileName )
```

Проверить наличие файла на диске.

Аргументы

fileName Путь к файлу.

Возвращает

true, если файл существует, *false*, если нет.

19.32.4.8. fileSize()

```
int fileSize (  
    const char * fileName )
```

Получить размер файла.

Аргументы

fileName Путь к файлу.

Возвращает

Размер файла или -1 в случае ошибки.

19.32.4.9. findDev()

```
DEV_HDR* findDev (  
    const char * devName )
```

Получить заголовок с блоком управления устройством по его имени.

Аргументы

devName Идентификатор устройства.

Возвращает

Указатель на структуру типа *DEV_HDR* или *NULL* при ошибке.

19.32.4.10. findFCB()

```
FCB* findFCB (  
    int fd )
```

Получить структуру блока управления файла по его дескриптору.

Аргументы

fd Дескриптор файла.

Возвращает

Указатель на структуру блока управления файла или NULL при ошибке.

19.32.4.11. findFd()

```
int findFd (
    const char * name )
```

Найти открытый дескриптор по имени файла/устройства.

Аргументы

<i>name</i>	Имя файла/устройства.
-------------	-----------------------

Возвращает

Дескриптор в виде положительного числа, либо -1, если дескриптор не был найден.

19.32.4.12. findFirst()

```
STATUS findFirst (
    const char * pathMask,
    FFBLK * ff)
```

Функция отыскивает первый файл в каталоге, удовлетворяющий заданным условиям.

Аргументы

<i>pathMask</i>	Маска для поиска нужных файлов.
-----------------	---------------------------------

<i>ff</i>	Указатель на вспомогательную структуру для поиска. Функция заносит в эту структуру данные для последующих поисковых запросов.
-----------	---

Возвращает

Если файл, удовлетворяющий условиям поиска, найден, то возвращает **0**, в другом случае возвращает ненулевое значение.

19.32.4.13. findNext()

```
STATUS findNext (
    FFBLK * ff)
```

Функция отыскивает следующий файл в каталоге, удовлетворяющий заданным условиям.

Аргументы

ff Указатель на вспомогательную структуру для поиска, заполненную функцией *findFirst()*.

Возвращает

Если очередной файл, удовлетворяющий условиям поиска, найден, то возвращает **0**, в другом случае возвращает ненулевое значение.

19.32.4.14. flush()

```
int flush (  
    int fd )
```

Функция сбрасывает все изменения для файла в памяти в файл на устройстве.

Аргументы

fd Дескриптор файла.

Возвращает

При успешном завершении функция возвращает 0 и ненулевое значение при неудаче.

19.32.4.15. formatVolume()

```
STATUS formatVolume (  
    const char * devName )
```

Отформатировать устройство, если это возможно.

Аргументы

devName Идентификатор устройства.

Возвращает

OK при успехе, *ERROR* при ошибке или если форматирование не возможно для устройства.

19.32.4.16. freeFCB()

```
void freeFCB (  
    FCB * fcf )
```

Освободить структуру блока управления файла.

Аргументы

<i>fcf</i>	Указатель на структуру блока управления файла.
------------	--

19.32.4.17. freeSpace()

```
int freeSpace (
    const char * devName )
```

Получить размер свободного места на устройстве, если это имеет смысл.

Аргументы

<i>devName</i>	Идентификатор устройства.
----------------	---------------------------

Возвращает

Свободное место в байтах в виде целого ≥ 0 при успехе, -1 при ошибке или если "свободное место" не имеет смысла для устройства.

19.32.4.18. getFilesInDir()

```
char** getFilesInDir (
    const char * dirName,
    int * filesAmount )
```

Функция получает двумерный массив названий файлов (не системных, не скрытых, не папок) из указанной папки. Массив аллоцируется из кучи, по окончании использования его необходимо освободить! Максимальное количество файлов - до 1000, если файлов больше, то они не будут выведены. *

Аргументы

	<i>dirName</i>	Имя каталога для поиска.
out	<i>filesAmount</i>	Указатель на выходную переменную. При успешной отработке функции в переменную будет записан размер итогового массива.

Возвращает

Двумерный массив строк размером, соответствующим значению переменной *filesAmount*, или NULL при ошибке или отсутствии файлов.



Для освобождения памяти следует по-очери освободить каждую строку в массиве, после чего освободить сам массив.

19.32.4.19. getFullFileName()

```
char* getFullFileName (  
    int fd )
```

Получить полный путь до файла по его дескриптору.

Аргументы

fd Дескриптор файла.

Возвращает

Выделенная в памяти строка с полным путем к файлу или *NULL* при ошибке.

19.32.4.20. getLongName()

```
int getLongName (  
    int fd )
```

ioctl-запрос к файлу с кодом запроса *FIOGETLNAME* без аргументов.

Аргументы

fd Дескриптор файла.

Возвращает

Определяется типом запроса. Если запрос не поддерживается драйвером, то возвращается -1.

19.32.4.21. getVolumeLabel()

```
STATUS getVolumeLabel (  
    const char * devName,  
    char buf[static 12] )
```

Получить метку тома для устройства, если она имеется.

Аргументы

devName Идентификатор устройства.

buf Буфер под метку размером минимум 12 байт.

Возвращает

OK при успехе, *ERROR* при ошибке или если устройство не может иметь идентификатор тома.

19.32.4.22. `getWorkDir()`

```
STATUS getWorkDir (  
    char * pwd )
```

Получить текущий рабочий каталог.

Аргументы

pwd Буфер, в который будет записан текущий диск:каталог.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.23. `getWorkDir_s()`

```
STATUS getWorkDir_s (  
    char * pwd,  
    size_t bufSize )
```

Получить текущий рабочий каталог (безопасный аналог `getWorkDir`).

Аргументы

pwd Буфер размера *bufSize*, в который будет записан текущий диск:каталог.

bufSize Размер буфера.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.24. `ioctl()`

```
int ioctl (  
    int fd,  
    int function,  
    int arg )
```

Функция выполняет **ioctl-запрос** к файлу

Аргументы

<i>fd</i>	Дескриптор файла.
<i>function</i>	Код запроса.
<i>arg</i>	Аргумент запроса.

Возвращает

Определяется типом запроса. Если запрос не поддерживается драйвером, то возвращается -1.

19.32.4.25. ioGlobalStdSet()

```
STATUS ioGlobalStdSet (  
    int ioe,  
    int fd )
```

Функция перенаправляет весь стандартный ввод/вывод на указанное устройство. Не меняет потоки ввода-вывода для задач с явно указанными потоками.

Аргументы

<i>ioe</i>	Поток: 0 = stdin , 1 = stdout , 2 = stderr .
<i>fd</i>	Дескриптор устройства, куда требуется перенаправить поток.

Возвращает

OK при успешном выполнении.
ERROR при ошибке в параметрах.

19.32.4.26. iosDevAdd()

```
STATUS iosDevAdd (  
    DCB * iosDev,  
    const char * name,  
    int iosDrvNum )
```

Функция регистрирует в системе драйвер устройства под заданным именем.

Аргументы

<i>iosDev</i>	Указатель на блок управления устройством.
<i>name</i>	Имя, под которым устройство будет доступно
<i>iosDrvNum</i>	Номер, под которым зарегистрировано устройство (результат возврата <i>iosDrvInstall</i>).

Возвращает

OK при успешном завершении.
ERROR при неудаче.

19.32.4.27. iosDevRemove()

```
STATUS iosDevRemove (  
    DCB * iosDev )
```

Функция удаляет из системы заданное устройство по его блоку управления.

Аргументы

<i>iosDev</i>	Блок управления удаляемым устройством.
---------------	--

Возвращает

OK при успешном завершении.
ERROR при неудаче.

19.32.4.28. iosDevShow()

```
size_t iosDevShow (  
    void )
```

Функция выводит список всех подключенных устройств в stdout.

Возвращает

Общее кол-во подключенных устройств.

19.32.4.29. iosDrvInstall()

```
int iosDrvInstall (  
    iosDevCreateProc createProc,  
    iosDevRemoveProc removeProc,  
    iosDevOpenProc openProc,  
    iosDevCloseProc closeProc,  
    iosDevRdProc readProc,  
    iosDevWrProc writeProc,  
    iosDevIoctlProc ioctlProc )
```

Функция устанавливает драйвер устройства, или файловую систему. Драйвер должен иметь семь функций:

- Функцию создания файла на устройстве *FCB* *createProc(*DEV_HDR* *iosDev, char *name, int mode).
- Функцию удаления файла на устройстве *STATUS* removeProc(*DEV_HDR* *iosDev, char *name).

- Функцию открытия файла *FCB* *openProc(*DEV_HDR* *iosDev, char *name, int mode).
- Функцию закрытия файла *STATUS* closeProc(*FCB* *fp).
- Функцию чтения данных int readProc(*FCB* *fp, char *buf, int count).
- Функцию записи данных int writeProc(*FCB* *fp, char *buf, int count).
- Функцию обработки специальных запросов int ioctlProc(*FCB* *fp, int requestCode, int arg).

Не все функции обязательны, так, например, для устройства типа консоли имя файла указывается пустым, а функции создания и удаления вовсе отсутствуют. Если вместо указателя на функцию указан **NULL**, это означает, что функция не поддерживается устройством.

Аргументы	
<i>createProc</i>	Функция вида <i>FCB</i> *createProc(<i>DEV_HDR</i> *iosDev, char *name, int mode).
<i>removeProc</i>	Функция вида <i>STATUS</i> removeProc(<i>DEV_HDR</i> *iosDev, char *name).
<i>openProc</i>	Функция вида <i>FCB</i> *openProc(<i>DEV_HDR</i> *iosDev, char *name, int mode).
<i>closeProc</i>	Функция вида <i>STATUS</i> closeProc(<i>FCB</i> *fp).
<i>readProc</i>	Функция вида int readProc(<i>FCB</i> *fp, char *buf, int count).
<i>writeProc</i>	Функция вида int writeProc(<i>FCB</i> *fp, char *buf, int count).
<i>ioctlProc</i>	Функция вида int ioctlProc(<i>FCB</i> *fp, int requestCode, int arg).

Возвращает

Целое число — номер, под которым зарегистрирован в системе драйвер этого устройства.

19.32.4.30. iosFdShow()

```
void iosFdShow (
    void )
```

Функция выводит список всех открытых файлов в stdout.

19.32.4.31. ioTaskStdSet()

```
STATUS ioTaskStdSet (
    TASK_ID task,
    int ioe,
    int fd )
```

Функция перенаправляет ввод/вывод для отдельной задачи.

Аргументы	
<i>task</i>	Задача, ввод/вывод которой требуется перенаправить или NULL, для перенаправления ввода/вывода текущей задачи.
<i>ioe</i>	Поток: 0 = stdin , 1 = stdout , 2 = stderr .

Продолжение на следующей странице

Аргументы (Продолжение.)

fd Дескриптор устройства, куда требуется перенаправить поток.

Возвращает

OK при успешном выполнении.

ERROR при ошибке в параметрах.

19.32.4.32. lseek()

```
int lseek (
    int fd,
    int shift,
    int seektype )
```

Функция перемещает указатель чтения / записи в файле.

Аргументы

fd дескриптор файла

shift смещение указателя чтения/записи в файле

seektype Позиция указателя, относительно которой будет выполняться смещение. Такая позиция задаётся одной из следующих констант, определённых в *stdio.h*:

- *SEEK_SET* — Начало файла.
- *SEEK_CUR* — Текущее положение файла.
- *SEEK_END* – Конец файла.

Возвращает

OK.

ERROR при неудаче.

19.32.4.33. mkdir()

```
STATUS mkdir (
    const char * name )
```

Создать каталог.

Если файл/каталог с таким названием уже существует, то функция вернет ошибку.

Аргументы

name Путь к каталогу.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.34. mountVolume()

```
STATUS mountVolume (  
    int fsDrvNum,  
    const char * devName,  
    int volume,  
    int partNr,  
    BLK_DEV * driver )
```

Функция привязки *блочного устройства* к файловой системе.

Аргументы

<i>fsDrvNum</i>	Дескриптор файловой системы (например, <i>DosDriver</i>).
<i>devName</i>	Имя устройства, например C: или G: (для файловой системы MSDOS всегда состоит из одной буквы и стоящего за ней двоеточия).
<i>volume</i>	Если на устройстве имеются разделы (как на жестком диске), то нужно задать значение 0x80 . В противном случае (как на дискете) требуется задать значение 0 .
<i>partNr</i>	Номер раздела (0).
<i>driver</i>	Указатель на структуру — драйвер блочного устройства.

Возвращает

OK.
ERROR при неудаче монтирования.

19.32.4.35. open()

```
int open (  
    const char * path,  
    int flags )
```

Функция открывает файл или устройство по его имени.

Аргументы

path Имя открываемого файла или устройства

Продолжение на следующей странице

Аргументы (Продолжение.)

flags Флаги для открытия, такие как *O_RDONLY*, *O_WRONLY*, *O_RDWR*, *O_CREAT*.

См. также

Полный перечень флагов см. в разделе *Флаги базовой системы ввода / вывода*.

Возвращает

Дескриптор открытого файла, небольшое целое число. При неудаче вернет отрицательное значение.

19.32.4.36. read()

```
int read (  
    int fd,  
    void * buffer,  
    int maxBytes )
```

Функция читает в буфер из файла заданное кол-во байт.

Аргументы

fd Дескриптор файла.
buffer Указатель на буфер.
maxBytes Заданное количество байт.

Возвращает

Количество фактически прочитанных байт.

19.32.4.37. remove()

```
STATUS remove (  
    const char * path )
```

Функция удаляет файл по его имени.

Аргументы

path Имя удаляемого файла.

Возвращает

OK.
ERROR при неудаче.

19.32.4.38. removeDevice()

STATUS removeDevice (
 const char * devName)

Функция удаляет из системы заданное устройство по его имени.

Аргументы

devName Символьное имя устройства.

Возвращает

OK при успешном завершении.
ERROR при неудаче.

19.32.4.39. reset()

int reset (
 const char * devName)

Сброс контроллера **IDE**.

Аргументы

devName Идентификатор устройства.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.40. rmdir()

STATUS rmdir (
 const char * name)

Удалить пустой каталог.

Не-пустой или read-only каталог удалить не получится.

Аргументы

name Путь к каталогу.

Возвращает

OK при успехе, *ERROR* при ошибке.

19.32.4.41. seek()

```
int seek (  
    int fd,  
    int shift,  
    int seektype )
```

Функция перемещает указатель чтения / записи в файле.

Аргументы

fd дескриптор файла

shift смещение указателя чтения/записи в файле

seektype Позиция указателя, относительно которой будет выполняться смещение. Такая позиция задаётся одной из следующих констант, определённых в *stdio.h*:

- *SEEK_SET* — Начало файла.
 - *SEEK_CUR* — Текущее положение файла.
 - *SEEK_END* – Конец файла.
-

Возвращает

OK.
ERROR при неудаче.

19.32.4.42. tell()

```
int tell (  
    int fd )
```

Функция возвращает текущее значение положения указателя в файле.

Аргументы

fd Дескриптор файла.

Возвращает

Функция возвращает текущую позицию.

19.32.4.43. `unloadVolume()`

```
int unloadVolume (  
    const char * devName )
```

Разгрузка съемного носителя.

Аргументы

<code>devName</code>	Идентификатор устройства.
----------------------	---------------------------

Возвращает

`OK` при успехе, `ERROR` при ошибке.

19.32.4.44. `upd()`

```
STATUS upd (  
    void )
```

Изменить рабочий каталог на его родительский каталог.

Аналог команды `'cd ../'`.

Возвращает

`OK` при успехе, `ERROR` при ошибке.

19.32.4.45. `write()`

```
int write (  
    int fd,  
    const void * buffer,  
    int maxBytes )
```

Функция записывает из буфера в файл заданное кол-во байт.

Аргументы

<code>fd</code>	Дескриптор файла.
-----------------	-------------------

<code>buffer</code>	Указатель на буфер.
---------------------	---------------------

<code>maxBytes</code>	Заданное количество байт.
-----------------------	---------------------------

Возвращает

Количество фактически записанных байт.

19.32.5. Переменные

19.32.5.1. sysDeviceList

*DEV_HDR** sysDeviceList [extern]

Список устройств.

19.33. Файл `iso646.h`

Текстовые макросы для символьных операторов Си.

Текстовые макросы для символьных операторов Си.

- `#define and &&`
- `#define and_eq &=`
- `#define bitand &`
- `#define bitor |`
- `#define compl ~`
- `#define not !`
- `#define not_eq !=`
- `#define or ||`
- `#define or_eq |=`
- `#define xor ^`
- `#define xor_eq ^=`

19.33.1. Подробное описание

См. стандарт C11 7.9.

См. также

`C11 standard 7.9.`

19.33.2. Макросы

19.33.2.1. `and`

```
#define and &&
```

19.33.2.2. `and_eq`

```
#define and_eq &=
```

19.33.2.3. `bitand`

```
#define bitand &
```

19.33.2.4. `bitor`

```
#define bitor |
```

19.33.2.5. `compl`

```
#define compl ~
```

19.33.2.6. not

```
#define not !
```

19.33.2.7. not_eq

```
#define not_eq !=
```

19.33.2.8. or

```
#define or ||
```

19.33.2.9. or_eq

```
#define or_eq |=
```

19.33.2.10. xor

```
#define xor ^
```

19.33.2.11. xor_eq

```
#define xor_eq ^=
```

19.34. Файл kernel.dox

19.35. Файл `limits.h`

Характеристики общих типов.

Макросы

- `#define CHAR_BIT` (8)
- `#define MB_LEN_MAX` 6

signed char

Минимальное и максимальное значения типа:

- `#define SCHAR_MAX` (127)
- `#define SCHAR_MIN` (-128)

unsigned char

Максимальное значение типа (минимальное - 0):

- `#define UCHAR_MAX` (255)

char

Минимальное и максимальное значения типа:

- `#define CHAR_MAX` `UCHAR_MAX`
- `#define CHAR_MIN` 0

signed short int

Минимальное и максимальное значения типа:

- `#define SHRT_MAX` (32767)
- `#define SHRT_MIN` (-32768)

unsigned short int

Максимальное значение типа (минимальное - 0):

- `#define USHRT_MAX` (65535)

signed int.

Минимальное и максимальное значения типа:

- `#define INT_MAX` (2147483647)
- `#define INT_MIN` (-`INT_MAX` - 1)

unsigned int

Максимальное значение типа (минимальное - 0):

- `#define UINT_MAX` 4294967295U

signed long int

Минимальное и максимальное значения типа:

- #define *LONG_MAX* (2147483647L)
- #define *LONG_MIN* (-*LONG_MAX* - 1L)

unsigned long int

Максимальное значение типа (минимальное - 0):

- #define *ULONG_MAX* 4294967295UL

unsigned long long int.

Минимальное и максимальное значения типа:

- #define *LLONG_MAX* 9223372036854775807LL
- #define *LLONG_MIN* (-*LLONG_MAX* - 1LL)

unsigned long long int

Максимальное значение типа (минимальное - 0):

- #define *ULLONG_MAX* 18446744073709551615ULL

19.35.1. Подробное описание

См. стандарт C11 7.10.

См. также

[C11 standard 7.10.](#)

19.35.2. Макросы

19.35.2.1. CHAR_BIT

```
#define CHAR_BIT (8)
```

Количество бит в типе **char**.

19.35.2.2. CHAR_MAX

```
#define CHAR_MAX UCHAR_MAX
```

19.35.2.3. CHAR_MIN

```
#define CHAR_MIN 0
```

19.35.2.4. INT_MAX

```
#define INT_MAX (2147483647)
```

19.35.2.5. INT_MIN

```
#define INT_MIN (-INT_MAX - 1)
```

19.35.2.6. LLONG_MAX

```
#define LLONG_MAX 9223372036854775807LL
```

19.35.2.7. LLONG_MIN

```
#define LLONG_MIN (-LLONG_MAX - 1LL)
```

19.35.2.8. LONG_MAX

```
#define LONG_MAX (2147483647L)
```

19.35.2.9. LONG_MIN

```
#define LONG_MIN (-LONG_MAX - 1L)
```

19.35.2.10. MB_LEN_MAX

```
#define MB_LEN_MAX 6
```

Максимальная длина мультибайтного символа во всех возможных локалях.

19.35.2.11. SCHAR_MAX

```
#define SCHAR_MAX (127)
```

19.35.2.12. SCHAR_MIN

```
#define SCHAR_MIN (-128)
```


19.35.2.13. SHRT_MAX

```
#define SHRT_MAX (32767)
```

19.35.2.14. SHRT_MIN

```
#define SHRT_MIN (-32768)
```

19.35.2.15. UCHAR_MAX

```
#define UCHAR_MAX (255)
```

19.35.2.16. UINT_MAX

```
#define UINT_MAX 4294967295U
```

19.35.2.17. ULLONG_MAX

```
#define ULLONG_MAX 18446744073709551615ULL
```

19.35.2.18. ULONG_MAX

```
#define ULONG_MAX 4294967295UL
```

19.35.2.19. USHRT_MAX

```
#define USHRT_MAX (65535)
```

19.36. Файл list.h

Не-потокобезопасный двухсвязный список.

Структуры данных

- struct *listNode*
- struct *sList*

Макросы

- #define *INIT_STATIC_LIST*() (&(sList){.pFirst = 0, .pLast = 0, .numOfNodes = 0})
Макрос для инициализации списка в сегменте данных. Инициализация производится в виде sList MyListPtr = INIT_STATIC_LIST(); Освобождение такого списка недопустимо!*
- #define *NODE_FIELD*(node, ptrType, field) ((ptrType) (node->pData))->field)
Макрос для получения поля структуры, хранимой в списке.

Определения типов

- typedef struct *listNode* *sListNode*

Функции

- *STATUS list_AddToBack* (*sList* *pList, *sListNode* *pNewNode)
Вставить узел в конец списка.
- *STATUS list_AddToHead* (*sList* *pList, *sListNode* *pNewNode)
Вставить узел в начало списка.
- *sList* * *list_CreateNewList* (void)
Создать новый двухсвязный список.
- *sListNode* * *list_CreateNewNode* (const void *pData, size_t size)
Создать новый узел списка и скопировать в него данные.
- *sListNode* * *list_CreateNewNodeWithoutDataCopy* (void *pData, size_t size)
Создать новый узел списка без копирования данных.
- *STATUS list_CutByIndex* (*sList* *pList, size_t index)
Вырезать узел по его индексу.
- *STATUS list_CutByNodePointer* (*sList* *pList, *sListNode* *pTargetNode)
Вырезать узел по его указателю.
- *STATUS list_CutFromBack* (*sList* *pList)
Вырезать узел из конца списка.
- *STATUS list_CutFromHead* (*sList* *pList)
Вырезать узел из начала списка.
- void *list_DoForeach* (*sList* *pList, void(*pFunc)(void *pData))
Вызвать функцию для каждого элемента в списке.
- void *list_FreeAll* (*sList* *pList)
Освободить все ресурсы занимаемые списком и хранящимися в нем данными.
- void *list_FreeNodesAndList* (*sList* *pList)
Освободить все ресурсы занимаемые списком.
- *STATUS list_RemoveByIndex* (*sList* *pList, size_t index)
Удалить узел по его индексу.
- *STATUS list_RemoveByNodePointer* (*sList* *pList, *sListNode* *pTargetNode)
Удалить узел по его указателю.
- *STATUS list_RemoveFromBack* (*sList* *pList)
Удалить узел из конца списка.
- *STATUS list_RemoveFromHead* (*sList* *pList)
Удалить узел из начала списка.

19.36.1. Подробное описание

Работа с списками.

19.36.2. Макросы

19.36.2.1. INIT_STATIC_LIST

```
#define INIT_STATIC_LIST( ) (&(sList){.pFirst = 0, .pLast = 0, .numOfNodes = 0})
```

19.36.2.2. NODE_FIELD

```
#define NODE_FIELD(  
    node,  
    ptrType,  
    field ) (( (ptrType) ( node->pData ) )->field)
```

19.36.3. Типы

19.36.3.1. sListNode

```
typedef struct listNode sListNode
```

Узел двухсвязного списка.

19.36.4. Функции

19.36.4.1. list_AddToBack()

```
STATUS list_AddToBack (  
    sList * pList,  
    sListNode * pNewNode )
```

Аргументы

<i>pList</i>	Указатель на список.
<i>pNewNode</i>	Указатель на вставляемый узел.

Возвращает

ОК при успехе, ERROR иначе.

19.36.4.2. list_AddToHead()

```
STATUS list_AddToHead (  
    sList * pList,  
    sListNode * pNewNode )
```

Аргументы

<i>pList</i>	Указатель на список.
<i>pNewNode</i>	Указатель на вставляемый узел.

Возвращает

ОК при успехе, ERROR иначе.

19.36.4.3. list_CreateNewList()

```
sList* list_CreateNewList (  
    void )
```

Возвращает

Указатель на *sList* или NULL при ошибке.

19.36.4.4. list_CreateNewNode()

```
sListNode* list_CreateNewNode (  
    const void * pData,  
    size_t size )
```

Данные из аргумента будут скопированы, а значение *pData* в узле будет указывать на эту копию.

Аргументы

<i>pData</i>	Указатель на данные.
<i>size</i>	Размер данных.

Возвращает

Указатель на *sListNode* или NULL при ошибке.



При освобождении списка нужно будет предварительно вручную освободить копии данных.

19.36.4.5. list_CreateNewNodeWithoutDataCopy()

```
sListNode* list_CreateNewNodeWithoutDataCopy (  
    void * pData,  
    size_t size )
```

Данные из аргумента HE будут скопированы, а значение pData в узле будет иметь значение аргумента.

Аргументы	
<i>pData</i>	Указатель на данные или NULL.
<i>size</i>	Размер данных.

Возвращает

Указатель на sListNode или NULL при ошибке.

19.36.4.6. list_CutByIndex()

```
STATUS list_CutByIndex (  
    sList * pList,  
    size_t index )
```

Аргументы	
<i>pList</i>	Указатель на список.
<i>index</i>	Индекс вырезаемого узла.

Возвращает

OK при успехе, ERROR иначе.

Узел и данные в нем не будут освобождены.

19.36.4.7. list_CutByNodePointer()

```
STATUS list_CutByNodePointer (  
    sList * pList,  
    sListNode * pTargetNode )
```

Аргументы	
<i>pList</i>	Указатель на список.
<i>pTargetNode</i>	Указатель на вырезаемый узел.

Возвращает

OK при успехе, ERROR иначе.

Узел и данные в нем не будут освобождены.

19.36.4.8. list_CutFromBack()

```
STATUS list_CutFromBack (  
    sList * pList )
```

Аргументы

pList Указатель на список.

Возвращает

ОК при успехе, ERROR иначе.

Узел и данные в нем не будут освобождены.

19.36.4.9. list_CutFromHead()

```
STATUS list_CutFromHead (  
    sList * pList )
```

Аргументы

pList Указатель на список.

Возвращает

ОК при успехе, ERROR иначе.

Узел и данные в нем не будут освобождены.

19.36.4.10. list_DoForeach()

```
void list_DoForeach (  
    sList * pList,  
    void(*)(void *pData) pFunc )
```

Аргументы

pList Указатель на список.

pFunc Указатель на функцию, которая будет вызывана для каждого узла в списке. В качестве аргумента в функцию будет отправлен указатель на данные в узле.

19.36.4.11. list_FreeAll()

```
void list_FreeAll (  
    sList * pList )
```

Аргументы

pList Указатель на список.

Работает идентично `list_FreeNodesAndList`, но дополнительно вызывает `free()` для поля `pData` в каждом узле списка.

19.36.4.12. list_FreeNodesAndList()

```
void list_FreeNodesAndList (  
    sList * pList )
```

Аргументы

pList Указатель на список.

Не освобождает ресурсы, занимаемые хранящимися в списке данными.

19.36.4.13. list_RemoveByIndex()

```
STATUS list_RemoveByIndex (  
    sList * pList,  
    size_t index )
```

Аргументы

pList Указатель на список.

index Индекс удаляемого узла.

Возвращает

ОК при успехе, ERROR иначе.

И узел, и данные в нем будут освобождены.

19.36.4.14. list_RemoveByNodePointer()

```
STATUS list_RemoveByNodePointer (  
    sList * pList,  
    sListNode * pTargetNode )
```

Аргументы

pList Указатель на список.

pTargetNode Указатель на удаляемый узел.

Возвращает

ОК при успехе, ERROR иначе.

И узел, и данные в нем будут освобождены.

19.36.4.15. list_RemoveFromBack()

STATUS list_RemoveFromBack (
 sList * *pList*)

Аргументы

pList Указатель на список.

Возвращает

ОК при успехе, ERROR иначе.

И узел, и данные в нем будут освобождены.

19.36.4.16. list_RemoveFromHead()

STATUS list_RemoveFromHead (
 sList * *pList*)

Аргументы

pList Указатель на список.

Возвращает

ОК при успехе, ERROR иначе.

И узел, и данные в нем будут освобождены.

19.37. Файл manual.dox

19.38. Файл `mapstr.h`

Список пар типа **строка-значение**.

Структуры данных

- struct `tMapIterators`
Набор указателей для работы со списками.

Макросы

- #define `MAPSTR_SIGNATURE` 0x75AADD00

Перечисления

- enum `eMapstrValueType` { `mapTypeUnknown` = 0 , `mapTypeString` = MAPSTR_SIGNATURE | 1 , `mapTypeInt` = MAPSTR_SIGNATURE | 2 }
- Типы данных, используемые в качестве значений.*

Создание и удаление списков

- void `mapFree` (`tMapIterators` *iter)
Удаление списка — очистка памяти, отведённой под список.
- `tMapIterators` * `newMap` ()
Создание нового списка.

Работа со списком

- void `mapAppendInt` (`tMapIterators` *iter, const char *key, int value, const char *description)
*Добавить значение типа **целое** к списку.*
- void `mapAppendIntArray` (`tMapIterators` *iter, const char *key, int *values, int count, const char *description)
*Добавить массив значений типа **целое** к списку.*
- void `mapAppendString` (`tMapIterators` *iter, const char *key, const char *value, const char *description)
*Добавить значение типа **строка** к списку.*
- bool `mapCheckString` (const `tMapIterators` *iter, const char *key, const char *value)
Проверка строкового значения на соответствие заданному значению.
- const void * `mapFind` (const `tMapIterators` *iter, const char *key, int *count, `eMapstrValueType` *vType)
Поиск по списку.
- void `mapPrint` (const `tMapIterators` *iter, const char *header)
Печать списка в консоль.

Работа с файлами

Списки можно сохранять и читать из файлов.

- void `mapRestore` (const char *path, `tMapIterators` *iter)
Прочитать список из файла.
- void `mapRestorefFromEverywhere` (const char *ext, `tMapIterators` *iter)
Найти все возможные файлы конфигурации и прочитать их.
- void `mapStore` (const char *path, const `tMapIterators` *iter)
Сохранить список.

19.38.1. Подробное описание

Разработано специально для описания аппаратной части проектов. Под каждую запись выделяется минимально возможное место в ОЗУ, кратное блоку в 512 байт. Записи в списке находятся по уникальному ключу, заданному при создании записи. Если ключи некоторых записей совпадают - будет возвращено значение первой найденной записи.

19.38.2. Макросы

19.38.2.1. MAPSTR_SIGNATURE

```
#define MAPSTR_SIGNATURE 0x75AADD00
```

Просто число, отличное от нуля с пустым последним байтом.

19.38.3. Перечисления

19.38.3.1. eMapstrValueType

```
enum eMapstrValueType
```

Элементы перечислений

mapTypeUnknown Значение не определено.

mapTypeString Значение - строка.

mapTypeInt Значение (массив значений) - целое со знаком.

```
00037                {
00038        mapTypeUnknown = 0,
00039        mapTypeString    = MAPSTR_SIGNATURE | 1,
00040        mapTypeInt        = MAPSTR_SIGNATURE | 2,
00041 } eMapstrValueType;
```

19.38.4. Функции

19.38.4.1. mapAppendInt()

```
void mapAppendInt (
    tMapIterators * iter,
    const char * key,
    int value,
    const char * description )
```

Функция добавляет значение к имеющемуся списку и обновляет значение итераторов.

Аргументы	
<i>iter</i>	Итераторы списка.
<i>key</i>	Ключ, по которому будет осуществляться поиск значения.
<i>value</i>	Значение, соответствующее ключу.
<i>description</i>	Строковое описание параметра может быть использовано для вывода списка на печать. Если не используется, может быть NULL.

19.38.4.2. mapAppendIntArray()

```
void mapAppendIntArray (
    tMapIterators * iter,
    const char * key,
    int * values,
    int count,
    const char * description )
```

Функция добавляет массив значений к имеющемуся списку и обновляет значение итераторов.

Аргументы	
<i>iter</i>	Итераторы списка.
<i>key</i>	Ключ, по которому будет осуществляться поиск значения.
<i>values</i>	Добавляемый массив.
<i>count</i>	Количество элементов массива.
<i>description</i>	Строковое описание параметра может быть использовано для вывода списка на печать. Если не используется, может быть NULL.

19.38.4.3. mapAppendString()

```
void mapAppendString (
    tMapIterators * iter,
    const char * key,
    const char * value,
    const char * description )
```

Функция добавляет строку к имеющемуся списку и обновляет значение итераторов.

Аргументы	
<i>iter</i>	Итераторы списка.
<i>key</i>	Ключ, по которому будет осуществляться поиск значения.
<i>value</i>	Значение, соответствующее ключу.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>description</i>	Строковое описание параметра может быть использовано для вывода списка на печать. Если не используется, может быть NULL.
--------------------	--

19.38.4.4. mapCheckString()

```
bool mapCheckString (
    const tMapIterators * iter,
    const char * key,
    const char * value )
```

Функция осуществляет поиск значения по ключу в списке и сравнивает найденное с заданным.

Аргументы

<i>iter</i>	Итераторы списка.
<i>key</i>	Ключ, по которому будет осуществляться поиск значения.
<i>value</i>	Проверяемое значение.

Возвращает

true Если значение найдено и соответствует заданному.
false Во всех остальных случаях.

19.38.4.5. mapFind()

```
const void* mapFind (
    const tMapIterators * iter,
    const char * key,
    int * count,
    eMapstrValueType * vType )
```

Функция осуществляет поиск по ключу в списке и возвращает найденное значение, либо **NULL**.

Аргументы

in	<i>iter</i>	Итераторы списка.
in	<i>key</i>	Ключ, по которому будет осуществляться поиск значения.
out	<i>count</i>	Количество элементов, содержащихся в массиве значений. Для строк это значение всегда равно 1. Если параметр не требуется, указатель может быть равен NULL.
out	<i>vType</i>	Указатель на переменную, в которую будет помещён тип найденного значения. Если тип указывать не нужно – следует передать NULL вместо указателя.

Возвращает

Указатель на данные. Если тип данных заранее не известен можно ориентироваться по возвращаемому параметру **vType**.

19.38.4.6. mapFree()

```
void mapFree (  
    tMapIterators * iter )
```

Функция освобождает всю память выделенную под список и его итераторы.

Аргументы

iter Итераторы списка.

19.38.4.7. mapPrint()

```
void mapPrint (  
    const tMapIterators * iter,  
    const char * header )
```

Функция выводит в консоль все элементы списка в две колонки. Может использоваться для отладки.

Аргументы

iter Итераторы списка.

header Заголовок выводимый в начале печати.

19.38.4.8. mapRestore()

```
void mapRestore (  
    const char * path,  
    tMapIterators * iter )
```

Элементы прочитанные из файла будут добавлены к указанному списку.

Аргументы

path Полный путь к файлу.

iter Итераторы уже существующего списка. **ВАЖНО!** Если список ещё не создан его следует создать с помощью *newMap()*.

19.38.4.9. mapRestoreFromEverywhere()

```
void mapRestoreFromEverywhere (  
    const char * ext,  
    tMapIterators * iter )
```

Функция ищет файлы с указанным расширением (обычно **arc**) в корневых каталогах дисков и пытается извлечь из них списки. Каждый следующий найденный список будет добавлен в конец имеющегося.

Аргументы

ext Расширение файлов со списками (обычно **arc**).

iter Итераторы уже существующего списка. **ВАЖНО!** Если список ещё не создан его следует создать с помощью *newMap()*.

19.38.4.10. mapStore()

```
void mapStore (  
    const char * path,  
    const tMapIterators * iter )
```

Функция сохраняет список в текстовый файл по указанному пути. Если файл существует он будет перезаписан.

Аргументы

path Полный путь к файлу.

iter Итераторы списка.

19.38.4.11. newMap()

```
tMapIterators* newMap ( )
```

Создание и инициализация набора итераторов для нового списка.

Возвращает

tMapIterators* Готовый набор итераторов.

19.39. Файл `math.h`

Стандартные математические функции

Структуры данных

- struct `complex`

Макросы

- #define `M_LN2` 0.693147180559945309417
- #define `M_PI` 3.1415926535897932384626433832795
- #define `M_SQRT2` 1.4142135623730950488016887242097

Функции

- double `acos` (double)
- double `asin` (double)
- double `atan` (double)
- double `atan2` (double, double)
- float `atan2f` (float, float)
- float `atanf` (float)
- double `cabs` (`complex`)
- double `cbrt` (double)
- float `cbrtf` (float)
- double `ceil` (double)
- float `ceilf` (float)
- double `cos` (double)
- float `cosf` (float)
- double `cosh` (double x)
- double `exp` (double)
- double `exp2` (double)
- float `exp2f` (float)
- float `expf` (float)
- double `fabs` (double)
- float `fabsf` (float)
- double `floor` (double)
- float `floorf` (float)
- double `fmod` (double, double)
- double `frac` (double)
- double `frexp` (double, int *)
- int `isinf` (double x)
- int `isnan` (double x)
- double `ldexp` (double, int)
- float `ldexpf` (float, int)
- long long int `llrint` (double x)
- long long int `llrintf` (float x)
- double `log` (double)
- double `log10` (double)
- float `log10f` (float)
- float `log2f` (float)
- long int `lrint` (double x)
- long int `lrintf` (float x)
- double `pi` ()
- double `pow` (double, double)
- float `powf` (float, float)
- double `round` (double)
- float `roundf` (float)
- double `sin` (double)
- float `sinf` (float)
- double `sinh` (double x)
- double `sqrt` (double)
- float `sqrtf` (float)
- double `tan` (double)
- double `tanh` (double x)
- double `trunc` (double x)
- float `truncf` (float)

19.39.1. Макросы

19.39.1.1. M_LN2

```
#define M_LN2 0.693147180559945309417
```

19.39.1.2. M_PI

```
#define M_PI 3.1415926535897932384626433832795
```

19.39.1.3. M_SQRT2

```
#define M_SQRT2 1.4142135623730950488016887242097
```

19.39.2. Функции

19.39.2.1. acos()

```
double acos (  
    double )
```

19.39.2.2. asin()

```
double asin (  
    double )
```

19.39.2.3. atan()

```
double atan (  
    double )
```

19.39.2.4. atan2()

```
double atan2 (  
    double ,  
    double )
```

19.39.2.5. atan2f()

```
float atan2f (  
    float ,  
    float )
```

float,
float)

19.39.2.6. atanf()

float atanf (
float)

19.39.2.7. cabs()

double cabs (
complex)

19.39.2.8. cbrt()

double cbrt (
double)

19.39.2.9. cbrtf()

float cbrtf (
float)

19.39.2.10. ceil()

double ceil (
double)

19.39.2.11. ceilf()

float ceilf (
float)

19.39.2.12. cos()

double cos (
double)

19.39.2.13. cosf()

float cosf (
float)

float)

19.39.2.14. cosh()

double cosh (
double x)

19.39.2.15. exp()

double exp (
double)

19.39.2.16. exp2()

double exp2 (
double)

19.39.2.17. exp2f()

float exp2f (
float)

19.39.2.18. expf()

float expf (
float)

19.39.2.19. fabs()

double fabs (
double)

19.39.2.20. fabsf()

float fabsf (
float)

19.39.2.21. floor()

double floor (
double)

```
double )
```

19.39.2.22. floorf()

```
float floorf (  
    float )
```

19.39.2.23. fmod()

```
double fmod (  
    double ,  
    double )
```

19.39.2.24. frac()

```
double frac (  
    double )
```

19.39.2.25. frexp()

```
double frexp (  
    double ,  
    int * )
```

19.39.2.26. isinf()

```
int isinf (  
    double x )
```

19.39.2.27. isnan()

```
int isnan (  
    double x )
```

19.39.2.28. ldexp()

```
double ldexp (  
    double ,  
    int )
```

19.39.2.29. ldexpf()

```
float ldexpf (
    float ,
    int )
```

19.39.2.30. llrint()

```
long long int llrint (
    double x )
```

19.39.2.31. llrintf()

```
long long int llrintf (
    float x )
```

19.39.2.32. log()

```
double log (
    double )
```

19.39.2.33. log10()

```
double log10 (
    double )
```

19.39.2.34. log10f()

```
float log10f (
    float )
```

19.39.2.35. log2f()

```
float log2f (
    float )
```

19.39.2.36. lrint()

```
long int lrint (
    double x )
```

19.39.2.37. lrintf()

```
long int lrintf (
    float x )
```

19.39.2.38. pi()

```
double pi ( )
```

19.39.2.39. pow()

```
double pow (
    double ,
    double )
```

19.39.2.40. powf()

```
float powf (
    float ,
    float )
```

19.39.2.41. round()

```
double round (
    double )
```

19.39.2.42. roundf()

```
float roundf (
    float )
```

19.39.2.43. sin()

```
double sin (
    double )
```

19.39.2.44. sinf()

```
float sinf (
    float )
```

19.39.2.45. sinh()

```
double sinh (  
    double x )
```

19.39.2.46. sqrt()

```
double sqrt (  
    double )
```

19.39.2.47. sqrtf()

```
float sqrtf (  
    float )
```

19.39.2.48. tan()

```
double tan (  
    double )
```

19.39.2.49. tanh()

```
double tanh (  
    double x )
```

19.39.2.50. trunc()

```
double trunc (  
    double x )
```

19.39.2.51. truncf()

```
float truncf (  
    float )
```

19.40. Файл memlib.h

Управление диспетчером памяти.

Функции

- void *initMemLib* (size_t begAddr, size_t endAddr)
Инициализация диспетчера памяти.
- size_t *maxAvail* (void)
Получить размер максимального непрерывного сегмента памяти.
- size_t *memAvail* (void)
Получить размер свободной памяти.
- size_t *memReport* (bool allBlocks)
Получить отчёт о занятых блоках памяти.

Макросы обеспечения совместимости

- #define *__free* *free*
- #define *kfree*(x) *free* (x)
- #define *kmalloc*(n, z) *malloc* (n)

Функции из stdlib.

- void * *calloc* (size_t nmemb, size_t size)
- *STATUS free* (void *ptr)
- void * *malloc* (size_t size)
- void * *mallocForOwner* (size_t size, const char *owner)
Выделить подписанный блок памяти для объекта.
- *STATUS mfree* (void *p, const char *name)
Освободить блок памяти (с подписью).
- void *minfo* (const void *ptr)
Вывод на печать заголовка блока памяти.
- void * *realloc* (void *ptr, size_t size)

19.40.1. Подробное описание

Файл содержит объявления методов управления диспетчером памяти.

См. также

Общее описание см. в главе [Диспетчер памяти](#).

19.40.2. Макросы

19.40.2.1. __free

```
#define __free free
```

19.40.2.2. kfree

```
#define kfree(  
    x ) free (x)
```


19.40.2.3. kcalloc

```
#define kcalloc(  
    n,  
    z ) malloc (n)
```

19.40.3. Функции

19.40.3.1. calloc()

```
void* calloc (  
    size_t nmemb,  
    size_t size )
```

Выделить блок памяти для массива и инициализировать его нулями.

Аргументы

nmemb Количество элементов в массиве.

size Размер одного элемента.

Возвращает

Указатель на блок памяти или NULL, в случае ошибки.

19.40.3.2. free()

```
STATUS free (  
    void * ptr )
```

Освободить блок памяти.

Аргументы

ptr Указатель на блок памяти для освобождения.



Функция проверяет корректность заголовка блока памяти и освобождает память только в случае его правильности.

Возвращает

OK при удачном освобождении памяти, иначе *ERROR*.

19.40.3.3. `initMemLib()`

```
void initMemLib (
    size_t begAddr,
    size_t endAddr )
```

Функция инициализирует диспетчер памяти.



Эта функция вызывается ядром *MULTEX-ARM* внутри вызова **kernelInit**. Пользователь не должен вызывать эту функцию из своих задач, так как это приведет к краху системы.

Аргументы

begAddr Start of heap memory pool.

endAddr End of heap memory pool.

19.40.3.4. `malloc()`

```
void* malloc (
    size_t size )
```

Выделить блок памяти для объекта.

Аргументы

size Размер запрашиваемого блока.

Возвращает

Указатель на блок памяти или `NULL`, в случае ошибки.

19.40.3.5. `mallocForOwner()`

```
void* mallocForOwner (
    size_t size,
    const char * owner )
```

Выделение памяти с помощью *malloc()* с добавлением подписи владельца блока памяти. В дальнейшем подписи блоков можно посмотреть с помощью *memReport()*.

Аргументы

<i>size</i>	Размер запрашиваемого блока.
<i>owner</i>	Владелец выделяемого блока – строка не более 239 символов.

Возвращает

Указатель на блок памяти или NULL, в случае ошибки.

19.40.3.6. maxAvail()

```
size_t maxAvail (  
    void )
```

Функция возвращает размер самого большого свободного сегмента динамической памяти в байтах.

Возвращает

Размер сегмента в байтах.

19.40.3.7. memAvail()

```
size_t memAvail (  
    void )
```

Функция возвращает общий размер свободной динамической памяти в байтах.

Возвращает

Размер памяти в байтах.

19.40.3.8. memReport()

```
size_t memReport (  
    bool allBlocks )
```

Функция возвращает работает аналогично *memAvail()*, кроме того выводит отчёт о всех выделенных с помощью *mallocForOwner()* блоках памяти в консоль.

Аргументы

<i>allBlocks</i>	Показывать все блоки, иначе будут показаны только подписанные блоки (владелец которых определён).
------------------	---

Возвращает

Размер памяти в байтах.

19.40.3.9. mfree()

```
STATUS mfree (
    void * p,
    const char * name )
```

Функция используется для отладки. В случае неудачного освобождение памяти выводит сообщение об ошибке с указанным именем.

Аргументы

p Указатель на блок памяти для освобождения.

name Подпись, выводимая при ошибке.

Возвращает

OK при удачном освобождении памяти, иначе *ERROR*.

19.40.3.10. minfo()

```
void minfo (
    const void * ptr )
```

Функция выводит в консоль информацию, записанную в заголовке выделенного с помощью *malloc()* блока памяти. Заголовок располагается перед каждым выделяемым блоком памяти и содержит системную информацию. Функция может пригодиться для отладки процесса управления памятью в проекте.

Аргументы

ptr Указатель на начало блока памяти.

19.40.3.11. realloc()

```
void* realloc (
    void * ptr,
    size_t size )
```

Выделить новый блок памяти заданного размера вместо старого, сохранив при этом данные из старого блока.

Аргументы

ptr Указатель на старый блок памяти.

size Размер нового блока памяти.

Возвращает

Указатель на блок памяти или NULL, в случае ошибки.

19.41. Файл `mpeg4codec.h`

Программно-аппаратный декодер видео файлов формата MP4. Использует аппаратный видео-енкодер процессора и препроцессор **NEON**.

Функции

- void `freeMpeg4FrameMem` (int dd)
Освобождение ресурсов, выделенных для работы с кадром.
- unsigned char * `getMpeg4InptFrameBuff` (int dd)
Указатель на входной буфер декодера.
- int `getMpeg4OutFrame` (int dd)
Указатель на выходной буфер декодера.
- int `mpeg4DecodeBlock` (int dd, unsigned char *pOutData, int len)
Декодировать блок данных (обычно кадр).
- int `mpeg4InitDecoder` (int wWidth, int wHeight)
Запуск декодера MP4.

19.41.1. Функции

19.41.1.1. `freeMpeg4FrameMem()`

```
void freeMpeg4FrameMem (  
    int dd )
```

Функция освобождает память, выделенную для работы с текущим кадром.

Аргументы

`dd` Дескриптор декодера, полученный при создании `mpeg4InitDecoder()`.

19.41.1.2. `getMpeg4InptFrameBuff()`

```
unsigned char* getMpeg4InptFrameBuff (  
    int dd )
```

Функция возвращает указатель на входной буфер данных декодера. Во входной буфер будут помещаться кадры для декодирования.

Аргументы

`dd` Дескриптор декодера, полученный при создании `mpeg4InitDecoder()`.

Возвращает

Указатель на буфер памяти.

19.41.1.3. getMpeg4OutFrame()

```
int getMpeg4OutFrame (  
    int dd )
```

Функция возвращает указатель на выходной буфер данных декодера. В выходном буфере содержатся готовые кадры после процесса декодирования. Выходной буфер может использоваться для вывода в **overlay** в режиме **tiled**.

Аргументы

dd Дескриптор декодера, полученный при создании *mpeg4InitDecoder()*.

Возвращает

Указатель на буфер памяти.

19.41.1.4. mpeg4DecodeBlock()

```
int mpeg4DecodeBlock (  
    int dd,  
    unsigned char * pOutData,  
    int len )
```

Функция декодирует один кадр, расположенный во входном буфере и кладёт его в выходной. Дополнительно может быть сделано преобразование в nontiled формат - результат будет помещён в **pOutData**.

Аргументы

dd Дескриптор декодера, полученный при создании *mpeg4InitDecoder()*.

pOutData Буфер для преобразования выходного буфера в **nontiled** формат, пригодный для вывода на **2D-поверхность** с помощью *методов* работы с поверхностями.

len Размер данных во входном буфере в байтах.

Возвращает

OK при удачном завершении декодирования, иначе **ERROR**.

19.41.1.5. mpeg4InitDecoder()

```
int mpeg4InitDecoder (
```

```
int wWidth,  
int wHeight )
```

Инициализация переменных декодера.

Аргументы

wWidth,wHeight Размеры кадра в пикселях.

Возвращает

Дескриптор созданного декодера.

19.42. Файл `msgqlib.h`

Создание очередей сообщений.

Структуры данных

- struct `msgQID`
Структура блока управления очереди.

Определения типов

- typedef `msgQID * MSG_Q_ID`
Указатель на идентификатор очереди.

Функции

- `MSG_Q_ID msgQCreate` (int maxMsgs, int maxMsgLength, int options)
Создать очередь сообщений.
- `STATUS msgQDelete` (`MSG_Q_ID` msgQId)
Удалить очередь.
- int `msgQNumMsgs` (`MSG_Q_ID` msgQId)
Число сообщений в очереди.
- `STATUS msgQReceive` (`MSG_Q_ID` msgQId, void *buffer, size_t maxNBytes, int timeout)
Прочитать сообщение из очереди.
- `STATUS msgQSend` (`MSG_Q_ID` msgQId, const void *buffer, size_t nBytes, int timeout, int priority)
Поместить сообщение в очередь.

Порядок получения данных из очереди

- #define `MSG_Q_FIFO` (0x00)
*Порядок получения данных из очереди **FIFO**.*
- #define `MSG_Q_PRIORITY` (0x01)
*Порядок получения данных из очереди **PRIORITY**.*

Приоритеты сообщений в очереди

- #define `MSG_PRI_NORMAL` (0)
- #define `MSG_PRI_URGENT` (1)

19.42.1. Подробное описание

Очереди сообщений – удобный механизм межзадачного взаимодействия. В файле собраны методы работы с очередями сообщений.

См. также

Общее описание очередей см. в главе [Очереди сообщений](#).

Пример использования очереди сообщений в простой задаче:

```
MSG_Q_ID queue = NULL;
bool stop;

// Задача в отдельной функции
int taskSimple (int dummy) {
    // Инициализация переменных задачи (если есть)
```

```
void *p = malloc (size);

while (!stop) {
    // Ожидание получения сообщений
    int data;
    msgQReceive (queue, &data, sizeof (int), WAIT_FOREVER);
    // ... тело задачи
    p[0] = data;
    // ...
}

// Освобождение выделенной памяти
free (p);
// Завершение задачи
return 0;
}

// Создание и запуск задачи
void testSimple () {
    int size = 10;
    queue = msgQCreate (size, sizeof (int), MSG_Q_FIFO);
    stop = false;
    taskSpawn ("{task simple}{}", 10, 0, 0, taskSimple, 0);
}

// Некое внешнее событие
void testRelease (int data) {
    // Отправить сообщение
    msgQSend (queue, &data, sizeof (int), NO_WAIT, MSG_PRI_NORMAL);
}
```

19.42.2. Макросы

19.42.2.1. MSG_PRI_NORMAL

```
#define MSG_PRI_NORMAL (0)
```

Нормальный приоритет - сообщения помещаются в конец очереди.

19.42.2.2. MSG_PRI_URGENT

```
#define MSG_PRI_URGENT (1)
```

Повышенный приоритет - сообщения помещаются в начало очереди.

19.42.2.3. MSG_Q_FIFO

```
#define MSG_Q_FIFO (0x00)
```

Задачи получают сообщения из очереди в порядке обращения.

19.42.2.4. MSG_Q_PRIORITY

```
#define MSG_Q_PRIORITY (0x01)
```

Задачи получают сообщения из очереди в соответствии с их приоритетами.

19.42.3. Типы

19.42.3.1. MSG_Q_ID

```
typedef msgQID* MSG_Q_ID
```

Указатель на структуру блока управления (идентификатора) очереди *msgQID*.

19.42.4. Функции

19.42.4.1. msgQCreate()

```
MSG_Q_ID msgQCreate (
    int maxMsgs,
    int maxMsgLength,
    int options )
```

Функция создает очередь сообщений.

Аргументы	
<i>maxMsgs</i>	Максимальное число сообщений, буферизуемых очередью. Если задача будет пытаться поместить в полную очередь сообщение, она будет задержана до тех пор, пока в очереди не освободится место для этого сообщения. Впрочем, задача может ждать заданное время или не ждать вовсе.
<i>maxMsgLength</i>	Максимальный размер одного сообщения. Сообщения могут иметь и меньший размер, но память под очередь будет выделяться из расчета maxMsgs*maxMsgLength , поэтому рекомендуется задавать эти величины в соответствии с конкретным применением очереди.
<i>options</i>	Способ доступа задач к очереди. Возможные значения <i>MSG_Q_FIFO</i> и <i>MSG_Q_PRIORITY</i> .

Возвращает

Идентификатор созданной очереди.

NULL при неудаче, вызванной нехваткой динамической памяти.

19.42.4.2. msgQDelete()

```
STATUS msgQDelete (
    MSG_Q_ID msgQId )
```

Функция удаляет очередь и высвобождает занятую под нее динамическую память.



Следует соблюдать осторожность при удалении очереди – в этот момент отдельные задачи могут ждать сообщений из нее.

Аргументы

msgQId Идентификатор удаляемой очереди.

Возвращает

OK, или *ERROR* при задании неверного идентификатора очереди.

19.42.4.3. msgQNumMsgs()

```
int msgQNumMsgs (  
    MSG_Q_ID msgQId )
```

Функция возвращает число сообщений, находящихся в данный момент в очереди.

Аргументы

msgQId Идентификатор очереди.

Возвращает

Функция возвращает целое число, равное количеству сообщений, находящихся в очереди на момент вызова, или **-1**, если очередь не существует.

19.42.4.4. msgQReceive()

```
STATUS msgQReceive (  
    MSG_Q_ID msgQId,  
    void * buffer,  
    size_t maxNBytes,  
    int timeout )
```

Функция читает сообщение из очереди. Если в очереди нет сообщений, то задача переводится в состояние ожидания.



Если задача ждет неограниченное время (*WAIT_FOREVER*), а очередь будет удалена, то задача навсегда останется заблокированной!
Не следует использовать очереди сообщений в обработчиках прерываний! В обработчиках следует использовать *кольцевой буфер*.

Аргументы	
<i>msgQId</i>	Идентификатор очереди.
<i>buffer</i>	Указатель на буфер, в который будет помещено полученное сообщение.
<i>maxNBytes</i>	Размер сообщения. Он должен соответствовать размеру сообщения, помещенного в очередь. Если размер указан большим, чем фактический, то к сообщению будет добавлена случайная информация.
<i>timeout</i>	Время ожидания в тиках таймера в случае, если очередь пуста. Допустимы также значения <i>NO_WAIT</i> и <i>WAIT_FOREVER</i> .

Возвращает

OK при успешном выполнении, либо значение меньше нуля – код ошибки.

19.42.4.5. msgQSend()

```
STATUS msgQSend (
    MSG_Q_ID msgQId,
    const void * buffer,
    size_t nBytes,
    int timeout,
    int priority )
```

Функция помещает сообщение в очередь. Если имеется задача, ждущая сообщения из этой очереди, и ее приоритет выше той, которая поместила сообщение, произойдет немедленное переключение на эту задачу.



Не следует использовать очереди сообщений в обработчиках прерываний! В обработчиках следует использовать *кольцевой буфер*.

Аргументы	
<i>msgQId</i>	Идентификатор очереди.
<i>buffer</i>	Указатель на буфер, из которого сообщение будет помещено в очередь. После вызова функции буфер может быть использован задачей для своих целей – сообщение копируется во внутренний буфер очереди.
<i>nBytes</i>	Размер сообщения. Он не должен превышать максимальный размер сообщения, отводимый при создании очереди
<i>timeout</i>	Время ожидания в тиках таймера в случае, если очередь переполнена. Допустимы также значения <i>NO_WAIT</i> и <i>WAIT_FOREVER</i> .
<i>priority</i>	Приоритет помещаемого сообщения: <ul style="list-style-type: none"> • <i>MSG_PRI_NORMAL</i> - для обычных сообщений, помещаемых в конец очереди. • <i>MSG_PRI_URGENT</i> - для внеочередных сообщений, помещаемых в начало очереди.

Возвращает

OK при успешном выполнении, либо значение меньше нуля – код ошибки.

19.43. Файл multex.h

Основной подключаемый файл **RTOS MULTEX-ARM**.

Макросы

- `#define __be32_to_cpu(x)`
- `#define __LITTLE_ENDIAN`
- `#define _MULTEX`
- `#define ARCH_DMA_MINALIGN 64`
- `#define clamp(val, min, max)`
- `#define debug_cond(cond, fmt, args...)`
- `#define DIV_ROUND_CLOSEST(x, divisor)`
- `#define get_unaligned(ptr)`
- `#define INIT_TASK_NAME "Init"`
- `#define KERN_DEBUG "MK_Debug:"`
- `#define KERN_INFO "MK_Info:"`
- `#define likely(x) __builtin_expect((x),1)`
- `#define printk printf`
- `#define put_unaligned(val, ptr)`
- `#define SHELL_NAME "Shell"`
- `#define SHELL_PRIORITY 100`
- `#define unlikely(x) __builtin_expect((x),0)`
- `#define VX_SUPERVISOR_MODE 0`

Определения типов

- `typedef int(* FUNCPTR) (int)`
Указатель на функцию, которая используется как точка входа для процессов.

Перечисления

- `enum STATUS { OK = 0 , ERROR = -1 }`
Результат выполнения.

Макросы типовых операций

- `#define __ALIGN_MASK(x, mask) (((x)+(mask))&~(mask))`
- `#define ALIGN(x, a) __ALIGN_MASK((x),(typeof(x))(a)-1)`
- `#define ALLOC_ALIGN_BUFFER(type, name, size, align)`
- `#define ALLOC_CACHE_ALIGN_BUFFER(type, name, size) ALLOC_ALIGN_BUFFER(type, name, size, ARCH_DMA_MINALIGN)`
- `#define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))`
- `#define DEFINE_ALIGN_BUFFER(type, name, size, align)`
- `#define DEFINE_CACHE_ALIGN_BUFFER(type, name, size) DEFINE_ALIGN_BUFFER(type, name, size, ARCH_DMA_MINALIGN)`
- `#define DIV_ROUND(n, d) (((n) + ((d)/2)) / (d))`
- `#define DIV_ROUND_UP(n, d) (((n) + (d) - 1) / (d))`
- `#define MAX(a, b) (((a)>(b))?(a):(b))`
- `#define MAX_SAFE(a, b)`
- `#define MEMBER_SIZE(type, member) (sizeof(((type *)0)->member))`
- `#define MIN(a, b) (((a)<(b))?(a):(b))`
- `#define MIN_SAFE(a, b)`
- `#define ROUND(a, b) (((a) + (b) - 1) & ~((b) - 1))`
Округление.
- `#define roundup(x, y) (((x) + ((y) - 1)) / (y)) * (y)`
Округление в большую сторону.
- `#define START_ONCE do{static int Started = 0;if(Started == 1){return;} Started = 1;}while(0)`
- `#define START_ONCE_R(r) do{static int Started = 0;if(Started == 1){return (r);} Started = 1;}while(0)`

Различные типы, использующиеся в некоторых модулях.

- typedef unsigned int *dma_addr_t*
- typedef int *irqreturn_t*
- typedef unsigned int *resource_size_t*

19.43.1. Подробное описание

В данном файле содержатся базовые определения операционной системы **RTOS MULTEX-ARM**.

См. также

Операционная система жесткого реального времени MULTEX-ARM.

19.43.2. Макросы

19.43.2.1. __ALIGN_MASK

```
#define __ALIGN_MASK(  
    x,  
    mask ) (((x)+(mask))&~(mask))
```

19.43.2.2. __be32_to_cpu

```
#define __be32_to_cpu(  
    x )
```

Макроопределение:

```
((((x) \& 0x000000ff) << 24) | \  
((x) \& 0x0000ff00) << 8) | \  
((x) \& 0x00ff0000) >> 8) | \  
((x) \& 0xff000000) >> 24))
```

19.43.2.3. __LITTLE_ENDIAN

```
#define __LITTLE_ENDIAN
```

Используемый в системе порядок следования бит.

19.43.2.4. _MULTEX_

```
#define _MULTEX_
```

Используемая операционная система.

19.43.2.5. ALIGN

```
#define ALIGN(  
    x,  
    a) __ALIGN_MASK((x),(typeof(x))(a)-1)
```

19.43.2.6. ALLOC_ALIGN_BUFFER

```
#define ALLOC_ALIGN_BUFFER(  
    type,  
    name,  
    size,  
    align )
```

Макроопределение:

```
char __ \# \#name[ROUND(size * sizeof(type), align) + (align - 1)]; \   
type *name = (type *) ALIGN((uintptr_t)__ \# \#name, align)
```

19.43.2.7. ALLOC_CACHE_ALIGN_BUFFER

```
#define ALLOC_CACHE_ALIGN_BUFFER(  
    type,  
    name,  
    size ) ALLOC_ALIGN_BUFFER(type, name, size, ARCH_DMA_MINALIGN)
```

19.43.2.8. ARCH_DMA_MINALIGN

```
#define ARCH_DMA_MINALIGN 64
```

19.43.2.9. ARRAY_SIZE

```
#define ARRAY_SIZE(  
    x ) (sizeof(x) / sizeof((x)[0]))
```

Макрос, возвращающий количество элементов массива.

19.43.2.10. clamp

```
#define clamp(  
    val,  
    min,  
    max )
```

Макроопределение:

```
({
    \
    typeof(val) __val = (val);      \
    typeof(min) __min = (min);     \
    typeof(max) __max = (max);    \
    (void) (&__val == &__min);    \
    (void) (&__val == &__max);    \
    __val = __val < __min ? __min: __val; \
    __val > __max ? __max: __val; })
```

19.43.2.11. debug_cond

```
#define debug_cond(
    cond,
    fmt,
    args... )
```

Макроопределение:

```
do {
    \
    if (cond)
        printf(fmt, \#\#args); \
} while (0)
```

19.43.2.12. DEFINE_ALIGN_BUFFER

```
#define DEFINE_ALIGN_BUFFER(
    type,
    name,
    size,
    align )
```

Макроопределение:

```
static char __\#\#name[roundup(size * sizeof(type), align)] \
    __attribute__((aligned(align))); \
static type *name = (type *)__\#\#name
```

19.43.2.13. DEFINE_CACHE_ALIGN_BUFFER

```
#define DEFINE_CACHE_ALIGN_BUFFER(
    type,
    name,
    size ) DEFINE_ALIGN_BUFFER(type, name, size, ARCH_DMA_MINALIGN)
```

19.43.2.14. DIV_ROUND

```
#define DIV_ROUND(
    n,
    d) (((n) + ((d)/2)) / (d))
```

19.43.2.15. DIV_ROUND_CLOSEST

```
#define DIV_ROUND_CLOSEST(
    x,
    divisor )
```

Макроопределение:

```
(
    {
        typedef(x) __x = x;
        typedef(divisor) __d = divisor;
        (((typeof(x))-1) > 0 ||
         ((typeof(divisor))-1) > 0 || (__x) > 0) ?
            (((__x) + ((__d) / 2)) / (__d)) :
            (((__x) - ((__d) / 2)) / (__d));
    }
)
```

19.43.2.16. DIV_ROUND_UP

```
#define DIV_ROUND_UP(
    n,
    d) (((n) + (d) - 1) / (d))
```

19.43.2.17. get_unaligned

```
#define get_unaligned(
    ptr )
```

Макроопределение:

```
((__force typeof(*(ptr)))(
    __builtin_choose_expr(sizeof(*(ptr)) == 1, *(ptr),
    __builtin_choose_expr(sizeof(*(ptr)) == 2, get_unaligned_le16((ptr)),
    __builtin_choose_expr(sizeof(*(ptr)) == 4, get_unaligned_le32((ptr)),
    __builtin_choose_expr(sizeof(*(ptr)) == 8, get_unaligned_le64((ptr)),
    __bad_unaligned_access_size())));
}))
```

19.43.2.18. INIT_TASK_NAME

```
#define INIT_TASK_NAME "Init"
```

19.43.2.19. KERN_DEBUG

```
#define KERN_DEBUG "MK_Debug:"
```

19.43.2.20. KERN_INFO

```
#define KERN_INFO "MK_Info:"
```

19.43.2.21. likely

```
#define likely(  
    x) __builtin_expect((x),1)
```

Встроенная функция, дающая компилятору подсказку о том, что условие x - истинно.

19.43.2.22. MAX

```
#define MAX(  
    a,  
    b) (((a)>(b))?(a):(b))
```

Не-типобезопасный макрос, возвращающий максимальное значение из 2х вариантов.

19.43.2.23. MAX_SAFE

```
#define MAX_SAFE(  
    a,  
    b)
```

Макроопределение:

```
{ __typeof__ (a) _a = (a); \  
  __typeof__ (b) _b = (b); \  
  _a > _b ? _a : _b; }
```

Типобезопасный макрос, возвращающий максимальное значение из 2х вариантов.



Может быть использован только в теле функции.

19.43.2.24. MEMBER_SIZE

```
#define MEMBER_SIZE(
    type,
    member ) (sizeof(((type *)0)->member))
```

Макрос, возвращающий размер поля типа.

19.43.2.25. MIN

```
#define MIN(
    a,
    b ) (((a)<(b))?a):(b))
```

Не-типобезопасный макрос, возвращающий минимальное значение из 2х вариантов.

19.43.2.26. MIN_SAFE

```
#define MIN_SAFE(
    a,
    b )
```

Макроопределение:

```
({ __typeof__ (a) _a = (a); \
  __typeof__ (b) _b = (b); \
  _a < _b ? _a : _b; })
```

Типобезопасный макрос, возвращающий минимальное значение из 2х вариантов.



Может быть использован только в теле функции.

19.43.2.27. printk

```
#define printk printf
```

19.43.2.28. put_unaligned

```
#define put_unaligned(
    val,
    ptr )
```

Макроопределение:

```
({
    void *__gu_p = (ptr);
    switch (sizeof(*(ptr))) {
    case 1:
        *(__gu_p) = (val);
    }
```

```

        *(u8 *)__gu_p = (__force u8)(val);        \
        break;                                  \
    case 2:                                       \
        put_unaligned_le16((__force u16)(val), __gu_p); \
        break;                                  \
    case 4:                                       \
        put_unaligned_le32((__force u32)(val), __gu_p); \
        break;                                  \
    case 8:                                       \
        put_unaligned_le64((__force u64)(val), __gu_p); \
        break;                                  \
    default:                                     \
        break;                                  \
    }                                           \
    (void)0; })

```

19.43.2.29. ROUND

```

#define ROUND(
    a,
    b) (((a) + (b) - 1) & ~((b) - 1))

```

Округление величины **a** до числа кратного заданному **b**.

19.43.2.30. roundup

```

#define roundup(
    x,
    y) (((x) + ((y) - 1)) / (y)) * (y)

```

Округление величины **x** до большего числа кратного заданному **y**.

19.43.2.31. SHELL_NAME

```

#define SHELL_NAME "Shell"

```

19.43.2.32. SHELL_PRIORITY

```

#define SHELL_PRIORITY 100

```

19.43.2.33. START_ONCE

```

#define START_ONCE do{static int Started = 0;if(Started == 1){return;} Started = 1;}while(0)

```

Макрос, обеспечивающий запуск функции только один раз (для функций, не имеющих возвращаемого значения).

19.43.2.34. START_ONCE_R

```

#define START_ONCE_R(

```

```
r ) do{static int Started = 0;if(Started == 1){return (r);} Started = 1;}while(0)
```

Макрос, обеспечивающий запуск функции только один раз (для функций, имеющих возвращаемое значение).

19.43.2.35. unlikely

```
#define unlikely(  
    x ) __builtin_expect((x),0)
```

Встроенная функция, дающая компилятору подсказку о том, что условие *x* - ложно.

19.43.2.36. VX_SUPERVISOR_MODE

```
#define VX_SUPERVISOR_MODE 0
```

19.43.3. Типы

19.43.3.1. dma_addr_t

```
typedef unsigned int dma_addr_t
```

19.43.3.2. FUNCPTR

```
typedef int(* FUNCPTR) (int)
```

19.43.3.3. irqreturn_t

```
typedef int irqreturn_t
```

19.43.3.4. resource_size_t

```
typedef unsigned int resource_size_t
```

19.43.4. Перечисления

19.43.4.1. STATUS

```
enum STATUS
```

Результат выполнения действия (функции).

Элементы перечислений

OK Успешное выполнение.

Продолжение на следующей странице

Элементы перечислений

ERROR	Неудача.
-------	----------

```
00065      {  
00066      OK = 0,  
00067      ERROR = -1  
00068 } STATUS;
```


19.44. Файл multimedia.dox

19.45. Файл names.h

Функции для работы с таблицами символов.

Функции

- `void * _findSTName` (const char *pName, char *pTypeOutputVar)
- `STATUS appendSymbol` (dynSymTbl *pTable, const char *pName, void *addr, char *type*)
- `dynSymTbl * createDynSymTbl` (size_t size)
- `STATUS registerSymTbl` (dynSymTbl *pTable)

19.45.1. Функции

19.45.1.1. `_findSTName()`

```
void* _findSTName (
    const char * pName,
    char * pTypeOutputVar )
```

Найти символ в таблицах символов и вернуть связанный с ним указатель.

Аргументы

pName

out	<i>pTypeOutputVar</i>	Указатель на char-переменную, в которую будет записан тип символа.
-----	-----------------------	--

Возвращает

Связанный с символом указатель или NULL, если символ не был найден.

19.45.1.2. `appendSymbol()`

```
STATUS appendSymbol (
    dynSymTbl * pTable,
    const char * pName,
    void * addr,
    char type )
```

Добавить символ в таблицу символов.

Аргументы

pTable Таблица символов.

pName Имя символа.

addr Адрес, к которому должен быть привязан символ.

type Тип символа, как у компилятора GCC ('T' для функции, 'D' для переменной).

Возвращает

OK при успехе, *ERROR* иначе.



Таблица имеет ограниченное кол-во "слотов" под символы.

19.45.1.3. createDynSymTbl()

```
dynSymTbl* createDynSymTbl (  
    size_t size )
```

Создать новую таблицу символов.

Аргументы

<i>size</i>	Максимальный размер таблицы символов.
-------------	---------------------------------------

Возвращает

Указатель на выделенную таблицу или *NULL* при ошибке.

19.45.1.4. registerSymTbl()

```
STATUS registerSymTbl (  
    dynSymTbl * pTable )
```

Подключить дополнительную таблицу символов (можно подключить неограниченное кол-во таблиц).

Аргументы

<i>pTable</i>	Выделенная и заполненная таблица символов.
---------------	--

Возвращает

OK при успехе, *ERROR* иначе.

19.46. Файл net.dox

19.47. Файл `pipelib.h`

Межпроцессорные каналы.

Функции

- *STATUS* `pipeDevCreate` (const char *name)

19.47.1. Функции

19.47.1.1. `pipeDevCreate()`

STATUS `pipeDevCreate` (
const char * name)

Создать псевдоустройство межпроцессорного канала.

Аргументы

name Имя устройства.

Возвращает

OK при успехе, *ERROR* иначе.

19.48. Файл pll.h

Работа с PLL.

Функции

- void `pll ()`
Вывести текущую конфигурацию PLL.
- unsigned int `pllAhbGet` (unsigned int n, bool *ok)
Получить текущую частоту шины AHB.
- unsigned int `pllApbGet` (unsigned int n, bool *ok)
Получить текущую частоту шины APB.
- unsigned int `pllAxiGet` (bool *ok)
Получить текущую частоту шины AXI.
- unsigned int `pllGet` (unsigned int n, unsigned int out, bool *ok)
Получить текущую частоту заданной PLL.
- unsigned int `pllSet` (unsigned int n, unsigned int out, unsigned int frequency)
Установить частоту для PLL.
- bool `pllUpdate ()`
Расчёт частот всех PLL и шин на основе заданной конфигурации.

Макросы обозначения PLL

- #define `PLL_1` 0
- #define `PLL_2` 1
- #define `PLL_3` 2
- #define `PLL_4` 3
- #define `PLL_5` 4
- #define `PLL_6` 5
- #define `PLL_7` 6
- #define `PLL_8` 7
- #define `PLL_9` 8
- #define `PLL_AUDIO` 1
- #define `PLL_CPU` 0
- #define `PLL_PERIPH0` 5
- #define `PLL_VE` 3

Макросы обозначения шин AHB, APB

- #define `AHB_1` 0
- #define `AHB_2` 1
- #define `APB_1` 0
- #define `APB_2` 1

19.48.1. Подробное описание

Проверка и настройка конфигурации распределения частот в процессоре ARM.

Подключение:

```
#include <pll.h>
```

См. также

Общее описание PLL в разделе *PLL – распределение тактовых частот*.

19.48.2. Макросы

19.48.2.1. АНВ_1

```
#define АНВ_1 0
```

Индекс для шины **АНВ1**.

19.48.2.2. АНВ_2

```
#define АНВ_2 1
```

Индекс для шины **АНВ2**.

19.48.2.3. АРВ_1

```
#define АРВ_1 0
```

Индекс для шины **АРВ1**.

19.48.2.4. АРВ_2

```
#define АРВ_2 1
```

Индекс для шины **АРВ2**.

19.48.2.5. PLL_1

```
#define PLL_1 0
```

Индекс для **PLL1**.

19.48.2.6. PLL_2

```
#define PLL_2 1
```

Индекс для **PLL2**.

19.48.2.7. PLL_3

```
#define PLL_3 2
```

Индекс для **PLL3**.

19.48.2.8. PLL_4

```
#define PLL_4 3
```

Индекс для **PLL4**.

19.48.2.9. PLL_5

```
#define PLL_5 4
```

Индекс для **PLL5**.

19.48.2.10. PLL_6

```
#define PLL_6 5
```

Индекс для **PLL6**.

19.48.2.11. PLL_7

```
#define PLL_7 6
```

Индекс для **PLL7**.

19.48.2.12. PLL_8

```
#define PLL_8 7
```

Индекс для **PLL8**.

19.48.2.13. PLL_9

```
#define PLL_9 8
```

Индекс для **PLL9**.

19.48.2.14. PLL_AUDIO

```
#define PLL_AUDIO 1
```

Синоним для **PLL2**.

19.48.2.15. PLL_CPU

```
#define PLL_CPU 0
```

Синоним для **PLL1**.

19.48.2.16. PLL_PERIPH0

```
#define PLL_PERIPH0 5
```

Синоним для **PLL6**.

19.48.2.17. PLL_VE

```
#define PLL_VE 3
```

Синоним для **PLL4**.

19.48.3. Функции**19.48.3.1. pll()**

```
void pll ( )
```

Функция выводит текущие настройки используемых частот в консоль.

19.48.3.2. pllAhbGet()

```
unsigned int pllAhbGet (  
    unsigned int n,  
    bool * ok )
```

Функция возвращает рассчитанную частоту шины **AHB**. Для получения актуального значения частоты имеет смысл вызвать *pllUpdate()*.

Аргументы

n Номер шины **AHB** из группы *макросов*.

ok *true* — шина присутствует и значение частоты вычислено.

Возвращает

Частота шины в Гц.

19.48.3.3. pllApbGet()

```
unsigned int pllApbGet (  
    unsigned int n,  
    bool * ok )
```

Функция возвращает рассчитанную частоту шины **APB**. Для получения актуального значения частоты имеет смысл вызвать *pllUpdate()*.

Аргументы

n Номер шины **APB** из группы *макросов*.

ok *true* — шина присутствует и значение частоты вычислено.

Возвращает

Частота шины в Гц.

19.48.3.4. pllAxiGet()

```
unsigned int pllAxiGet (  
    bool * ok )
```

Функция возвращает рассчитанную частоту шины **AXI**. Для получения актуального значения частоты имеет смысл вызвать *pllUpdate()*.

Аргументы

ok *true* — шина присутствует и значение частоты вычислено.

Возвращает

Частота шины в Гц.

19.48.3.5. pllGet()

```
unsigned int pllGet (  
    unsigned int n,  
    unsigned int out,  
    bool * ok )
```

Функция возвращает рассчитанную частоту для выбранной **PLL**. Для получения актуального значения частоты имеет смысл вызвать *pllUpdate()*.

Аргументы

n Номер **PLL** из группы *макросов*.

out Номер выхода **PLL**. Обычно в **PLL** существует только один выход (0). В некоторых случаях **PLL** имеет 2 выхода с разными частотами — 0 и 1.

ok Признак нормальной работы **PLL**. Стабильная работа в заданном диапазоне частот.

Возвращает

Частота выхода **PLL** в Гц.

19.48.3.6. pllSet()

```
unsigned int pllSet (  
    unsigned int n,  
    unsigned int out,  
    unsigned int frequency )
```

Функция рассчитывает и устанавливает коэффициенты выбранной **PLL** таким образом, чтобы итоговая частота была как можно ближе к заданной.

Добавлено в качестве эксперимента Функция реализована с ограниченным функционалом и находится в стадии тестирования.

Аргументы

n Номер **PLL** из группы *макросов*.

Продолжение на следующей странице

Аргументы (Продолжение.)	
<i>out</i>	Номер выхода PLL . Обычно в PLL существует только один выход (0). В некоторых случаях PLL имеет 2 выхода с разными частотами — 0 и 1. Установка выполняется только для выхода 0.
<i>frequency</i>	Заданная частота в Гц.

Возвращает

Установленное значение частоты в Гц если настройка прошла успешно, иначе 0.

19.48.3.7. pllUpdate()

bool pllUpdate ()

Рекомендуется рассчитывать заново всю конфигурацию частот перед получением данных о работе какой либо **PLL** или шины. Это обусловлено тем, что каждый модуль при настройке может изменить частоты некоторых модулей под себя. Окончательную конфигурацию частот можно посмотреть с помощью команды *pll()*.

Возвращает

true Модуль настроен, данные о частотах обновлены.

false Модуль не настроен, не прошла инициализация. Скорее всего не указан процессор в модуле **arch**.

19.49. Файл project.docx

19.50. Файл `pwm.h`

Управление линиями ШИМ (PWM).

Макросы номеров каналов ШИМ

Синонимы номеров каналов возможно использовать в качестве параметров функций **channel** для улучшения читаемости кода.

- `#define PWM_0 0`
- `#define PWM_1 1`

Выбор начального уровня сигнала

Данные макросы рекомендуется использовать в качестве параметров функций **activeHigh** для улучшения читаемости кода.

- `#define PWM_ACTIVE_HIGH true`
- `#define PWM_ACTIVE_LOW false`

Непрерывный режим

- **STATUS** `pwmInit` (unsigned int channel, unsigned int frequency, *bool* activeHigh)
Настройка канала ШИМ.
- **STATUS** `pwmSetFillFactor` (unsigned int channel, unsigned int fillFactor)
Задать коэффициент заполнения ШИМ.

Импульсный режим

- **STATUS** `pwmInitPulse` (unsigned int channel, *bool* activeHigh)
Инициализация канала ШИМ в импульсном режиме.
- unsigned int `pwmPulseDurationCalc` (unsigned int duration_ns)
Расчёт значения регистра периода для импульсного режима.
- **STATUS** `pwmPulseStart` (unsigned int channel, unsigned int period_value)
Запуск импульса заданной длительности в выбранном канале ШИМ.

19.50.1. Подробное описание

Настройка выводов **PWM**, в зависимости от выбранного процессора.

Подключение:

```
#include <pwm.h>
```

См. также

Общее описание работы с модулем ШИМ в разделе [PWM \(ШИМ\)](#).

19.50.2. Макросы

19.50.2.1. PWM_0

```
#define PWM_0 0
```

Канал ШИМ 0.

19.50.2.2. PWM_1

```
#define PWM_1 1
```

Канал ШИМ 1.

19.50.2.3. PWM_ACTIVE_HIGH

```
#define PWM_ACTIVE_HIGH true
```

Активный уровень сигнала – высокий.

19.50.2.4. PWM_ACTIVE_LOW

```
#define PWM_ACTIVE_LOW false
```

Активный уровень сигнала – низкий.

19.50.3. Функции

19.50.3.1. pwmInit()

```
STATUS pwmInit (  
    unsigned int channel,  
    unsigned int frequency,  
    bool activeHigh )
```

Функция настраивает режим работы канала ШИМ. Возможен повторный вызов функции для перенастройки канала в процессе работы.

Аргументы	
<i>channel</i>	Канал ШИМ.
<i>frequency</i>	Частота следования импульсов в герцах. Возможный диапазон частот от 0 Гц до 1 МГц. Коэффициент заполнения задаётся с помощью <i>pwmSetFillFactor()</i> .
<i>activeHigh</i>	Уровень активной части сигнала: <i>true</i> – высокий уровень, иначе низкий уровень сигнала.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.50.3.2. pwmInitPulse()

```
STATUS pwmInitPulse (
    unsigned int channel,
    bool activeHigh )
```

Функция настраивает канал шим на запуск импульсов заданной длительности по команде *pwmPulseStart()*.

Аргументы	
<i>channel</i>	Канал ШИМ.
<i>activeHigh</i>	Уровень активной части сигнала: <i>true</i> – высокий уровень, иначе низкий уровень сигнала.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.50.3.3. pwmPulseDurationCalc()

```
unsigned int pwmPulseDurationCalc (
    unsigned int duration_ns )
```

Функция рассчитывает значение, записываемое в регистр определения периода импульса. Рассчитанное значение следует использовать в функции запуска импульса *pwmPulseStart()*.

Аргументы	
<i>duration_ns</i>	Длительность импульса в наносекундах. Возможный диапазон значений от 50 нс до 2,73 мс.

Возвращает

Значение для функции *pwmPulseStart()*.

19.50.3.4. pwmPulseStart()

```
STATUS pwmPulseStart (
    unsigned int channel,
    unsigned int period_value )
```

Функция запускает одиночный импульс в канале ШИМ. Канал должен быть сконфигурирован с помощью *pwmInitPulse()*. Для обеспечения максимального быстродействия при вызове функции в прерываниях следует использовать заранее рассчитанные с помощью функции *pwmPulseDurationCalc()* значения периода, которые будут записаны непосредственно в регистр аппаратного модуля ШИМ.

Аргументы

<i>channel</i>	Канал ШИМ.
<i>period_value</i>	Значение регистра периода ШИМ, рассчитанное с помощью <i>pwmPulseDurationCalc()</i> .

Возвращает

OK при успешном запуске, иначе *ERROR*.

19.50.3.5. *pwmSetFillFactor()*

STATUS *pwmSetFillFactor* (
 unsigned int *channel*,
 unsigned int *fillFactor*)

Функция устанавливает коэффициент заполнения ШИМ в процентах (всего 100 градаций). Период ШИМ предварительно задаётся в настройках *pwmInit()*.

Аргументы

<i>channel</i>	Канал ШИМ.
<i>fillFactor</i>	Коэффициент заполнения от 0 до 100 (максимальное заполнение).

Возвращает

OK если коэффициент успешно выставлен, иначе *ERROR*.

19.51. Файл ringbuffer.h

Кольцевой буфер.

Структуры данных

- struct *tRingBuffer*
Структура кольцевого буфера.

Управление буферами

- *STATUS deleteRingBuffer* (*tRingBuffer* *buf)
Удалить кольцевой буфер.
- *tRingBuffer* * *newRingBuffer* (unsigned int dataSize, unsigned int maxCount)
Создать кольцевой буфер.

Запись / чтение данных

- int *ringBufferCounter* (const *tRingBuffer* *buf)
Количество записей в буфере.
- *STATUS ringBufferFlush* (*tRingBuffer* *buf)
Очистка буфера.
- *STATUS ringBufferRead* (void *data, *tRingBuffer* *buf, int timeout)
Чтение элемента данных из буфера.
- *STATUS ringBufferWrite* (*tRingBuffer* *buf, const void *data, int timeout)
Запись элемента данных в буфер.

19.51.1. Подробное описание

Кольцевой буфер – альтернатива очереди сообщений для использования в прерываниях. Буфер может содержать заданное количество элементов фиксированной длины.

19.51.2. Функции

19.51.2.1. deleteRingBuffer()

```
STATUS deleteRingBuffer (  
    tRingBuffer * buf )
```

Функция освобождает память и удаляет буфер.

Аргументы

buf Удаляемый буфер.

Возвращает

OK в случае успешного удаления буфера, иначе *ERROR*.

19.51.2.2. newRingBuffer()

```
tRingBuffer* newRingBuffer (  
    unsigned int dataSize,  
    unsigned int maxCount )
```

Функция создаёт буфер и выделяет память под данные.

Аргументы

<i>dataSize</i>	Размер одного элемента данных в буфере.
<i>maxCount</i>	Количество элементов в буфере. Количество элементов должно быть больше одного.

Возвращает

Указатель на созданный буфер.

19.51.2.3. ringBufferCounter()

```
int ringBufferCounter (  
    const tRingBuffer * buf )
```

Функция возвращает количество не прочитанных записей в кольцевом буфере. Возвращаемое значение должно быть строго не отрицательным. Отрицательное число в результате говорит о некорректной работе буфера.

Аргументы

<i>buf</i>	Указатель на кольцевой буфер.
------------	-------------------------------

Возвращает

Количество записей.

19.51.2.4. ringBufferFlush()

```
STATUS ringBufferFlush (  
    tRingBuffer * buf )
```

Функция выравнивает счётчики входящих и исходящих записей колцевого буфера и обнуляет счётчик имеющихся записей.

Аргументы

<i>buf</i>

Возвращает

Всегда *OK*.

19.51.2.5. ringBufferRead()

```
STATUS ringBufferRead (  
    void * data,  
    tRingBuffer * buf,  
    int timeout )
```

Функция читает один элемент данных из кольцевого буфера и декрементирует счётчик данных. Чтение возможно только если в буфере есть не считанные данные. Если задан параметр **timeout** больший нуля – функция будет ждать появления данных заданное количество времени.

ВАЖНО! При чтении данных в обработчиках прерываний следует использовать значение таймаута *NO_WAIT*.

Аргументы

<i>data</i>	Указатель на данные, в который будет считан элемент буфера.
<i>buf</i>	Указатель на кольцевой буфер.
<i>timeout</i>	Таймаут ожидания наличия данных в тиках системного таймера. Если данные не появились в течение заданного времени функция вернёт ошибку. В качестве параметра можно также использовать значения <i>NO_WAIT</i> и <i>WAIT_FOREVER</i> .

Возвращает

OK в случае удачного чтения, либо *ERROR* если данных нет и они не появились за заданный промежуток времени.

19.51.2.6. ringBufferWrite()

```
STATUS ringBufferWrite (  
    tRingBuffer * buf,  
    const void * data,  
    int timeout )
```

Функция записывает один элемент данных в кольцевой буфер и инкрементирует входной счётчик данных. Запись возможна только пока в буфере есть свободное место. Если место закончилось необходимо прочитать все записи либо очистить буфер.

Аргументы

<i>buf</i>	Указатель на кольцевой буфер.
<i>data</i>	Записываемые данные – указатель на записываемый элемент.

Продолжение на следующей странице

Аргументы (Продолжение.)

timeout Таймаут ожидания появления свободного места в буфере в тиках системного таймера. Если свободное место для записи не появилось в течение заданного времени функция вернёт ошибку. В качестве параметра можно также использовать значения *NO_WAIT* и *WAIT_FOREVER*.

ВАЖНО! При записи данных в обработчиках прерываний следует использовать значение таймаута *NO_WAIT*.

Возвращает

OK в случае удачной записи, либо *ERROR* при переполнении буфера.

19.52. Файл semlib.h

Управление семафорами.

Структуры данных

- struct *Sem_Id*
Структура семафора.

Макросы

- #define *NO_WAIT* (0)
- #define *SEM_DELETE_SAFE* 0x4
Защита задачи.
- #define *SEM_INVERSION_SAFE* 0x8
Инверсия приоритетов.
- #define *SEM_MARKER* (0xA525E727)
- #define *SEM_Q_FIFO* 0x0
Порядок обработки задач - FIFO.
- #define *SEM_Q_PRIORITY* 0x1
Порядок обработки задач - Приоритет.
- #define *WAIT_FOREVER* (-1)

Определения типов

- typedef *Sem_Id* * *SEM_ID*
Указатель на семафор.

Перечисления

- enum *SEM_B_STATE* { *SEM_EMPTY* = 0 , *SEM_FULL* = 1 }
- enum *SEM_CLASS* { *scSemB* , *scSemC* , *scSemM* }
- enum *SEM_FLUSH_STATE* { *sfsNoFlush* = 0 , *sfsFlush* = 1 , *sfsUnblocked* = 2 }
Способ подъема семафора.

Макросы и надстройки для быстрой инициализации семафоров.

- #define *INIT_STATIC_MUTEX*(SemOptions)
Инициализация мьютекса в сегменте данных. Инициализация производится в следующем виде: 'SEM_ID Mutex = INIT_STATIC_MUTEX(SEM_Q_FIFO);'.
- #define *INIT_STATIC_MUTEX_DEFAULT*()
Инициализация мьютекса в сегменте данных. Инициализация производится в следующем виде: 'SEM_ID Mutex = INIT_STATIC_MUTEX_DEFAULT();'.
- #define *INIT_STATIC_SEM*(SemOptions, SemState)
Инициализация бинарного семафора в сегменте данных. Инициализация производится в следующем виде: 'SEM_ID Sem = INIT_STATIC_SEM(SEM_Q_FIFO, SEM_FULL);'.
- #define *INIT_STATIC_SEM_DEFAULT*()
Инициализация бинарного семафора в сегменте данных. Инициализация производится в следующем виде: 'SEM_ID Sem = INIT_STATIC_SEM_DEFAULT();'.
- *SEM_ID semBCreate_Default* (void)
- *SEM_ID semMCreate_Default* (void)

Основные функции для работы с семафорами.

- *SEM_ID semBCreate* (int options, *SEM_B_STATE* initialState)
Создать двоичный семафор.
- int *semCCount* (*SEM_ID* semId)
- *SEM_ID semCCreate* (int options, int initialCount)

- *Создать целочисленный семафор.*
• `STATUS semDelete (SEM_ID semId)`
- *Удалить семафор.*
• `STATUS semFlush (SEM_ID semId)`
- *Освободить все задачи, ожидающие захвата бинарного семафора.*
• `STATUS semGive (SEM_ID semId)`
- *Освободить семафор.*
• `SEM_ID semMCreate (int options)`
- *Создать семафор взаимного исключения (Mutex).*
• `STATUS semMCUnblock (SEM_ID semId)`
- `STATUS semTake (SEM_ID semId, int timeout)`
Попытаться захватить семафор.

19.52.1. Подробное описание

Семафоры в *MULTEX-ARM* – это основной механизм синхронизации задач в реальном времени и организации взаимноисключающего доступа задач к общим ресурсам.

См. также

Более подробное описание в главе *Семафоры*.

Пример использования семафора в простой задаче:

```
SEM_ID sem = NULL;
bool stop;

// Задача в отдельной функции
int taskSimple (int dummy) {
    // Инициализация переменных задачи (если есть)
    void *p = malloc (size);

    while (!stop) {
        // Ожидание семафора
        semTake (sem, WAIT_FOREVER);
        // ... тело задачи
    }

    // Освобождение выделенной памяти
    free (p);
    // Завершение задачи
    return 0;
}

// Создание и запуск задачи
void testSimple () {
    sem = semBCreate (SEM_Q_FIFO, SEM_EMPTY);
    stop = false;
    taskSpawn ("task simple"{}", 10, 0, 0, taskSimple, 0);
}

// Некое внешнее событие
void testRelease () {
    // Освободить семафор
    semGive (sem);
}
```

19.52.2. Макросы

19.52.2.1. INIT_STATIC_MUTEX

```
#define INIT_STATIC_MUTEX(  
    SemOptions )
```

Макроопределение:

```
\&(amp;Sem_Id){.marker = SEM_MARKER, .semclass = scSemM, .state = SEM_FULL, \  
.options = SemOptions, .flushed=FALSE, .count=0, .owner=NULL}
```

19.52.2.2. INIT_STATIC_MUTEX_DEFAULT

```
#define INIT_STATIC_MUTEX_DEFAULT()
```

Макроопределение:

```
\&(amp;Sem_Id){.marker = SEM_MARKER, .semclass = scSemM, .state = SEM_FULL, \  
.options = SEM_Q_FIFO, .flushed=FALSE, .count=0, .owner=NULL}
```

19.52.2.3. INIT_STATIC_SEM

```
#define INIT_STATIC_SEM(  
    SemOptions,  
    SemState )
```

Макроопределение:

```
\&(amp;Sem_Id){.marker = SEM_MARKER, .semclass = scSemB, .state = SemState, \  
.options = SemOptions, .flushed=FALSE, .count=0, .owner=NULL}
```

19.52.2.4. INIT_STATIC_SEM_DEFAULT

```
#define INIT_STATIC_SEM_DEFAULT()
```

Макроопределение:

```
\&(amp;Sem_Id){.marker = SEM_MARKER, .semclass = scSemB, .state = SEM_FULL, \  
.options = SEM_Q_FIFO, .flushed=FALSE, .count=0, .owner=NULL}
```

19.52.2.5. NO_WAIT

```
#define NO_WAIT (0)
```

Не ждать.

19.52.2.6. SEM_DELETE_SAFE

```
#define SEM_DELETE_SAFE 0x4
```

Защищать задачу от удаления, если она захватила семафор.



Применимо только для мьютексов.

19.52.2.7. SEM_INVERSION_SAFE

```
#define SEM_INVERSION_SAFE 0x8
```

Разрешить инверсию приоритетов для мьютекса.

19.52.2.8. SEM_MARKER

```
#define SEM_MARKER (0xA525E727)
```

Сигнатура семафора.

19.52.2.9. SEM_Q_FIFO

```
#define SEM_Q_FIFO 0x0
```

Обрабатывать семафоры с учетом модели **FIFO** задач.



Возможно не работает.

19.52.2.10. SEM_Q_PRIORITY

```
#define SEM_Q_PRIORITY 0x1
```

Обрабатывать семафоры с учетом приоритетов задач.



Возможно не работает.

19.52.2.11. WAIT_FOREVER

```
#define WAIT_FOREVER (-1)
```

Ждать до успеха.

19.52.3. Типы

19.52.3.1. SEM_ID

```
typedef Sem_Id* SEM_ID
```

Указатель на структуру семафора *Sem_Id*.

19.52.4. Перечисления

19.52.4.1. SEM_B_STATE

```
enum SEM_B_STATE
```

Состояние семафора.

Элементы перечислений

SEM_EMPTY Семафор закрыт (заблокирован, захвачен).

SEM_FULL Семафор открыт (разблокирован, свободен)

```
00095                {
00096 SEM_EMPTY = 0,
00097 SEM_FULL = 1
00098 } SEM_B_STATE;
```

19.52.4.2. SEM_CLASS

```
enum SEM_CLASS
```

Класс семафора.

Элементы перечислений

scSemB Бинарный семафор.

scSemC Семафор-счетчик.

scSemM Мьютекс.

```

00103      {
00104  scSemB,
00105  scSemC,
00106  scSemM,
00107 } SEM_CLASS;

```

19.52.4.3. SEM_FLUSH_STATE

enum *SEM_FLUSH_STATE*

Способ подъёма семафора - по какой причине задачу выдернули с ожидания семафора.



Устаревшие значения, ныне фактически не используются.

Элементы перечислений

sfsNoFlush	Вышло время на ожидание захвата задачи. Семафор не считается захваченным.
sfsFlush	Устаревшее значение, не должно возникать.
sfsUnblocked	Мьютекс или семафор-счетчик был принудительно сброшен. Семафор не считается захваченным.

```

00114      {
00115  sfsNoFlush = 0,
00116  sfsFlush = 1,
00117  sfsUnblocked = 2,
00118 } SEM_FLUSH_STATE;

```

19.52.5. Функции

19.52.5.1. semBCreate()

```

SEM_ID semBCreate (
    int options,
    SEM_B_STATE initialState )

```

Функция создает двоичный семафор. Такой семафор может быть использован для синхронизации задач, при этом начальное состояние семафора следует задать закрытым (*SEM_EMPTY*). При использовании его для защиты доступа к ресурсам общего пользования (взаимоисключения) следует создавать семафор изначально открытым (*SEM_FULL*).

Аргументы

<i>options</i>	Способ помещения в очередь задач, ждущих у семафора. Следует использовать следующие predefined константы SEM_Q_FIFO и SEM_Q_PRIORITY .
<i>initialState</i>	Начальное состояние семафора. Если семафор открыт (SEM_FULL), то первая захватившая его задача закрывает и продолжает выполняться. Если задача пытается захватить закрытый семафор, то ее выполнение задерживается до тех пор, пока другая задача не откроет этот семафор.

Возвращает

SEM_ID Идентификатор созданного семафора или 0 при неудаче.

19.52.5.2. semBCreate_Default()

```
SEM_ID semBCreate_Default (  
    void )
```

Создать бинарный семафор с параметрами по-умолчанию (**FIFO**, свободный).

Возвращает

Указатель на семафор или *NULL*.

19.52.5.3. semCCount()

```
int semCCount (  
    SEM_ID semId )
```

Получить значение счетчика для семафора-счетчика.

Аргументы

<i>semId</i>	Целевой семафор.
--------------	------------------

Возвращает

Значение счетчика для семафора-счетчика.

19.52.5.4. semCCreate()

```
SEM_ID semCCreate (  
    int options,  
    int initialCount )
```

Отличие целочисленного семафора от двоичного в том, что он имеет внутренний счётчик,

фиксирующий, сколько раз семафор был открыт. При очередном открытии семафора счётчик увеличивается, при закрытии – уменьшается. Когда счётчик доходит до нуля, задача, пытающаяся закрыть семафор, будет приостановлена. Такой тип семафора удобен для организации множественного доступа к буферам ограниченного размера, стекам, очередям, а также нескольким копиям каких-либо ресурсов. Начальное значение при этом задают равным максимальному размеру ресурса, что гарантирует его от переполнения. Внутренний механизм очередей *MULTEX-ARM* использует этот тип семафора при их организации.

Аргументы

<i>options</i>	Способ помещения в очередь задач, ждущих у семафора. Следует использовать следующие predetermined константы <i>SEM_Q_FIFO</i> и <i>SEM_Q_PRIORITY</i> .
<i>initialCount</i>	Начальное значение внутреннего счётчика открываний.

Возвращает

Указатель на семафор или *NULL*.

19.52.5.5. semDelete()

STATUS semDelete (
 SEM_ID semId)

Функция удаляет указанный семафор и освобождает задачи, ожидающие его освобождения.

Аргументы

<i>semId</i>	Идентификатор семафора, с которым производится действие.
--------------	--

Возвращает

OK при успешном выполнении.

Если идентификатор семафора, передаваемый функциям управления семафорами не является правильным или ссылается на удаленный семафор, функция возвращает значение *ERROR*.

19.52.5.6. semFlush()

STATUS semFlush (
 SEM_ID semId)

Функция разблокирует все задачи, ожидающие у этого семафора. Семафор при этом остается закрытым.



Не следует использовать семафоры в обработчиках прерываний!

Аргументы

semId Идентификатор семафора, с которым производится действие.

Возвращает

OK при успешном выполнении или *ERROR* при неудаче.



Только для семафора типа *scSemB*.

19.52.5.7. semGive()

STATUS semGive (
SEM_ID *semId*)

Функция освобождает семафор, обеспечивая тем самым возможность другим задачам его захватить. Если в карусели задержанных задач имеются задачи, ждущие это событие, то первая из них становится в карусель активных задач. При этом, если ее приоритет выше выполняемой задачи, производится немедленное переключение процессора на ее выполнение.



Не следует использовать семафоры в обработчиках прерываний!

Аргументы

semId Идентификатор семафора, с которым производится действие.

Возвращает

OK при успешном выполнении или *ERROR* при неудаче.

19.52.5.8. semMCreate()

SEM_ID semMCreate (
int *options*)

Mutex работает так же, как двоичный с открытым начальным состоянием: первая же захватившая его задача блокирует все дальнейшие попытки его захвата до тех пор, пока захватившая задача не освободит его.

Этот тип семафора имеет следующие особенности:

- Он может быть открыт только той задачей, которая его закрыла.
- Он не может быть открыт в обработчике прерывания.
- К нему нельзя применять функцию *semFlush()*.
- Он имеет дополнительную опцию *SEM_INVERSION_SAFE*, которая включает алгоритм наследования приоритета, заключающийся в том, что захватившая семафор задача автоматически получает приоритет, равный самому высокому приоритету из всех задач, ждущих у этого семафора.
- Он имеет дополнительную опцию *SEM_DELETE_SAFE*, которая защищает задачу от удаления на время захвата ею этого семафора.
- Он имеет возможность рекурсивного захвата, при этом задача может захватить семафор несколько раз подряд. При этом для того, чтобы другая задача смогла захватить этот семафор, захватившая задача должна освободить его столько же раз, сколько раз его захватила.

Аргументы

options Следует использовать следующие предопределенные константы *SEM_Q_FIFO* или *SEM_Q_PRIORITY* в комбинации с возможными опциями *SEM_INVERSION_SAFE* и *SEM_DELETE_SAFE*.

Возвращает

Указатель на семафор или *NULL*.

19.52.5.9. semMCreate_Default()

```
SEM_ID semMCreate_Default (  
    void )
```

Создать семафор-мьютекс с параметрами по-умолчанию (**FIFO**).

Возвращает

Указатель на семафор или *NULL*.

19.52.5.10. semMCUnblock()

```
STATUS semMCUnblock (  
    SEM_ID semId )
```

Разблокировать все задачи, ожидающие захвата мьютекса или семафора-счетчика.

Аргументы

semId Целевой семафор.

Возвращает

OK при успехе, *ERROR* иначе.

`semTake` в таком случае вернет *ERROR*. Нужно для удаления семафоров, на которых кто-нибудь сидит.



Только для мьютексов или семафоров-счетчиков (`scSemM` и `scSemC`).

19.52.5.11. `semTake()`

```
STATUS semTake (  
    SEM_ID semId,  
    int timeout )
```

Функция делает попытку захватить семафор. Если он на этот момент открыт, то задача захватывает его и продолжается. Если семафор был закрыт или захвачен другой задачей, то задача приостанавливается до его открытия, т.е. ожидает у этого семафора. Время ожидания может быть задано непосредственно в тиках таймера, либо быть неограниченным (*WAIT_FOREVER*), либо отсутствовать (*NO_WAIT*).



Не следует использовать семафоры в обработчиках прерываний!

Аргументы

<i>semId</i>	Идентификатор семафора, с которым производится действие.
<i>timeout</i>	Кол-во тиков таймера, в течении которых задача будет ждать захвата. Дополнительные возможные значения: <ul style="list-style-type: none">• <i>NO_WAIT</i> - не ждать вовсе.• <i>WAIT_FOREVER</i> - ждать до успеха.

Возвращает

OK - семафор был захвачен ИЛИ если бинарный семафор был разблокирован через `semFlush()`.

ERROR - семафор не был захвачен.

-2 - задача обрабатывала прерывание и мгновенный захват не удался.

-3 - установлен флаг блокировки задачи от переключения и мгновенный захват не удался.

19.53. Файл `setjmp.h`

Нелокальные переходы.

Структуры данных

- struct `jmp_buf`
- struct `REG_SET`

Функции

- `noreturn` void `longjmp` (`jmp_buf` env, int val)
- int `setjmp` (`jmp_buf` env)

19.53.1. Подробное описание

В файле описаны функции нелокальных переходов по стандарту [C11 standard 7.13](#).

См. также

Общее описание в главе [Нелокальные переходы](#).

19.53.2. Функции

19.53.2.1. `longjmp()`

```
noreturn void longjmp (  
    jmp_buf env,  
    int val )
```

Перейти к сохраненному контексту.

Аргументы

`env` Точка сохранения контекста.

`val` Значение, которое будет передано в функцию `setjmp()`.

19.53.2.2. `setjmp()`

```
int setjmp (  
    jmp_buf env )
```

Сохранить контекст для дальнейшего перехода.

Аргументы

`env` Буфер для сохранения контекста.

Возвращает

0, если функция установила точку, **не-0**, если возврат функции был вызван с помощью *longjmp()*.

19.54. Файл shell.dox

19.55. Файл shell.h

Терминал.

Функции

- void *printHeader* (void)
- *TASK_ID shell* (bool *printHeader*)
- int *shellTask* (int *printHeader*)

19.55.1. Функции

19.55.1.1. printHeader()

```
void printHeader (  
    void )
```

Вывести в stdout приветственное сообщение терминала.

19.55.1.2. shell()

```
TASK_ID shell (  
    bool printHeader )
```

Запустить терминал в отдельной задаче.

Функция запускает *shellTask()* в отдельной задаче.

Аргументы

<i>printHeader</i>	Следует ли выводить приветственное сообщение при запуске терминала.
--------------------	---

Возвращает

Идентификатор запущенной задачи.

19.55.1.3. shellTask()

```
int shellTask (  
    int printHeader )
```

Запустить терминал в текущей задаче.

Задача будет обслуживать терминал для установленных для неё stdin и stdout. Задача будет завершена при получении команды 'quit' на терминале (при этом stdout и stdin задачи, если они не являются системными stdout и stdin, будут закрыты).

Аргументы

<i>printHeader</i>	Следует ли выводить приветственное сообщение при запуске терминала.
--------------------	---

Возвращает

Никогда не возвращает значение.

19.56. Файл signal.h

Обработка сигналов (C11 + частично POSIX).

Структуры данных

- struct *sigaction*
Структура обработчика сигнала.
- struct *siginfo*
Структура данных сигнала.
- union *sigval*

Определения типов

- typedef int *sig_atomic_t*
- typedef void(* *sig_handle*) (int)
- typedef *uint32_t sigset_t*

Функции

- int *kill* (*TASK_ID* ti, int sig)
Отправить сигнал процессу.
- int *raise* (int sig)
Отправить сигнал текущему процессу.
- int *sigaction* (int signum, const struct *sigaction* *act, struct *sigaction* *oldact)
Установить обработчик сигнала.
- *sig_handle signal* (int sig, *sig_handle* func)
Установить обработчик сигнала.
- *TASK_ID wait* (int *status_p)
Приостановить выполнение текущей задачи.

Аргументы и возвращаемые значения для функции signal()

- #define *SIG_DFL* ((void *)0)
- #define *SIG_ERR* ((void *)-1)
- #define *SIG_IGN* ((void *)1)

Стандартные для Си сигналы

- #define *SIGABRT* 6
Запрос на ненормальное завершение программы.
- #define *SIGFPE* 8
Ошибка арифметики.
- #define *SIGILL* 4
Выполнение недопустимой инструкции.
- #define *SIGINT* 2
Получение интерактивного сигнала.
- #define *SIGSEGV* 11
Ошибка доступа к памяти.
- #define *SIGTERM* 15
Запрос на завершение программы.

Дополнительные сигналы (в т.ч. POSIX)

- #define *SIGALRM* 14
- #define *SIGBUS* 10
- #define *SIGCHLD* 18
- #define *SIGCONT* 25
- #define *SIGEMT* 7
- #define *SIGFMT* 19
- #define *SIGHUP* 1
- #define *SIGKILL* 9
- #define *SIGNONE* 0
- #define *SIGPIPE* 13
- #define *SIGPOLL* 22
- #define *SIGPROF* 29
- #define *SIGQUIT* 3
- #define *SIGRTMAX* 24
- #define *SIGRTMIN* 24
- #define *SIGSTOP* 23
- #define *SIGSYS* 12
- #define *SIGTRAP* 5
- #define *SIGTSTP* 20
- #define *SIGTTIN* 26
- #define *SIGTTOU* 27
- #define *SIGURG* 21
- #define *SIGUSR1* 16
- #define *SIGUSR2* 17
- #define *SIGVTALRM* 28
- #define *SIGXCPU* 30
- #define *SIGXFSZ* 31

Значения поля *sa_flags* в структуре *sigaction*

- #define *SA_INTERRUPT* 0x0008
- #define *SA_NOCLDSTOP* 0x0001
- #define *SA_ONSTACK* 0x0004
- #define *SA_RESETHAND* 0x0010
- #define *SA_SIGINFO* 0x0002

Значения *si_code* возвращаемые *siginfo*

- #define *SI_ASYNCIO* 4
- #define *SI_KILL* 1
- #define *SI_MESGQ* 5
- #define *SI_QUEUE* 2
- #define *SI_SYNC* 0
- #define *SI_TIMER* 3

Структуры и макросы для POSIX-совместимой обработки сигналов

- #define *SIG_BLOCK* 1
- #define *SIG_SETMASK* 3
- #define *SIG_UNBLOCK* 2
- typedef struct *siginfo* *siginfo_t*
 Структура данных сигнала.

Операции над наборами сигналов

Функции, позволяющие создавать наборы сигналов в стандарте **POSIX** для формирования поля *sa_mask* в структуре *sigaction*.

- `int sigaddset (sigset_t *set, int signum)`
Добавить сигнал к набору сигналов.
- `int sigdelset (sigset_t *set, int signum)`
Удалить сигнал из набора сигналов.
- `int sigemptyset (sigset_t *set)`
Проинициализировать набор сигналов.
- `int sigfillset (sigset_t *set)`
Проинициализировать набор сигналов.
- `int sigismember (sigset_t *set, int signum)`
Проверить, является ли сигнал членом набора сигналов.

19.56.1. Подробное описание

См. также

[C11 standard 7.14.](#)

Подробнее о сигналах см. в главе [Сигналы](#).

19.56.2. Макросы

19.56.2.1. SA_INTERRUPT

```
#define SA_INTERRUPT 0x0008
```

Don't restart the function.

19.56.2.2. SA_NOCLDSTOP

```
#define SA_NOCLDSTOP 0x0001
```

Do not generate SIGCHLD when children stop.

19.56.2.3. SA_ONSTACK

```
#define SA_ONSTACK 0x0004
```

Run on sigstack.

19.56.2.4. SA_RESETHAND

```
#define SA_RESETHAND 0x0010
```

Reset the handler, like sysV.

19.56.2.5. SA_SIGINFO

```
#define SA_SIGINFO 0x0002
```

Pass additional siginfo structure.

19.56.2.6. SI_ASYNCIO

```
#define SI_ASYNCIO 4
```

signal from completion of an async I/O.

19.56.2.7. SI_KILL

```
#define SI_KILL 1
```

signal from `.1 kill()` function.

19.56.2.8. SI_MESGQ

```
#define SI_MESGQ 5
```

signal from arrival of a message.

19.56.2.9. SI_QUEUE

```
#define SI_QUEUE 2
```

signal from `.4 sigqueue()` function.

19.56.2.10. SI_SYNC

```
#define SI_SYNC 0
```

(Not posix) generated by hardware.

19.56.2.11. SI_TIMER

```
#define SI_TIMER 3
```

signal from expiration of a `.4 timer`.

19.56.2.12. SIG_BLOCK

```
#define SIG_BLOCK 1
```

19.56.2.13. SIG_DFL

```
#define SIG_DFL ((void *)0)
```

Значение для установки поведения по умолчанию.

19.56.2.14. SIG_ERR

```
#define SIG_ERR ((void *)-1)
```

Возвращаемое значение, свидетельствующее об ошибке.

19.56.2.15. SIG_IGN

```
#define SIG_IGN ((void *)1)
```

Значение для игнорирования сигнала.

19.56.2.16. SIG_SETMASK

```
#define SIG_SETMASK 3
```

19.56.2.17. SIG_UNBLOCK

```
#define SIG_UNBLOCK 2
```

19.56.2.18. SIGABRT

```
#define SIGABRT 6
```

Ненормальное завершение работы. Например такое, которое генерируется функцией **abort**.

19.56.2.19. SIGALRM

```
#define SIGALRM 14
```

Истечение времени, заданного `alarm()`.

19.56.2.20. SIGBUS

```
#define SIGBUS 10
```

Ошибка шины памяти.

19.56.2.21. SIGCHLD

```
#define SIGCHLD 18
```

Дочерний процесс завершен или остановлен.

19.56.2.22. SIGCONT

```
#define SIGCONT 25
```

Возобновление приостановленного процесса.

19.56.2.23. SIGEMT

```
#define SIGEMT 7
```

Инструкция EMT.

19.56.2.24. SIGFMT

```
#define SIGFMT 19
```

Ошибка формата стека.

19.56.2.25. SIGFPE

```
#define SIGFPE 8
```

Ошибка операции с плавающей точкой, например переполнение, или деление на ноль.

19.56.2.26. SIGHUP

```
#define SIGHUP 1
```

Освобождение линии терминала.

19.56.2.27. SIGILL

```
#define SIGILL 4
```

Неправильный образ функции, например, неправильная инструкция. Такой сигнал может быть выдан в результате повреждения кода или при попытке исполнить данные вместо кода.

19.56.2.28. SIGINT

```
#define SIGINT 2
```

Сигнал-прерывание. Обычно генерируется пользователем приложения.

19.56.2.29. SIGKILL

```
#define SIGKILL 9
```

Немедленное завершение работы. Не может быть обработан, пойман или проигнорирован.

19.56.2.30. SIGNONE

```
#define SIGNONE 0
```

Особое значение для отсутствия сигнала.

19.56.2.31. SIGPIPE

```
#define SIGPIPE 13
```

Запись в разорванное соединение.

19.56.2.32. SIGPOLL

```
#define SIGPOLL 22
```

Событие, отслеживаемое poll().

19.56.2.33. SIGPROF

```
#define SIGPROF 29
```

Истечение таймера профилирования.

19.56.2.34. SIGQUIT

```
#define SIGQUIT 3
```

Прерывание с аварийным завершением.

19.56.2.35. SIGRTMAX

```
#define SIGRTMAX 24
```

Минимальное значение для сигналов реального времени.

19.56.2.36. SIGRTMIN

```
#define SIGRTMIN 24
```

Максимальное значение для сигналов реального времени.

19.56.2.37. SIGSEGV

```
#define SIGSEGV 11
```

Ошибка доступа к памяти. Программа пытается использовать ту область памяти, которая ей не принадлежит.

19.56.2.38. SIGSTOP

```
#define SIGSTOP 23
```

Нетерминальная остановка. Не может быть обработан, пойман или проигнорирован.

19.56.2.39. SIGSYS

```
#define SIGSYS 12
```

Неправильный системный вызов.

19.56.2.40. SIGTERM

```
#define SIGTERM 15
```

Запрос на прекращение работы программы.

19.56.2.41. SIGTRAP

```
#define SIGTRAP 5
```

Отладка.

19.56.2.42. SIGTSTP

```
#define SIGTSTP 20
```

Остановка с терминала.

19.56.2.43. SIGTTIN

```
#define SIGTTIN 26
```

Попытка чтения с терминала фоновым процессом.

19.56.2.44. SIGTTOU

```
#define SIGTTOU 27
```

Попытка записи на терминал фоновым процессом.

19.56.2.45. SIGURG

```
#define SIGURG 21
```

Получены срочные данные.

19.56.2.46. SIGUSR1

```
#define SIGUSR1 16
```

Пользовательский сигнал 1.

19.56.2.47. SIGUSR2

```
#define SIGUSR2 17
```

Пользовательский сигнал 2.

19.56.2.48. SIGVTALRM

```
#define SIGVTALRM 28
```

Истечение виртуального таймера.

19.56.2.49. SIGXCPU

```
#define SIGXCPU 30
```

Процесс привисил лимит процессорного времени.

19.56.2.50. SIGXFSZ

```
#define SIGXFSZ 31
```

Процесс превысил допустимый размер файла.

19.56.3. Типы

19.56.3.1. sig_atomic_t

```
typedef int sig_atomic_t
```

Целочисленный тип к которому можно получить доступ как к атомарной сущности даже при наличии асинхронных прерываний по сигналам.

19.56.3.2. sig_handle

```
typedef void(* sig_handle) (int)
```

Тип, описывающий функцию-обработчик сигнала.

19.56.3.3. siginfo_t

```
typedef struct siginfo siginfo_t
```

Структура данных сигнала.

19.56.3.4. sigset_t

```
typedef uint32_t sigset_t
```

Целочисленный тип, описывающий объект, используемый для хранения набора сигналов.

19.56.4. Функции

19.56.4.1. kill()

```
int kill (  
        TASK_ID ti,  
        int sig )
```

Функция посылает сигнал указанной задаче.

Аргументы

ti Идентификатор задачи, которой требуется послать сигнал.

sig Сигнал.

Возвращает

OK при успешной отправке, либо код ошибки.

19.56.4.2. raise()

```
int raise (  
        int sig )
```

Функция позволяет послать из программы сигнал.

Аргументы

sig Сигнал.

Возвращает

OK при успешной отправке, либо код ошибки.

19.56.4.3. sigaction()

```
int sigaction (
    int signum,
    const struct sigaction * act,
    struct sigaction * oldact )
```

Функция задает обработчик сигнала.

Аргументы

signum Номер сигнала.

act Новый обработчик, или *NULL*, если установка обработчика не требуется.

oldact Буфер под старый обработчик, или *NULL*, если сохранение старого обработчика не требуется.

Возвращает

OK при успешной установке, либо код ошибки.

19.56.4.4. sigaddset()

```
int sigaddset (
    sigset_t * set,
    int signum )
```

Функция добавляет сигнал **signum** к **set**.

Аргументы

set Буфер набора сигналов.

signum Добавляемый сигнал.

Возвращает

OK При успешном завершении.

ERROR При ошибке.

19.56.4.5. sigdelset()

```
int sigdelset (  
    sigset_t * set,  
    int signum )
```

Функция удаляет сигнал **signum** из набора **set**.

Аргументы

<i>set</i>	Буфер набора сигналов.
<i>signum</i>	Удаляемый сигнал.

Возвращает

OK При успешном завершении.
ERROR При ошибке.

19.56.4.6. sigemptyset()

```
int sigemptyset (  
    sigset_t * set )
```

Функция инициализирует набор сигналов, указанный в **set**, и "очищает" его от всех сигналов.

Аргументы

<i>set</i>	Буфер набора сигналов.
------------	------------------------

Возвращает

OK При успешном завершении.
ERROR При ошибке.

19.56.4.7. sigfillset()

```
int sigfillset (  
    sigset_t * set )
```

Функция полностью инициализирует набор **set**, в котором содержатся все сигналы.

Аргументы

<i>set</i>	Буфер набора сигналов.
------------	------------------------

Возвращает

OK При успешном завершении.
ERROR При ошибке.

19.56.4.8. sigismember()

```
int sigismember (  
    sigset_t * set,  
    int signum )
```

Функция проверяет, является ли **signum** членом набора **set**.

Аргументы

set Указатель на набор сигналов.

signum Проверяемый сигнал.

Возвращает

1 — если **signum** является членом набора **set**.
0 — если **signum** не является членом набора.
-1 — при ошибке.

19.56.4.9. signal()

```
sig_handle signal (  
    int sig,  
    sig_handle func )
```

Функция принимает в качестве аргумента сигнал и указатель на **void** функцию, которая принимает сигнал.

Аргументы

sig Сигнал, для которого устанавливается обработчик.

func *SIG_IGN*, для игнорирования сигнала, *SIG_DFL*, для обработки сигнала по-умолчанию или указатель на обработчик сигнала.

Возвращает

Значение **func** для прошлого успешного вызова функции *signal()* или *SIG_ERR*, в случае ошибки.

19.56.4.10. wait()


```
TASK_ID wait (  
    int * status_p )
```

Функция приостанавливает выполнение вызвавшей задачи до тех пор, пока не прекратит выполнение один из её потомков.

Аргументы

status_p Буфер под статус задачи-потомка.

Возвращает

Указатель на структуру задачи-потомка.

19.57. Файл `sleep.h`

Различные обертки над функционалом приостановки задачи.

Макросы

- `#define MKSINSEC (1000L*1000L)`
- `#define MSINSEC (1000L)`

Функции

- `STATUS nodesleep` (int nominator, int denominator)
- int `nodetick` (int nominator, int denominator)
- `STATUS sleep60` (int secDiv60)
- `STATUS sleepmks` (int mks)
- `STATUS sleepms` (int ms)
- int `tick60` (int secDiv60)
- int `tickmks` (int mks)
- int `tickms` (int ms)

19.57.1. Макросы

19.57.1.1. MKSINSEC

```
#define MKSINSEC (1000L*1000L)
```

Кол-во микросекунд в секунде.

19.57.1.2. MSINSEC

```
#define MSINSEC (1000L)
```

Кол-во миллисекунд в секунде.

19.57.2. Функции

19.57.2.1. `nodesleep()`

```
STATUS nodesleep (  
    int nominator,  
    int denominator )
```

Задержать задачу на ближайшее к `nominator/denominator`[секунд] время.

Задержка не блокирующая - в ее время будут исполняться другие задачи.

Аргументы	
<code>nominator</code>	Числитель в формуле.
<code>denominator</code>	Знаменатель в формуле.

Возвращает

Всегда возвращает *OK*.

19.57.2.2. nodetick()

```
int nodetick (  
    int nominator,  
    int denominator )
```

Подсчитать, сколько рабочих тиков ОС занимает ближайшее к *nominator/denominator*[секунд] время.

Аргументы

nominator Числитель в формуле.

denominator Знаменатель в формуле.

Возвращает

Соответствующее кол-во рабочих тиков ОС.

19.57.2.3. sleep60()

```
STATUS sleep60 (  
    int secDiv60 )
```

Задержать задачу на заданное кол-во миллисекунд.

Задержка не блокирующая - в ее время будут исполняться другие задачи.

Аргументы

secDiv60 Кол-во 1/60х долей секунды.

Возвращает

Всегда возвращает *OK*.

19.57.2.4. sleepmks()

```
STATUS sleepmks (  
    int mks )
```

Задержать задачу на заданное кол-во микросекунд.

Задержка не блокирующая - в ее время будут исполняться другие задачи.

Аргументы

mks Кол-во микросекунд (не более 17 секунд).

Возвращает

Всегда возвращает *OK*.

19.57.2.5. sleepms()

STATUS sleepms (
int *ms*)

Задержать задачу на заданное кол-во миллисекунд.

Задержка не блокирующая - в ее время будут исполняться другие задачи.

Аргументы

ms Кол-во миллисекунд (не более 17 секунд).

Возвращает

Всегда возвращает *OK*.

19.57.2.6. tick60()

int tick60 (
int *secDiv60*)

Подсчитать, сколько рабочих тиков ОС занимает заданное кол-во 1/60 долей секунды.

Аргументы

secDiv60 Кол-во 1/60х долей секунды.

Возвращает

Соответствующее кол-во рабочих тиков ОС.

19.57.2.7. tickmks()

int tickmks (
int *mks*)

Подсчитать, сколько рабочих тиков ОС занимает заданное кол-во микросекунд.

Аргументы

mks Кол-во микросекунд (не более 2х секунд).

Возвращает

Соответствующее кол-во рабочих тиков ОС.

19.57.2.8. tickms()

int tickms (
 int *ms*)

Подсчитать, сколько рабочих тиков ОС занимает заданное кол-во миллисекунд.

Аргументы

ms Кол-во миллисекунд (не более 35 минут).

Возвращает

Соответствующее кол-во рабочих тиков ОС.

19.58. Файл `socket.h`

Протокол TCP.

Структуры данных

- struct `in_addr`
- struct `sockaddr`
- struct `sockaddr_in`
Структура адреса сокета для связи через сеть.

Макросы

- #define `INADDR_ANY` {-1}
- #define `inet_addr(x) strtoul(x)`
- #define `INVALID_SOCKET` (-1)

Определения типов

- typedef unsigned short `in_port_t`
- typedef unsigned short `sa_family_t`

Функции

- int `accept` (int sockfd, struct `sockaddr` *address, size_t *add_len)
Приём запроса на установку TCP-соединения.
- int `bind` (int sockfd, const struct `sockaddr` *address, size_t add_len)
Связывание сокета с адресом.
- int `connect` (int sockfd, const struct `sockaddr` *address, size_t add_len)
Подключение клиента.
- int `getsockopt` (int sockfd)
- int `listen` (int sockfd, int queue_size)
Включение приема TCP-соединения.
- int `setsockopt` (int sockfd, int timeout)
Задать таймаут сокета.
- int `shutdown` (int sockfd, int how)
Прекращение обмена сокетом.
- int `socket` (int domain, int type, int protocol)
Создание сокета.

Коммуникационный домен

Допустим домен **Internet**.

- #define `AF_INET` 2
- #define `PF_INET` `AF_INET`

Типы сокетов

- #define `SOCK_DGRAM` 0x2000
- #define `SOCK_RAW` 0
- #define `SOCK_STREAM` 0x1000
- #define `SOCK_TYPE_MASK` 0x3000

Значения параметра `flags` для `send`, `receive` и т.п.

- `#define MSG_DONTROUTE 0x4`
- `#define MSG_EOF 0x100`
- `#define MSG_EOR 0x8`
- `#define MSG_OOB 0x1`
- `#define MSG_PEEK 0x2`

Значения параметра `how` функции `shutdown()`.

- `#define SD_BOTH 3`
- `#define SD_RECEIVE 1`
- `#define SD_SEND 2`

19.58.1. Подробное описание

В файле описаны методы работы с сетевой подсистемой с использованием протокола **TCP/IP**.

Подключение:

```
#include <socket.h>
```

config.h:

```
#define INCLUDE_NETINET
```

Makefile:

```
LIBRARIES += -l_enet -l_socket
```

См. также

Общее описание в главе *Сетевая подсистема*.

19.58.2. Макросы

19.58.2.1. AF_INET

```
#define AF_INET 2
```

19.58.2.2. INADDR_ANY

```
#define INADDR_ANY {-1}
```

19.58.2.3. inet_addr

```
#define inet_addr(  
    x ) strtoip(x)
```

19.58.2.4. INVALID_SOCKET

```
#define INVALID_SOCKET (-1)
```

19.58.2.5. MSG_DONTROUTE

```
#define MSG_DONTROUTE 0x4
```

Bypass routing, use direct interface.

19.58.2.6. MSG_EOF

```
#define MSG_EOF 0x100
```

Data completes transaction.

19.58.2.7. MSG_EOR

```
#define MSG_EOR 0x8
```

Data completes record.

19.58.2.8. MSG_OOB

```
#define MSG_OOB 0x1
```

Process out-of-band data.

19.58.2.9. MSG_PEEK

```
#define MSG_PEEK 0x2
```

Peek at incoming message.

19.58.2.10. PF_INET

```
#define PF_INET AF_INET
```

19.58.2.11. SD_BOTH

```
#define SD_BOTH 3
```

Прекратить обмен в обе стороны.

19.58.2.12. SD_RECEIVE

```
#define SD_RECEIVE 1
```

Прекратить прием данных сокетом.

19.58.2.13. SD_SEND

```
#define SD_SEND 2
```

Прекратить выдачу данных сокетом.

19.58.2.14. SOCK_DGRAM

```
#define SOCK_DGRAM 0x2000
```

19.58.2.15. SOCK_RAW

```
#define SOCK_RAW 0
```

19.58.2.16. SOCK_STREAM

```
#define SOCK_STREAM 0x1000
```

19.58.2.17. SOCK_TYPE_MASK

```
#define SOCK_TYPE_MASK 0x3000
```

19.58.3. Типы**19.58.3.1. in_port_t**

```
typedef unsigned short in_port_t
```

Порт в сети (16 бит).

19.58.3.2. sa_family_t

```
typedef unsigned short sa_family_t
```

19.58.4. Функции**19.58.4.1. accept()**

```
int accept (  
    int sockfd,
```

```
struct sockaddr * address,  
size_t * add_len )
```

Процедура подтверждает запрос на установление соединения от удаленного хоста.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
<i>address</i>	Указатель на структуру <i>sockaddr</i> для получения информации об адресе клиента.
<i>add_len</i>	Указатель на переменную, содержащую размер структуры адреса.

Возвращает

Функция возвращает дескриптор сокета, связанного с этим конкретным соединением, кроме того, в переменную **add_len** записывается фактический размер адреса.

19.58.4.2. bind()

```
int bind (  
    int sockfd,  
    const struct sockaddr * address,  
    size_t add_len )
```

Связывает сокет с конкретным адресом. Когда сокет создается при помощи *socket()*, он ассоциируется с некоторым семейством адресов, но не с конкретным адресом. До того как сокет сможет принять входящие соединения, он должен быть связан с адресом.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
<i>address</i>	Указатель на структуру <i>sockaddr</i> , представляющую адрес, к которому привязываем.
<i>add_len</i>	Длина структуры адреса.

Возвращает

OK при успехе.
ERROR при неудаче.

19.58.4.3. connect()

```
int connect (  
    int sockfd,  
    const struct sockaddr * address,  
    size_t add_len )
```

Функция устанавливает соединение с сервером.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
<i>address</i>	Указатель на структуру адреса.
<i>add_len</i>	Размер структуры адреса.

Возвращает

OK

Код ошибки при неудаче.

19.58.4.4. getsockopttimeout()

```
int getsockopttimeout (  
    int sockfd )
```

Функция возвращает таймаут, установленный для сокета.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
---------------	--------------------

Возвращает

Функция возвращает таймаут, установленный для данного сокета.

19.58.4.5. listen()

```
int listen (  
    int sockfd,  
    int queue_size )
```

Подготавливает привязываемый сокет к принятию входящих соединений.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
<i>queue_size</i>	Число установленных соединений, которые могут быть обработаны одновременно.

Возвращает

OK при успехе.
ERROR при неудаче.

19.58.4.6. setsockopttimeout()

```
int setsockopttimeout (  
    int sockfd,  
    int timeout )
```

Функция устанавливает таймаут для сокета.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
<i>timeout</i>	Таймаут в тиках таймера.

Возвращает

OK при успешном завершении.
ERROR при неудаче.

19.58.4.7. shutdown()

```
int shutdown (  
    int sockfd,  
    int how )
```

Функция прекращает обмен для указанного сокета.

Аргументы

<i>sockfd</i>	Дескриптор сокета.
<i>how</i>	Что именно прекратить - значение из группы <i>значений</i> : <ul style="list-style-type: none">• <i>SD_RECEIVE</i>• <i>SD_SEND</i>• <i>SD_BOTH</i>

Возвращает

OK при успехе.
ERROR при неудаче.

19.58.4.8. socket()

```
int socket (  
    int domain,  
    int type,  
    int protocol )
```

Функция создает сокет.

Аргументы

domain Нужно указывать *AF_INET*.

type Нужно указывать *SOCK_STREAM*.

protocol Нужно указывать **0** — значение по умолчанию для данного соединения.

Возвращает

Возвращает дескриптор сокета, как стандартного устройства ввода / вывода.

19.59. Файл `softgraph.dox`

19.60. Файл softgraph.h

Аппаратная реализация работы с поверхностями.

Структуры данных

- struct *sDisplayInfo*
Описание параметров дисплея.
- struct *tagSURFACE*
Описание параметров поверхности.

Определения типов

- typedef *SURFACE * HDCp*
- typedef struct *tagSURFACE SURFACE*
Описание параметров поверхности.

Инициализация модуля

- *HDCp softGraphConstr ()*
Получить указатель на конструируемую поверхность.
- void *softGraphConstrUpdate (void *constr)*
Изменить изображение конструируемой поверхности.
- void *softGraphInit (const sDisplayInfo *info, void *constr)*
Инициализация графической поверхности.

Работа с поверхностями

- void *clearSurface (HDCp psf)*
Очистить поверхность.
- *HDCp createScreenSurface (void)*
Создание первичной (конструируемой) поверхности.
- *HDCp createSHdr (int XRes, int YRes)*
Создать поверхность со стандартными параметрами.
- *HDCp createSurface (int X, int Y)*
Создать пустую поверхность.
- *HDCp emptySurface ()*
Создать поверхность-заглушку.
- void *fillSurface (HDCp psf, int color)*
Залить поверхность цветом.
- void *freeSurface (HDCp sf)*
Удалить поверхность.
- *HDCp loadFromBitmap (char *fileName)*
Загрузить поверхность из BMP.
- *HDCp loadFromJPG (char *fileName)*
Загрузить поверхность из JPG.
- *HDCp loadFromPNG (char *name)*
Загрузить поверхность из PNG.
- *HDCp sfClone (HDCp SF)*
Создание клона с копией содержимого.
- *HDCp sfCloneSample (HDCp SF, int X, int Y, HDCp SAM)*
Создание из зоны по образцу.
- *HDCp sfCloneZone (HDCp SF, int X, int Y, int W, int H)*
Создание из зоны источника.
- *STATUS sfCopy (HDCp DST, HDCp SRC)*
Копирование содержимого одинакового размера.

Преобразование цвета

Функции преобразования цвета в используемый формат, который определяется при инициализации библиотеки *softGraphInit()*.

- void *fromRGB* (unsigned int c, unsigned int *R, unsigned int *G, unsigned int *B)
Разобрать цвет на составляющие.
- unsigned int *RGB* (unsigned int R, unsigned int G, unsigned int B)
Получить значение цвета в используемом формате.

Рисование на поверхности

- void *sfBar* (*HDCp* psf, int X, int Y, int W, int H, unsigned int color)
Рисование прямоугольника сплошным цветом.
- BOOL *sfBitBlt* (*HDCp* hdcDest, int nXDest, int nYDest, int nWidth, int nHeight, *HDCp* hdcSrc, int nXSrc, int nYSrc)
Наложение прямоугольной зоны.
- bool *sfBitBltAlphaColor* (*HDCp* hdcDest, int nXDest, int nYDest, int nWidth, int nHeight, *HDCp* hdcSrc, int nXSrc, int nYSrc, *uint32_t* color)
Наложение цвета с использованием альфа-канала из прямоугольной зоны.
- void *sfCircle* (*HDCp* psf, int xc, int yc, int r, unsigned int color)
Рисование окружности.
- void *sfCurvedRectangle* (*HDCp* psf, int X, int Y, unsigned W, unsigned H, unsigned curveRadius, *uint32_t* fillColor, unsigned borderWidth, *uint32_t* borderColor)
Рисование прямоугольника с закругленными краями.
- void *sfCurvedRectangleTest* (*HDCp* psf, int x, int y, unsigned width, unsigned height, unsigned curveRadius, *uint32_t* fillColor, unsigned borderWidth, *uint32_t* borderColor)
- BOOL *sfDraw* (*HDCp* hdcDest, int nXDest, int nYDest, *HDCp* hdcSrc)
Отрисовка источника целиком в приёмнике.
- BOOL *sfDrawPolygon* (*HDCp* hdc, int points[][2], int n, unsigned int color)
Рисование произвольной закрашенной области
- BOOL *sfDrawTransparent* (*HDCp* hdcDest, int nXDest, int nYDest, *HDCp* hdcSrc, unsigned char nTsp)
Отрисовка с заданной степенью прозрачности.
- void *sfFilledCircle* (*HDCp* psf, int xc, int yc, unsigned r, *uint32_t* fillColor, unsigned borderWidth, *uint32_t* borderColor)
Рисование заполненной окружности с границей.
- unsigned int *sfGetPixel* (*HDCp* psf, unsigned int X, unsigned int Y)
Считывание цвета точки на поверхности.
- void *sfLine* (*HDCp* psf, int X1, int Y1, int X2, int Y2, unsigned int color)
Рисование линии.
- void *sfLineTo* (*HDCp* psf, int X, int Y, unsigned int color)
Нарисовать линию от текущей позиции карандаша и переместить его в конец линии.
- void *sfMoveTo* (*HDCp* psf, int X, int Y)
Перемещение карандаша.
- void *sfPutPixel* (*HDCp* psf, unsigned int X, unsigned int Y, unsigned int color)
Рисование точки заданного цвета на поверхности.
- void *sfRectangle* (*HDCp* psf, int X, int Y, int W, int H, unsigned int fillColor, unsigned int borderColor)
Рисование прямоугольника с рамкой цвета карандаша и заливкой цвета кисти.
- void *sfSmoothCircle* (*HDCp* psf, int xc, int yc, unsigned r, *uint32_t* fillColor, unsigned borderWidth, *uint32_t* borderColor)
Рисование гладкой заполненной окружности с границей.
- BOOL *sfStretchBlt* (*HDCp* hdc, int hdcX, int hdcY, int hdcW, int hdcH, *HDCp* src, int srcX, int srcY, int srcW, int srcH)
Отрисовка с масштабированием.

19.60.1. Подробное описание

Аппаратная реализация работы с поверхностями для процессоров, не имеющих аппаратного модуля работы с 2D-графикой.

Подключение:

```
#include <multimedia/softgraph.h>
```

Makefile:

```
LIBRARIES += -l_softgraph -l_png -l_z
```

См. также

Общее описание работы с графической подсистемой в главе [Графическая подсистема](#).

19.60.2. Типы

19.60.2.1. HDCp

```
typedef SURFACE* HDCp
```

19.60.2.2. SURFACE

```
typedef struct tagSURFACE SURFACE
```

19.60.3. Функции

19.60.3.1. clearSurface()

```
void clearSurface (  
    HDCp psf)
```

Функция заполняет поверхность сплошным цветом.
Цвет соответствует значению **sfBrushColor** в структуре *SURFACE*.

Аргументы

psf Указатель на поверхность *SURFACE*.

19.60.3.2. createScreenSurface()

```
HDCp createScreenSurface (  
    void )
```

Возвращает

Указатель на поверхность *SURFACE*.

19.60.3.3. createSHdr()

```
HDCp createSHdr (  
    int XRes,  
    int YRes )
```

Аргументы

XRes Ширина поверхности.

YRes Высота поверхности.

Возвращает

указатель на новую поверхность *SURFACE*.

19.60.3.4. createSurface()

```
HDCp createSurface (  
    int X,  
    int Y )
```

Функция создаёт поверхность с указанными сторонами и выделяет память для изображения.

Аргументы

X Ширина поверхности.

Y Высота поверхности.

Возвращает

Указатель на созданную поверхность *SURFACE*.

19.60.3.5. emptySurface()

```
HDCp emptySurface ( )
```

Функция создаёт поверхность с размерами 100x100 и заполняет чёрным цветом.

Возвращает

Указатель на созданную поверхность *SURFACE*.

19.60.3.6. fillSurface()

```
void fillSurface (  
    HDCp psf,  
    int color )
```

Функция заполняет поверхность указанным цветом.

Аргументы

psf Указатель на поверхность *SURFACE*.

color Цвет заливки.

19.60.3.7. freeSurface()

```
void freeSurface (  
    HDCp sf)
```

Аргументы

sf Указатель на поверхность *SURFACE*.

19.60.3.8. fromRGB()

```
void fromRGB (  
    unsigned int c,  
    unsigned int * R,  
    unsigned int * G,  
    unsigned int * B )
```

Функция разбирает значение на **RGB** компоненты.

Аргументы

c Исходное значение.

R,G,B Полученные значения.

19.60.3.9. loadFromBitmap()

```
HDCp loadFromBitmap (
```

```
char * fileName )
```

Функция загружает поверхность данными из **BMP** - Файла.
Поддерживаемые форматы файла: **RGB 24-бита, RGBX и RGBA 32-бита**.

Важно! Если формат файла не **RGBA**, то в поверхность записывается альфа-канал со стандартным значением **0xff**.

Форматы 16-бит и ниже не поддерживаются.

Вызывает *emptySurface()*, если не удалось декодировать файл.

Аргументы

<i>fileName</i>	Имя файла на диске.
-----------------	---------------------

Возвращает

Указатель на созданную поверхность *SURFACE*.

19.60.3.10. loadFromJPG()

```
HDCp loadFromJPG (  
    char * fileName )
```

Функция загружает поверхность данными из **JPG** - Файла.

Важно! В поверхность также записывается альфа-канал со стандартным значением **0xff**.

Файлы формата **JPEG** распознаются системой некорректно, чтобы открыть изображение, введите имя в виде: **f_name.jpe**

Вызывает *emptySurface()*, если не удалось декодировать файл.

Аргументы

<i>fileName</i>	Имя файла на диске.
-----------------	---------------------

Возвращает

Указатель на созданную поверхность *SURFACE*.

19.60.3.11. loadFromPNG()

```
HDCp loadFromPNG (  
    char * name )
```

Функция загружает поверхность данными из **PNG** - Файла.
Поддерживаемый формат: **8-bit RGBA**.

Вызывает *emptySurface()*, если не удалось декодировать файл.

Аргументы

name Имя файла на диске.

Возвращает

Указатель на созданную поверхность *SURFACE*.

19.60.3.12. RGB()

```
unsigned int RGB (  
    unsigned int R,  
    unsigned int G,  
    unsigned int B )
```

Функция собирает значение цвета из значений яркости для каждого канала.

Аргументы

R,G,B Значения цвета каждого из каналов в пределах от 0 до 255.

Возвращает

Результирующее значение.

19.60.3.13. sfBar()

```
void sfBar (  
    HDCp psf,  
    int X,  
    int Y,  
    int W,  
    int H,  
    unsigned int color )
```

Зарисовать прямоугольную область сплошным цветом. Можно указывать отрицательные координаты, тогда будет отрисовываться только видимая часть прямоугольника.

Аргументы

psf Указатель на поверхность.

X,Y Координаты левого верхнего угла на поверхности.

W Ширина прямоугольника.

H Высота прямоугольника.

color Цвет заливки с альфа-каналом.

19.60.3.14. sfBitBlit()

```

BOOL sfBitBlit (
    HDCp hdcDest,
    int nXDest,
    int nYDest,
    int nWidth,
    int nHeight,
    HDCp hdcSrc,
    int nXSrc,
    int nYSrc )

```

Функция обрабатывает источники с альфа-каналом.

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть источника.

Аргументы	
<i>hdcDest</i>	Указатель на поверхность-приёмник.
<i>nXDest, nYDest</i>	Координаты левого верхнего угла на поверхности-приёмнике.
<i>nWidth</i>	Ширина поверхности-приёмника.
<i>nHeight</i>	Высота поверхности-приёмника.
<i>hdcSrc</i>	Указатель на поверхность-источник.
<i>nXSrc, nYSrc</i>	Координаты левого верхнего угла на поверхности-источника.

Возвращает

false, если не удалось наложить источник, иначе *true*.

19.60.3.15. sfBitBlitAlphaColor()

```

bool sfBitBlitAlphaColor (
    HDCp hdcDest,
    int nXDest,
    int nYDest,
    int nWidth,
    int nHeight,
    HDCp hdcSrc,
    int nXSrc,
    int nYSrc,
    uint32_t color )

```

Предназначение функции - быстрый и экономичный вывод глифов текста. Функция использует значение альфа-канала из зоны *hdcSrc*, но игнорирует цветовую составляющую этой зоны. В качестве цвета вывода используется значение *color*.

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть источника.

Аргументы	
<i>hdcDest</i>	Указатель на поверхность-приёмник.
<i>nXDest, nYDest</i>	Координаты левого верхнего угла на поверхности-приёмнике.
<i>nWidth</i>	Ширина поверхности-приёмника.
<i>nHeight</i>	Высота поверхности-приёмника.
<i>hdcSrc</i>	Указатель на поверхность-источник, из которой будет браться значение альфа-канала.
<i>nXSrc, nYSrc</i>	Координаты левого верхнего угла на поверхности-источника.
<i>color</i>	Цвет, который будет использоваться при наложении.

Возвращает

false, если не удалось наложить источник, иначе *true*.

19.60.3.16. sfCircle()

```
void sfCircle (
    HDCp psf,
    int xc,
    int yc,
    int r,
    unsigned int color )
```

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть окружности.

Фигура не отрисовывается, если $r \leq 0$.

Аргументы	
<i>psf</i>	Указатель на поверхность.
<i>xc, yc</i>	Координаты центра окружности.
<i>r</i>	Радиус.
<i>color</i>	Цвет линии с альфа-каналом.

19.60.3.17. sfClone()

```
HDCp sfClone (
    HDCp SF)
```

Аргументы

SF Указатель на клонируемую поверхность *SURFACE*.

Возвращает

Указатель на поверхность *SURFACE*.

19.60.3.18. sfCloneSample()

```
HDCp sfCloneSample (  
    HDCp SF,  
    int X,  
    int Y,  
    HDCp SAM )
```

Функция вызывает **sfCloneZone**, передавая ширину и высоту поверхности **SAM**.

Аргументы

SF Указатель на поверхность-источник *SURFACE*.

X,Y Координаты левого верхнего угла на поверхности-источнике.

SAM Указатель на поверхность-пример. Новая поверхность будет иметь стороны примера.

Возвращает

Указатель на поверхность *SURFACE*.

19.60.3.19. sfCloneZone()

```
HDCp sfCloneZone (  
    HDCp SF,  
    int X,  
    int Y,  
    int W,  
    int H )
```

Отрицательные *X* и *Y* приравниваются к 0.

Если *W* или *H* < 0, функция вызывает *emptySurface()*.

Если *X* + *W* больше ширины поверхности, функция скопирует источник от *X* до правого края.
Если *Y* + *H* больше высоты поверхности, функция скопирует источник от *Y* до нижнего края.

Аргументы	
<i>SF</i>	Указатель на копируемую поверхность <i>SURFACE</i> .
<i>X,Y</i>	Координаты левого верхнего угла на поверхности-источнике.
<i>W</i>	Ширина поверхности.
<i>H</i>	Высота поверхности.

Возвращает

Указатель на поверхность *SURFACE*.

19.60.3.20. sfCopy()

```
STATUS sfCopy (
    HDCp DST,
    HDCp SRC )
```

Аргументы	
<i>DST</i>	Указатель на поверхность-приёмник.
<i>SRC</i>	Указатель на поверхность-источник.

Возвращает

ERROR, если приёмник и источник разных размеров, иначе возвращает *OK*.

19.60.3.21. sfCurvedRectangle()

```
void sfCurvedRectangle (
    HDCp psf,
    int X,
    int Y,
    unsigned W,
    unsigned H,
    unsigned curveRadius,
    uint32_t fillColor,
    unsigned borderWidth,
    uint32_t borderColor )
```

Можно указывать отрицательные координаты, тогда будет отрисовываться только видимая часть прямоугольника. Границы будут отрисовываться "внутри" указанных координат.

Аргументы	
<i>psf</i>	Указатель на поверхность.

Продолжение на следующей странице

Аргументы (Продолжение.)	
<i>X,Y</i>	Координаты левого верхнего угла на поверхности.
<i>W,H</i>	Ширина и высота прямоугольника.
<i>curveRadius</i>	Радиус кривизны краев прямоугольника (радиус будет усечен, если он будет слишком большим).
<i>fillColor</i>	Цвет заливки с альфа-каналом.
<i>borderWidth</i>	Толщина границы.
<i>borderColor</i>	Цвет границы с альфа-каналом.

19.60.3.22. sfCurvedRectangleTest()

```
void sfCurvedRectangleTest (
    HDCp psf,
    int x,
    int y,
    unsigned width,
    unsigned height,
    unsigned curveRadius,
    uint32_t fillColor,
    unsigned borderWidth,
    uint32_t borderColor )
```

19.60.3.23. sfDraw()

```
BOOL sfDraw (
    HDCp hdcDest,
    int nXDest,
    int nYDest,
    HDCp hdcSrc )
```

Функция обрабатывает источники с альфа-каналом.

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть источника.

Аргументы	
<i>hdcDest</i>	Указатель на поверхность-приёмник.
<i>nXDest,nYDest</i>	Координаты левого верхнего угла на поверхности-приёмнике.
<i>hdcSrc</i>	Указатель на поверхность-источник.

Возвращает

false, если не удалось отрисовать источник, иначе *true*.

19.60.3.24. sfDrawPolygon()

```

BOOL sfDrawPolygon (
    HDCp hdc,
    int points[][2],
    int n,
    unsigned int color )

```

Функция может отрисовывать выпуклые, вогнутые и пересекающие сами себя фигуры. Можно указывать точки за пределами экрана. Функция использует библиотеку **vector**.

Важно! Если некоторые из рёбер фигуры находятся внутри неё и создают отдельную область, то эта область не будет закрашена. Если **n** указан меньше реального размера массива, область отрисуеться по оставшимся точкам. Если **n** указан больше реального размера массива, система упадёт.

Аргументы

<i>hdc</i>	Указатель на поверхность.
<i>points</i>	Массив точек с координатами X и Y. Каждая точка - это массив из двух элементов.
<i>n</i>	Размер массива.
<i>color</i>	Цвет заливки с альфа-каналом.

Возвращает

Возвращает 1 в случае ошибки.

19.60.3.25. sfDrawTransparent()

```

BOOL sfDrawTransparent (
    HDCp hdcDest,
    int nXDest,
    int nYDest,
    HDCp hdcSrc,
    unsigned char nTsp )

```

Указанная прозрачность смешивается с альфа-каналом источника.

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть источника.

Аргументы

<i>hdcDest</i>	Указатель на поверхность-приёмник.
<i>nXDest, nYDest</i>	Координаты левого верхнего угла на поверхности-приёмнике.
<i>hdcSrc</i>	Указатель на поверхность-источник.
<i>nTsp</i>	Прозрачность, от 0 до 0xff, где 0 - полностью прозрачный.

Возвращает

false, если не удалось наложить источник, иначе *true*.

19.60.3.26. sfFilledCircle()

```
void sfFilledCircle (  
    HDCp psf,  
    int xc,  
    int yc,  
    unsigned r,  
    uint32_t fillColor,  
    unsigned borderWidth,  
    uint32_t borderColor )
```

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть окружности.

Фигура не отрисовывается, если $r \leq 0$. Итоговый радиус фигуры будет равен r , граница будет отрисована "внутри" этого радиуса.

Аргументы	
<i>psf</i>	Указатель на поверхность.
<i>xc,yc</i>	Координаты центра окружности.
<i>r</i>	Радиус окружности.
<i>fillColor</i>	Цвет окружности.
<i>borderWidth</i>	Толщина границы окружности. Может быть нулевой.
<i>borderColor</i>	Цвет границы окружности.

19.60.3.27. sfGetPixel()

```
unsigned int sfGetPixel (  
    HDCp psf,  
    unsigned int X,  
    unsigned int Y)
```

Функция получает цвет точки на указанной поверхности.

Аргументы	
<i>psf</i>	Поверхность для рисования.
<i>X,Y</i>	Координаты точки.

Возвращает

Цвет точки.

19.60.3.28. sfLine()

```
void sfLine (  
    HDCp psf,  
    int X1,  
    int Y1,  
    int X2,  
    int Y2,  
    unsigned int color )
```

Можно указывать координаты за пределами поверхности.

Аргументы	
<i>psf</i>	Указатель на поверхность.
X1,Y1	Координаты первой точки.
X2,Y1	Координаты последней точки.
<i>color</i>	Цвет линии с альфа-каналом

19.60.3.29. sfLineTo()

```
void sfLineTo (  
    HDCp psf,  
    int X,  
    int Y,  
    unsigned int color )
```

Можно указывать координаты за пределами поверхности.

Аргументы	
<i>psf</i>	Указатель на поверхность.
X,Y	Координаты последней точки линии.
<i>color</i>	Цвет линии с альфа-каналом.

19.60.3.30. sfMoveTo()

```
void sfMoveTo (  
    HDCp psf,  
    int X,  
    int Y)
```

Можно указывать координаты за пределами поверхности.

Аргументы	
<i>psf</i>	Указатель на поверхность.
<i>X,Y</i>	Новые координаты карандаша.

19.60.3.31. sfPutPixel()

```
void sfPutPixel (  
    HDCp psf,  
    unsigned int X,  
    unsigned int Y,  
    unsigned int color )
```

Функция закрашивает точку заданным цветом на указанной поверхности.

Аргументы	
<i>psf</i>	Поверхность для рисования.
<i>X,Y</i>	Координаты точки.
<i>color</i>	Цвет точки с альфа-каналом.

19.60.3.32. sfRectangle()

```
void sfRectangle (  
    HDCp psf,  
    int X,  
    int Y,  
    int W,  
    int H,  
    unsigned int fillColor,  
    unsigned int borderColor )
```

Можно указывать отрицательные координаты, тогда будет отрисовываться только видимая часть прямоугольника.

Фигура будет полой, если значение **sfBClear** равно *true*.

Аргументы	
<i>psf</i>	Указатель на поверхность.
<i>X,Y</i>	Координаты левого верхнего угла на поверхности.
<i>W</i>	Ширина прямоугольника.
<i>H</i>	Высота прямоугольника.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>fillColor</i>	Цвет заливки с альфа-каналом.
<i>borderColor</i>	Цвет границы с альфа-каналом.

19.60.3.33. sfSmoothCircle()

```
void sfSmoothCircle (  
    HDCp psf,  
    int xc,  
    int yc,  
    unsigned r,  
    uint32_t fillColor,  
    unsigned borderWidth,  
    uint32_t borderColor )
```

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть окружности.

Фигура не отрисовывается, если $r \leq 0$. Итоговый радиус фигуры будет равен не r , гладкая граница может незначительно (не более чем на 1 пиксель) выходить за эти границы.

Функция работает существенно дольше, чем *sfFilledCircle*.

Аргументы

<i>psf</i>	Указатель на поверхность.
<i>xc,yc</i>	Координаты центра окружности.
<i>r</i>	Радиус окружности.
<i>fillColor</i>	Цвет окружности.
<i>borderWidth</i>	Толщина границы окружности. Может быть нулевой.
<i>borderColor</i>	Цвет границы окружности.

19.60.3.34. sfStretchBlit()

```
BOOL sfStretchBlit (  
    HDCp hdc,  
    int hdcX,  
    int hdcY,  
    int hdcW,  
    int hdcH,  
    HDCp src,  
    int srcX,  
    int srcY,  
    int srcW,  
    int srcH )
```

Функция обрабатывает источники с альфа-каналом.

Можно указывать координаты за пределами поверхности, тогда будет отрисовываться только видимая часть источника.

Важно! Если обе поверхности одинакового размера, вызывается функция `sfBitBlt`. Эта функция не учитывает параметры `srcW,srcH`

Аргументы	
<code>hdc</code>	Указатель на поверхность-приёмник.
<code>hdcX,hdcY</code>	Координаты левого верхнего угла на поверхности-приёмнике.
<code>hdcW,hdcH</code>	Ширина и высота поверхности-приёмника.
<code>src</code>	Указатель на поверхность-источник.
<code>srcX,srcY</code>	Координаты левого верхнего угла на поверхности-источника.
<code>srcW,srcH</code>	Ширина и высота поверхности-источника.

Возвращает

`false`, если не удалось отрисовать источник, иначе `true`

19.60.3.35. `softGraphConstr()`

`HDCp softGraphConstr ()`

Возвращает

Указатель на поверхность `SURFACE`.

19.60.3.36. `softGraphConstrUpdate()`

`void softGraphConstrUpdate (`
`void * constr)`

Аргументы	
<code>constr</code>	Указатель на новое изображение.

19.60.3.37. `softGraphInit()`

`void softGraphInit (`
`const sDisplayInfo * info,`
`void * constr)`

Аргументы

info Параметры дисплея .

constr Указатель на конструируемую поверхность.

19.61. Файл sound.h

Работа со звуковой подсистемой.

Настройка и инициализация

- int *setMasterVol* (unsigned char vol)
Установить громкость звука.
- int *soundInit* ()
Инициализация звуковой системы.

Воспроизведение сырых данных

- int *playSndBuffer* (char *buffer, int size)
Воспроизведение звука из буфера.
- int *setSndFmt* (int Channels, int SampleSize, int SampleRate)
Задать формат воспроизведения буфера.

Поддержка WAV-файлов

- int *playWaveBuffer* (char *Buffer, int limit, int BufLen)
Воспроизведение буфера в формате RIFF.
- int *playWaveFile* (char *name)
Воспроизведение файла в формате WAV.

Поддержка MP3

- int *playMP3File* (char *name)
Воспроизведение файла MP3.
- void *stopMP3* (void)
Остановить воспроизведение файла MP3.

Запись звука

- char * *getRecSndBuffer* (int *len)
Указатель на записанный фрагмент звука.
- int *recStart* (void)
Начало звукозаписи.
- int *recStop* (void)
Остановить звукозапись.

19.61.1. Подробное описание

В файле собраны функции поддержки записи/воспроизведения звука.

Подключение:

```
#include <sound.h>
```

См. также

Общее описание работы звуковой подсистемы в главе [Звуковая подсистема](#).

19.61.2. Функции

19.61.2.1. getRecSndBuffer()

```
char* getRecSndBuffer (  
    int * len )
```

Функция возвращает указатель на записанный фрагмент в формате **PCM**.

Аргументы

len Указатель на длину фрагмента в байтах.

Возвращает

Указатель на буфер, содержащий фрагмент, записанный между двумя вызовами функции.

19.61.2.2. playMP3File()

```
int playMP3File (  
    char * name )
```

Функция проигрывает звуковой файл в стандарте **MP3**. Воспроизведение производится в отдельном потоке. Для работы требуется подключение библиотеки **libMP3.a**.

Аргументы

name Имя **MP3**-файла.

Возвращает

OK при удачном завершении.
ERROR при ошибке.

19.61.2.3. playSndBuffer()

```
int playSndBuffer (  
    char * buffer,  
    int size )
```

Функция проигрывает содержимое буфера, как звуковой файл в формате **PCM**.

Аргументы

buffer Указатель на звуковой буфер.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>size</i>	Размер буфера в байтах.
-------------	-------------------------

Возвращает

OK при удачном завершении.

ERROR при ошибке.

19.61.2.4. playWaveBuffer()

```
int playWaveBuffer (  
    char * Buffer,  
    int limit,  
    int BufLen )
```

Функция проигрывает буфер, содержащий звук в формате **RIFF (Wave)**. Формат воспроизведения установится в соответствии с данными, записанными в заголовке буфера.

Аргументы

<i>Buffer</i>	Указатель на буфер.
---------------	---------------------

<i>limit</i>	Размер проигрывания в байтах или 0 , чтобы не проверять.
--------------	---

<i>BufLen</i>	Размер буфера.
---------------	----------------

Возвращает

OK при удачном завершении.

ERROR при ошибке.

19.61.2.5. playWaveFile()

```
int playWaveFile (  
    char * name )
```

Функция проигрывает звуковой файл типа **WAV (Waveform Audio)**.

Аргументы

<i>name</i>	Имя WAV -файла.
-------------	------------------------

Возвращает

OK при удачном завершении.

ERROR при ошибке.

19.61.2.6. recStart()

```
int recStart (
    void )
```

Функция включает режим звукозаписи.

Возвращает

OK при удачном завершении.
ERROR при ошибке.

19.61.2.7. recStop()

```
int recStop (
    void )
```

Функция выключает режим звукозаписи.

Возвращает

OK при удачном завершении.
ERROR при ошибке.

19.61.2.8. setMasterVol()

```
int setMasterVol (
    unsigned char vol )
```

Функция устанавливает громкость проигрывания звука.

Аргументы

vol Громкость звука от **0** до **100**.

Возвращает

OK при удачном завершении, иначе *ERROR*.

19.61.2.9. setSndFmt()

```
int setSndFmt (
    int Channels,
    int SampleSize,
    int SampleRate )
```

Функция устанавливает заданный формат для воспроизведения буфера.

Аргументы	
<i>Channels</i>	Количество каналов для воспроизведения (1 или 2).
<i>SampleSize</i>	Размер выборки в битах.
<i>SampleRate</i>	Частота дискретизации звука в герцах.

Возвращает

OK при удачном завершении.

ERROR при ошибке.

19.61.2.10. soundInit()

```
int soundInit ( )
```

Инициализация звуковой системы выбранного процессора. Процессор указывается в файле проекта config.h.

Возвращает

OK при удачном запуске аппаратного кодека, иначе *ERROR*.

19.61.2.11. stopMP3()

```
void stopMP3 (
    void )
```

Функция останавливает проигрывание файла в формате **MP3**.

19.62. Файл spi.h

Интерфейс SPI.

Макросы выбора каналов SPI

- #define *SPI_0* 0
- #define *SPI_1* 1
- #define *SPI_2* 2
- #define *SPI_3* 3

Макросы выбора линий Chip Select

- #define *SPI_CS_0* 0
- #define *SPI_CS_1* 1

Макросы выбора набора выходных линий SPI

- #define *SPI_GPIO_ADDITIONAL* 1
- #define *SPI_GPIO_DEFAULT* 0

Настройка параметров передачи данных

Параметры передачи можно изменять как до инициализации модуля, так и после. Также параметры передачи можно изменять между сеансами обмена с устройствами.

- *STATUS spiSetBitOrderLsbFirst* (unsigned int n, *bool* enable)
Выбор порядка следования бит.
- *STATUS spiSetClkFrequency* (unsigned int n, unsigned int f)
*Задать частоту тактового сигнала **CLK** на шине **SPI**.*
- *STATUS spiSetClkInitialHigh* (unsigned int n, *bool* enable)
*Задать начальный уровень сигнала **CLK**.*
- *STATUS spiSetClkTrailingEdge* (unsigned int n, *bool* enable)
*Задать фронт фиксации данных сигнала **CLK** на шине **SPI**.*
- *STATUS spiSetCsInitialHigh* (unsigned int n, *bool* enable)
*Задать начальный уровень сигнала **Chip Select**.*

Инициализация аппаратного модуля SPI

- *STATUS spiInit* (unsigned int n, unsigned int gpion)
*Инициализация аппаратного модуля **SPI**.*
- *STATUS spiInitChipSelectLine* (unsigned int n, unsigned int gpion, unsigned int csn)
*Задать аппаратное управление линий **CS**.*

Обмен данными по шине SPI

- *STATUS spiTransfer* (unsigned int n, unsigned int csn, void *data, unsigned int txLength, unsigned int totalLength)
*Запуск обмена по шине данных **SPI**.*

19.62.1. Подробное описание

Настройка аппаратного модуля **SPI**.

Подключение:

```
#include <spi.h>
```

См. также

Общее описание работы с интерфейсом **SPI** в разделе *SPI*.

19.62.2. Макросы

19.62.2.1. SPI_0

```
#define SPI_0 0
```

Индекс для **SPI0**.

19.62.2.2. SPI_1

```
#define SPI_1 1
```

Индекс для **SPI1**.

19.62.2.3. SPI_2

```
#define SPI_2 2
```

Индекс для **SPI2**.

19.62.2.4. SPI_3

```
#define SPI_3 3
```

Индекс для **SPI3**.

19.62.2.5. SPI_CS_0

```
#define SPI_CS_0 0
```

Номер линии Chip Select **0**.

19.62.2.6. SPI_CS_1

```
#define SPI_CS_1 1
```

Номер линии Chip Select **1**.

19.62.2.7. SPI_GPIO_ADDITIONAL

```
#define SPI_GPIO_ADDITIONAL 1
```

Выбор дополнительного набора линий модуля SPI (обычно расположен в старших портах **GPIO**).

19.62.2.8. SPI_GPIO_DEFAULT

```
#define SPI_GPIO_DEFAULT 0
```

Выбор основного набора линий модуля SPI (обычно расположен в младших портах **GPIO**).

19.62.3. Функции

19.62.3.1. spiInit()

```
STATUS spiInit (  
    unsigned int n,  
    unsigned int gpion )
```

Функция инициализации аппаратного модуля **SPI** настраивает и запускает аппаратный модуль.

Аргументы

<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>gpion</i>	Номер набора портов ввода/вывода. Рекомендуется использовать значение из группы <i>макросов</i> .

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.62.3.2. spiInitChipSelectLine()

```
STATUS spiInitChipSelectLine (  
    unsigned int n,  
    unsigned int gpion,  
    unsigned int csn )
```

Функция подключает заданную линию **Chip Select** к аппаратному модулю **SPI**. К одному модулю могут быть подключены несколько линий **CS**, в этом случае следует вызвать функцию для каждой используемой линии. Далее управление линией происходит аппаратно при вызове *spiTransfer()*. При использовании программного управления выбором устройств данную функцию вызывать не нужно.

Аргументы

<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>gpion</i>	Номер набора портов ввода/вывода. Рекомендуется использовать значение из группы <i>макросов</i> .
<i>csn</i>	Номер линии Chip Select , если такая линия задана с помощью <i>spiInitChipSelectLine()</i> . Рекомендуется использовать значение из группы <i>макросов</i> .

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.62.3.3. spiSetBitOrderLsbFirst()

STATUS spiSetBitOrderLsbFirst (
 unsigned int *n*,
 bool *enable*)

Функция задаёт порядок следования бит при передаче по шине **SPI**. Параметр можно изменять в паузах между передачами разным устройствам. Если данный параметр не изменяется в ходе выполнения программы и не отличается от значения по умолчанию – данную функцию вызывать не обязательно.

Аргументы

<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>enable</i>	<i>true</i> – младший бит передаётся первым, иначе первым передаётся старший бит (значение по умолчанию).

Возвращает

OK в случае успешного выполнения настройки, иначе *ERROR*.

19.62.3.4. spiSetClkFrequency()

STATUS spiSetClkFrequency (
 unsigned int *n*,
 unsigned int *f*)

Функция задаёт частоту тактового сигнала **CLK**. Реальное значение тактовой частоты будет совпадать или максимально приближаться к заданному значению. Параметр можно изменять в паузах между передачами разным устройствам. Если данный параметр не изменяется в ходе выполнения программы и не отличается от значения по умолчанию – данную функцию вызывать не обязательно.

Аргументы

<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>f</i>	Тактовая частота сигнала CLK в герцах. Значение по умолчанию – 1 МГц.

Возвращает

OK в случае успешного выполнения настройки, иначе *ERROR*.

19.62.3.5. spiSetClkInitialHigh()

STATUS spiSetClkInitialHigh (
 unsigned int *n*,
 bool *enable*)

Функция задаёт начальный уровень сигнала **CLK** на шине **SPI**. Параметр можно изменять в паузах между передачами разным устройствам. Если данный параметр не изменяется в ходе выполнения программы и не отличается от значения по умолчанию – данную функцию вызывать не обязательно.

Аргументы

<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>enable</i>	<i>true</i> – высокий начальный уровень сигнала CLK , иначе начальным является низкий уровень (значение по умолчанию).

Возвращает

OK в случае успешного выполнения настройки, иначе *ERROR*.

19.62.3.6. spiSetClkTrailingEdge()

STATUS spiSetClkTrailingEdge (
 unsigned int *n*,
 bool *enable*)

Функция задаёт фронт сигнала **CLK**, по которому будут считываться данные. Параметр можно изменять в паузах между передачами разным устройствам. Если данный параметр не изменяется в ходе выполнения программы и не отличается от значения по умолчанию – данную функцию вызывать не обязательно.

Аргументы

<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>enable</i>	<i>true</i> – фиксация данных происходит по заднему фронту сигнала CLK , иначе фиксация по переднему фронту (значение по умолчанию).

Возвращает

OK в случае успешного выполнения настройки, иначе *ERROR*.

19.62.3.7. spiSetCsInitialHigh()

STATUS spiSetCsInitialHigh (
 unsigned int *n*,
 bool *enable*)

Функция задаёт начальный уровень сигнала **CS** на шине **SPI**. Настройка относится ко всем имеющимся в выбранном модуле линиям **CS**. Параметр можно изменять в паузах между передачами разным устройствам. Если данный параметр не изменяется в ходе выполнения программы и не отличается от значения по умолчанию – данную функцию вызывать не обязательно.

Аргументы	
<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>enable</i>	<i>true</i> – высокий начальный уровень сигнала CS , иначе начальным является низкий уровень (значение по умолчанию).

Возвращает

OK в случае успешного выполнения настройки, иначе *ERROR*.

19.62.3.8. spiTransfer()

STATUS spiTransfer (
 unsigned int *n*,
 unsigned int *csn*,
 void * *data*,
 unsigned int *txLength*,
 unsigned int *totalLength*)

Функция запускает побайтовый обмен данными по шине **SPI** и ожидает его окончания. Во время ожидания управление передаётся другим задачам.

Аргументы	
<i>n</i>	Номер модуля SPI рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>csn</i>	Номер линии Chip Select , если такая линия задана с помощью <i>spiInitChipSelectLine()</i> . Рекомендуется использовать значение из группы <i>макросов</i> . Если используется программное управление линиями выбора ведомых устройств, то данный параметр будет проигнорирован (в этом случае следует использовать csn=0).
<i>data</i>	Указатель на буфер обмен данными. Перед началом отправки буфер должен содержать отправляемые данные, если таковые имеются. После окончания отправки в буфере будут храниться данные полученные от выбранного устройства. Размер буфера должен быть достаточным для приёма запрашиваемого количества данных.
<i>txLength</i>	Количество байт для передачи. Если количество передаваемых байт меньше, чем общее количество обменов по шине, недостающие байты при передаче будут заполнены значениями по умолчанию (нулями). Количество передаваемых байт не может превышать общее количество обменов totalLength .
<i>totalLength</i>	Общее количество обменов байтами по шине. Максимально возможное значение – 64 байта.

Возвращает

OK если обмен данными прошёл успешно, иначе *ERROR*.

19.63. Файл `stdarg.h`

Макросы для поддержки функций с неопределенным числом аргументов неопределенного типа.

Макросы

- `#define va_arg(vaList, type) __builtin_va_arg(vaList, type)`
- `#define va_copy(vaListDest, vaListSrc) __builtin_va_copy(vaListDest, vaListSrc)`
- `#define va_end(vaList) __builtin_va_end(vaList)`
- `#define va_start(vaList, parmN) __builtin_va_start(vaList, parmN)`

Определения типов

- `typedef __builtin_va_list va_list`

19.63.1. Подробное описание

См. стандарт C11 7.16.

См. также

[C11 standard 7.16.](#)

19.63.2. Макросы

19.63.2.1. `va_arg`

```
#define va_arg(  
    vaList,  
    type ) __builtin_va_arg(vaList, type)
```

Макрос, позволяющий последовательно получать неопределенные аргументы функции.

Аргументы

`vaList` Объект типа `va_list`.

`type` Тип аргумента, который следует извлечь.

19.63.2.2. `va_copy`

```
#define va_copy(  
    vaListDest,  
    vaListSrc ) __builtin_va_copy(vaListDest, vaListSrc)
```

Инициализировать объект типа `va_list` и скопировать туда другой подобный объект.

Аргументы

`vaListDest` Объект типа `va_list`, который будет проинициализирован.

Продолжение на следующей странице

Аргументы (Продолжение.)

vaListSrc Объект типа *va_list*, данные которого будут скопированы в **vaListDest**.

19.63.2.3. *va_end*

```
#define va_end(  
    vaList ) __builtin_va_end(vaList)
```

Освободить объект типа *va_list*.

Аргументы

vaList Объект типа *va_list*.

19.63.2.4. *va_start*

```
#define va_start(  
    vaList,  
    parmN ) __builtin_va_start(vaList, parmN)
```

Инициализировать объект типа *va_list*.

Аргументы

vaList Объект типа *va_list*.

parmN Последний аргумент функции, стоящий перед ', ...'.

19.63.3. Типы

19.63.3.1. *va_list*

```
typedef __builtin_va_list va_list
```

Тип, позволяющий работать с макросами *va_start*, *va_end*, *va_arg* и *va_copy*.

19.64. Файл `stdbool.h`

Стандартные логические типы данных.

Макросы

- `#define __bool_true_false_are_defined 1`
Проверка наличия определения логических типов.
- `#define bool int`
Определение типа `bool`.
- `#define false 0`
Определение типа `false`.
- `#define true 1`
Определение типа `true`.

19.64.1. Подробное описание

В файле описаны логические типы данных, появившиеся в новом стандарте **C11**.

См. также

[C11 standard 7.18](#).

19.64.2. Макросы

19.64.2.1. `__bool_true_false_are_defined`

```
#define __bool_true_false_are_defined 1
```

Данный макрос можно использовать для проверки наличия определения типов **true** и **false**.

19.64.2.2. `bool`

```
#define bool int
```

Стандарт требует, чтобы макрос **bool** был определен как **_Bool** (в имплементации используемого компилятора - 1 байт), но в исходном коде *MULTEX-ARM* в течении очень долгого времени использовалось определение через **4x** байтный **int**. Соответственно, смена макроса всё ломает. В связи с этим сделана поправка для данного типа.

19.64.2.3. `false`

```
#define false 0
```

Определение типа **false** по стандарту **C11**.

19.64.2.4. `true`

```
#define true 1
```

Определение типа **true** по стандарту **C11**.

19.65. Файл `stddef.h`

Стандартные определения.

Макросы

- `#define NULL ((void*)0)`
- `#define offsetof(TYPE, MEMBER) __builtin_offsetof (TYPE, MEMBER)`

Определения типов

- `typedef int ptrdiff_t`
- `typedef int wchar_t`

Функции

- `typedef __typeof (sizeof(0)) size_t`

19.65.1. Подробное описание

См. также

[C11 standard 7.19.](#)

19.65.2. Макросы

19.65.2.1. NULL

```
#define NULL ((void*)0)
```

Константа нулевого указателя.

19.65.2.2. `offsetof`

```
#define offsetof(  
    TYPE,  
    MEMBER) __builtin_offsetof (TYPE, MEMBER)
```

Вычисление смещение в байтах поля структуры или объединения от начала структуры или объединения.

19.65.3. Типы

19.65.3.1. `ptrdiff_t`

```
typedef int ptrdiff_t
```

Тип, представляющий результат операции вычитания над двумя указателями.

19.65.3.2. `wchar_t`

```
typedef int wchar_t
```

Тип, представляющий широкий символ.

19.65.4. Функции

19.65.4.1. `__typeof()`

```
typedef __typeof (  
    sizeof(0) )
```

Тип, представляющий возврат оператора **sizeof**. Определён как **`__typeof(sizeof(0))`**.

19.66. Файл `stdint.h`

Целочисленные типы заданного размера.

Максимальные и минимальные значения типов, определенных в `limits.h`

- `#define INT16_MAX (SHRT_MAX)`
- `#define INT16_MIN (SHRT_MIN)`
- `#define INT32_MAX (INT_MAX)`
- `#define INT32_MIN (INT_MIN)`
- `#define INT64_MAX (INT64_C(LLONG_MAX))`
- `#define INT64_MIN (INT64_C(LLONG_MIN))`
- `#define INT8_MAX (SCHAR_MAX)`
- `#define INT8_MIN (SCHAR_MIN)`
- `#define INT_FAST16_MAX (INT_MAX)`
- `#define INT_FAST16_MIN (INT_MIN)`
- `#define INT_FAST32_MAX (INT_MAX)`
- `#define INT_FAST32_MIN (INT_MIN)`
- `#define INT_FAST64_MAX (INT64_C(LLONG_MAX))`
- `#define INT_FAST64_MIN (INT64_C(LLONG_MIN))`
- `#define INT_FAST8_MAX (SCHAR_MAX)`
- `#define INT_FAST8_MIN (SCHAR_MIN)`
- `#define INT_LEAST16_MAX (SHRT_MAX)`
- `#define INT_LEAST16_MIN (SHRT_MIN)`
- `#define INT_LEAST32_MAX (INT_MAX)`
- `#define INT_LEAST32_MIN (INT_MIN)`
- `#define INT_LEAST64_MAX (INT64_C(LLONG_MAX))`
- `#define INT_LEAST64_MIN (INT64_C(LLONG_MIN))`
- `#define INT_LEAST8_MAX (SCHAR_MAX)`
- `#define INT_LEAST8_MIN (SCHAR_MIN)`
- `#define INTMAX_MAX (INT64_C(LLONG_MAX))`
- `#define INTMAX_MIN (INT64_C(LLONG_MIN))`
- `#define INTPTR_MAX (INT_MAX)`
- `#define INTPTR_MIN (INT_MIN)`
- `#define UINT16_MAX (USHRT_MAX)`
- `#define UINT32_MAX (UINT_MAX)`
- `#define UINT64_MAX (UINT64_C(ULLONG_MAX))`
- `#define UINT8_MAX (UCHAR_MAX)`
- `#define UINT_FAST16_MAX (UINT_MAX)`
- `#define UINT_FAST32_MAX (UINT_MAX)`
- `#define UINT_FAST64_MAX (UINT64_C(ULLONG_MAX))`
- `#define UINT_FAST8_MAX (UCHAR_MAX)`
- `#define UINT_LEAST16_MAX (USHRT_MAX)`
- `#define UINT_LEAST32_MAX (UINT_MAX)`
- `#define UINT_LEAST64_MAX (UINT64_C(ULLONG_MAX))`
- `#define UINT_LEAST8_MAX (UCHAR_MAX)`
- `#define UINTMAX_MAX (UINT64_C(ULLONG_MAX))`
- `#define UINTPTR_MAX (UINT_MAX)`

Максимальные и минимальные значения некоторых дополнительных типов

- `#define PTRDIFF_MAX (INT_MAX)`
- `#define PTRDIFF_MIN (INT_MIN)`
- `#define SIG_ATOMIC_MAX (INT_MAX)`
- `#define SIG_ATOMIC_MIN (INT_MIN)`
- `#define SIZE_MAX (UINT_MAX)`
- `#define WCHAR_MAX (INT_MAX)`
- `#define WCHAR_MIN (INT_MIN)`
- `#define WINT_MAX (INT_MAX)`
- `#define WINT_MIN (INT_MIN)`

Макросы для приведения констант к типам с минимально-заданным размером

- #define *INT16_C*(c) c
- #define *INT32_C*(c) c
- #define *INT64_C*(c) c ## LL
- #define *INT8_C*(c) c
- #define *INTMAX_C*(c) c ## LL
- #define *UINT16_C*(c) c
- #define *UINT32_C*(c) c
- #define *UINT64_C*(c) c ## ULL
- #define *UINT8_C*(c) c
- #define *UINTMAX_C*(c) c ## ULL

Типы заданного размера

- typedef signed short *int16_t*
- typedef signed int *int32_t*
- typedef signed long long *int64_t*
- typedef signed char *int8_t*
- typedef unsigned short *uint16_t*
- typedef unsigned int *uint32_t*
- typedef unsigned long long *uint64_t*
- typedef unsigned char *uint8_t*

Типы с минимально-заданным размером

- typedef signed short *int_least16_t*
- typedef signed int *int_least32_t*
- typedef signed long long *int_least64_t*
- typedef signed char *int_least8_t*
- typedef unsigned short *uint_least16_t*
- typedef unsigned int *uint_least32_t*
- typedef unsigned long long *uint_least64_t*
- typedef unsigned char *uint_least8_t*

Типы с максимальным быстродействием

- typedef signed int *int_fast16_t*
- typedef signed int *int_fast32_t*
- typedef signed long long *int_fast64_t*
- typedef signed char *int_fast8_t*
- typedef unsigned int *uint_fast16_t*
- typedef unsigned int *uint_fast32_t*
- typedef unsigned long long *uint_fast64_t*
- typedef unsigned char *uint_fast8_t*

Типы для хранения указателей

- typedef int *intptr_t*
- typedef unsigned int *uintptr_t*

Типы максимального размера

- typedef signed long long int *intmax_t*
- typedef unsigned long long int *uintmax_t*

19.66.1. Подробное описание

См. стандарт C11 7.20.

См. также

[C11 standard 7.20.](#)

19.66.2. Макросы

19.66.2.1. INT16_C

```
#define INT16_C(  
    c) c
```

19.66.2.2. INT16_MAX

```
#define INT16_MAX (SHRT_MAX)
```

19.66.2.3. INT16_MIN

```
#define INT16_MIN (SHRT_MIN)
```

19.66.2.4. INT32_C

```
#define INT32_C(  
    c) c
```

19.66.2.5. INT32_MAX

```
#define INT32_MAX (INT_MAX)
```

19.66.2.6. INT32_MIN

```
#define INT32_MIN (INT_MIN)
```

19.66.2.7. INT64_C

```
#define INT64_C(  
    c) c ## LL
```

19.66.2.8. INT64_MAX

```
#define INT64_MAX (INT64_C(LLONG_MAX))
```

19.66.2.9. INT64_MIN

```
#define INT64_MIN (INT64_C(LLONG_MIN))
```

19.66.2.10. INT8_C

```
#define INT8_C(  
    c) c
```

19.66.2.11. INT8_MAX

```
#define INT8_MAX (SCHAR_MAX)
```

19.66.2.12. INT8_MIN

```
#define INT8_MIN (SCHAR_MIN)
```

19.66.2.13. INT_FAST16_MAX

```
#define INT_FAST16_MAX (INT_MAX)
```

19.66.2.14. INT_FAST16_MIN

```
#define INT_FAST16_MIN (INT_MIN)
```

19.66.2.15. INT_FAST32_MAX

```
#define INT_FAST32_MAX (INT_MAX)
```

19.66.2.16. INT_FAST32_MIN

```
#define INT_FAST32_MIN (INT_MIN)
```

19.66.2.17. INT_FAST64_MAX

```
#define INT_FAST64_MAX (INT64_C(LLONG_MAX))
```

19.66.2.18. INT_FAST64_MIN

```
#define INT_FAST64_MIN (INT64_C(LLONG_MIN))
```

19.66.2.19. INT_FAST8_MAX

```
#define INT_FAST8_MAX (SCHAR_MAX)
```

19.66.2.20. INT_FAST8_MIN

```
#define INT_FAST8_MIN (SCHAR_MIN)
```

19.66.2.21. INT_LEAST16_MAX

```
#define INT_LEAST16_MAX (SHRT_MAX)
```

19.66.2.22. INT_LEAST16_MIN

```
#define INT_LEAST16_MIN (SHRT_MIN)
```

19.66.2.23. INT_LEAST32_MAX

```
#define INT_LEAST32_MAX (INT_MAX)
```

19.66.2.24. INT_LEAST32_MIN

```
#define INT_LEAST32_MIN (INT_MIN)
```

19.66.2.25. INT_LEAST64_MAX

```
#define INT_LEAST64_MAX (INT64_C(LLONG_MAX))
```

19.66.2.26. INT_LEAST64_MIN

```
#define INT_LEAST64_MIN (INT64_C(LLONG_MIN))
```

19.66.2.27. INT_LEAST8_MAX

```
#define INT_LEAST8_MAX (SCHAR_MAX)
```

19.66.2.28. INT_LEAST8_MIN

```
#define INT_LEAST8_MIN (SCHAR_MIN)
```

19.66.2.29. INTMAX_C

```
#define INTMAX_C(  
    c) c ## LL
```

19.66.2.30. INTMAX_MAX

```
#define INTMAX_MAX (INT64_C(LLONG_MAX))
```

19.66.2.31. INTMAX_MIN

```
#define INTMAX_MIN (INT64_C(LLONG_MIN))
```

19.66.2.32. INTPTR_MAX

```
#define INTPTR_MAX (INT_MAX)
```

19.66.2.33. INTPTR_MIN

```
#define INTPTR_MIN (INT_MIN)
```

19.66.2.34. PTRDIFF_MAX

```
#define PTRDIFF_MAX (INT_MAX)
```

19.66.2.35. PTRDIFF_MIN

```
#define PTRDIFF_MIN (INT_MIN)
```


19.66.2.36. SIG_ATOMIC_MAX

```
#define SIG_ATOMIC_MAX (INT_MAX)
```

19.66.2.37. SIG_ATOMIC_MIN

```
#define SIG_ATOMIC_MIN (INT_MIN)
```

19.66.2.38. SIZE_MAX

```
#define SIZE_MAX (UINT_MAX)
```

19.66.2.39. UINT16_C

```
#define UINT16_C(  
    c) c
```

19.66.2.40. UINT16_MAX

```
#define UINT16_MAX (USHRT_MAX)
```

19.66.2.41. UINT32_C

```
#define UINT32_C(  
    c) c
```

19.66.2.42. UINT32_MAX

```
#define UINT32_MAX (UINT_MAX)
```

19.66.2.43. UINT64_C

```
#define UINT64_C(  
    c) c ## ULL
```

19.66.2.44. UINT64_MAX

```
#define UINT64_MAX (UINT64_C(ULLONG_MAX))
```

19.66.2.45. UINT8_C

```
#define UINT8_C(  
    c) c
```

19.66.2.46. UINT8_MAX

```
#define UINT8_MAX (UCHAR_MAX)
```

19.66.2.47. UINT_FAST16_MAX

```
#define UINT_FAST16_MAX (UINT_MAX)
```

19.66.2.48. UINT_FAST32_MAX

```
#define UINT_FAST32_MAX (UINT_MAX)
```

19.66.2.49. UINT_FAST64_MAX

```
#define UINT_FAST64_MAX (UINT64_C(ULLONG_MAX))
```

19.66.2.50. UINT_FAST8_MAX

```
#define UINT_FAST8_MAX (UCHAR_MAX)
```

19.66.2.51. UINT_LEAST16_MAX

```
#define UINT_LEAST16_MAX (USHRT_MAX)
```

19.66.2.52. UINT_LEAST32_MAX

```
#define UINT_LEAST32_MAX (UINT_MAX)
```

19.66.2.53. UINT_LEAST64_MAX

```
#define UINT_LEAST64_MAX (UINT64_C(ULLONG_MAX))
```

19.66.2.54. UINT_LEAST8_MAX

```
#define UINT_LEAST8_MAX (UCHAR_MAX)
```

19.66.2.55. UINTMAX_C

```
#define UINTMAX_C(  
    c) c ## ULL
```

19.66.2.56. UINTMAX_MAX

```
#define UINTMAX_MAX (UINT64_C(ULLONG_MAX))
```

19.66.2.57. UINTPTR_MAX

```
#define UINTPTR_MAX (UINT_MAX)
```

19.66.2.58. WCHAR_MAX

```
#define WCHAR_MAX (INT_MAX)
```

19.66.2.59. WCHAR_MIN

```
#define WCHAR_MIN (INT_MIN)
```

19.66.2.60. WINT_MAX

```
#define WINT_MAX (INT_MAX)
```

19.66.2.61. WINT_MIN

```
#define WINT_MIN (INT_MIN)
```

19.66.3. Типы**19.66.3.1. int16_t**

```
typedef signed short int16_t
```

19.66.3.2. int32_t

typedef signed int *int32_t*

19.66.3.3. int64_t

typedef signed long long *int64_t*

19.66.3.4. int8_t

typedef signed char *int8_t*

19.66.3.5. int_fast16_t

typedef signed int *int_fast16_t*

19.66.3.6. int_fast32_t

typedef signed int *int_fast32_t*

19.66.3.7. int_fast64_t

typedef signed long long *int_fast64_t*

19.66.3.8. int_fast8_t

typedef signed char *int_fast8_t*

19.66.3.9. int_least16_t

typedef signed short *int_least16_t*

19.66.3.10. int_least32_t

typedef signed int *int_least32_t*

19.66.3.11. int_least64_t

typedef signed long long *int_least64_t*

19.66.3.12. int_least8_t

typedef signed char *int_least8_t*

19.66.3.13. intmax_t

typedef signed long long int *intmax_t*

19.66.3.14. intptr_t

typedef int *intptr_t*

19.66.3.15. uint16_t

typedef unsigned short *uint16_t*

19.66.3.16. uint32_t

typedef unsigned int *uint32_t*

19.66.3.17. uint64_t

typedef unsigned long long *uint64_t*

19.66.3.18. uint8_t

typedef unsigned char *uint8_t*

19.66.3.19. uint_fast16_t

typedef unsigned int *uint_fast16_t*

19.66.3.20. uint_fast32_t

typedef unsigned int *uint_fast32_t*

19.66.3.21. uint_fast64_t

typedef unsigned long long *uint_fast64_t*

19.66.3.22. uint_fast8_t

typedef unsigned char *uint_fast8_t*

19.66.3.23. uint_least16_t

typedef unsigned short *uint_least16_t*

19.66.3.24. uint_least32_t

typedef unsigned int *uint_least32_t*

19.66.3.25. uint_least64_t

typedef unsigned long long *uint_least64_t*

19.66.3.26. uint_least8_t

typedef unsigned char *uint_least8_t*

19.66.3.27. uintmax_t

typedef unsigned long long int *uintmax_t*

19.66.3.28. uintptr_t

typedef unsigned int *uintptr_t*

19.67. Файл `stdio.h`

Стандартные функции ввода-вывода.

Структуры данных

- `struct FILE`
Структура файла на блочном устройстве.

Макросы

- `#define BUFSIZ (8192)`
Размер буфера `setbuf` по-умолчанию.
- `#define EOF (-1)`
Значение, индицирующее ситуацию 'конец файла' для потока.
- `#define FILE_SIGNATURE 0xF12EF12E`
Сигнатура корректной структуры `FILE`.
- `#define FILENAME_MAX (256)`
Максимально допустимая длина строки, которая разрешена системой для определения имени файла.
- `#define FOPEN_MAX (16)`
Минимальное количество файлов, которое гарантированно возможно иметь одновременно открытыми.
- `#define L_tmpnam (128)`
Длина имен временных файлов, создаваемых `tmpnam`.
- `#define TMP_MAX (10)`
Максимальное число уникальных имен файлов, которые может создать функция `tmpnam`.

Определения типов

- `typedef long int fpos_t`
Тип, позволяющий однозначно указать позицию в файле.
- `#define _IOFBF 0`
Значения для 3го аргумента функции `setvbuf`.
- `#define _IOLBF 1`
Линейная/строчная буферизация.
- `#define _IONBF 2`
Отключенная буферизация.
- `#define SEEK_CUR 1`
Позицию нужно сдвигать относительно текущей позиции в файле.
- `#define SEEK_END 2`
Позицию нужно сдвигать относительно конца файла.
- `#define SEEK_SET 0`
Значения для третьего аргумента функции `fseek`, относительно чего нужно сдвигать позицию чтения/записи в файле.
- `FILE * stderr`
Стандартный поток вывода сообщений об ошибках.
- `FILE * stdin`
Стандартные потоки.
- `FILE * stdout`
Стандартный поток вывода.

Функции операций над файлами

- `int remove (const char *filename)`
- `int rename (const char *oldName, const char *newName)`

Функции доступа к файлам

- int *fclose* (*FILE* *stream)
- *FILE* * *fdopen* (int fd, const char *mode)
- int *fflush* (*FILE* *stream)
- *FILE* * *fopen* (const char *restrict filename, const char *restrict mode)
- *FILE* * *freopen* (const char *restrict filename, const char *restrict mode, *FILE* *restrict stream)
- void *setbuf* (*FILE* *restrict stream, char *restrict buf)
- int *setvbuf* (*FILE* *restrict stream, char *restrict buf, int mode, size_t size)

Функции форматированного ввода-вывода

- int *fprintf* (*FILE* *restrict stream, const char *restrict format,...)
- int *fscanf* (*FILE* *restrict stream, const char *restrict format,...)
- int *printf* (const char *restrict format,...)
- int *scanf* (const char *restrict format,...)
- int *snprintf* (char *restrict s, size_t n, const char *restrict format,...)
- int *sprintf* (char *restrict s, const char *restrict format,...)
- int *sscanf* (const char *restrict s, const char *restrict format,...)
- int *vfprintf* (*FILE* *restrict stream, const char *restrict format, *va_list* arg)
- int *vscanf* (*FILE* *restrict stream, const char *restrict format, *va_list* arg)
- int *vsnprintf* (char *restrict s, size_t n, const char *restrict format, *va_list* arg)
- int *vsprintf* (char *restrict s, const char *restrict format, *va_list* arg)
- int *vsscanf* (const char *restrict s, const char *restrict format, *va_list* arg)

Функции ввода-вывода символов

- int *fgetc* (*FILE* *stream)
- char * *fgets* (char *restrict s, int n, *FILE* *restrict stream)
- int *fputc* (int c, *FILE* *stream)
- int *fputs* (const char *restrict s, *FILE* *restrict stream)
- int *getc* (*FILE* *stream)
- int *getchar* (void)
- char * *gets* (char *s)
- int *putbytes* (char *ch, int len)
- int *putc* (int c, *FILE* *stream)
- int *putchar* (int c)
- int *puts* (const char *s)
- int *ungetc* (int c, *FILE* *stream)

Функции прямого ввода-вывода

- size_t *fread* (void *restrict ptr, size_t size, size_t nmemb, *FILE* *restrict stream)
- size_t *fwrite* (const void *restrict ptr, size_t size, size_t nmemb, *FILE* *restrict stream)

Функции позиционирования в файлах

- int *fgetpos* (*FILE* *restrict stream, *fpos_t* *restrict pos)
- int *fseek* (*FILE* *stream, long int offset, int whence)
- int *fsetpos* (*FILE* *stream, const *fpos_t* *pos)
- long int *ftell* (*FILE* *stream)
- void *rewind* (*FILE* *stream)

Функции обработки ошибок

- void *clearerr* (*FILE* *stream)
- int *feof* (*FILE* *stream)
- int *ferror* (*FILE* *stream)

Прочие нестандартные функции

- int *getch* (void)
- int *printerr* (const char *restrict format,...)

19.67.1. Подробное описание

См. стандарт C11 7.21.

См. также

[C11 standard 7.21.](#)

19.67.2. Макросы

19.67.2.1. _IOFBF

```
#define _IOFBF 0
```

Полная/блочная буферизация.

19.67.2.2. _IOLBF

```
#define _IOLBF 1
```

19.67.2.3. _IONBF

```
#define _IONBF 2
```

19.67.2.4. BUFSIZ

```
#define BUFSIZ (8192)
```

19.67.2.5. EOF

```
#define EOF (-1)
```

19.67.2.6. FILE_SIGNATURE

```
#define FILE_SIGNATURE 0xF12EF12E
```

19.67.2.7. FILENAME_MAX

```
#define FILENAME_MAX (256)
```

19.67.2.8. FOPEN_MAX

```
#define FOPEN_MAX (16)
```

19.67.2.9. L_tmpnam

```
#define L_tmpnam (128)
```

19.67.2.10. SEEK_CUR

```
#define SEEK_CUR 1
```

19.67.2.11. SEEK_END

```
#define SEEK_END 2
```

19.67.2.12. SEEK_SET

```
#define SEEK_SET 0
```

Позицию нужно сдвигать относительно начала файла.

19.67.2.13. TMP_MAX

```
#define TMP_MAX (10)
```

19.67.3. Типы**19.67.3.1. fpos_t**

```
typedef long int fpos_t
```

19.67.4. Функции**19.67.4.1. clearerr()**

```
void clearerr (  
    FILE * stream )
```

Очистить указатели конца файла и ошибок для потока.

Аргументы

stream Поток.



Учет ошибок фактически не ведется.

19.67.4.2. fclose()

```
int fclose (
    FILE * stream )
```

Закрывает поток, ранее открытый с помощью `fdopen` или `freopen`.

Аргументы

stream Поток, который требуется закрыть.

Возвращает

0 при успехе, EOF иначе.

Функция также вызовет принудительную буферизацию данных.

19.67.4.3. fdopen()

```
FILE* fdopen (
    int fd,
    const char * mode )
```

Связать с потоком открытый файл.

Аргументы

fd Дескриптор открытого файла.

mode Строка, описывающая режим открытия (r, r+, w, w+, a, a+). Режим должен быть совместим с режимом открытого файла.

Возвращает

Указатель на открытый поток или NULL при ошибке.

19.67.4.4. feof()

```
int feof (
```

*FILE * stream*)

Проверить указатель конца файла для потока.

Аргументы

stream Поток.

Возвращает

Не-0, если поток указывает на конец файла.

19.67.4.5. **ferror()**

int ferror (
*FILE * stream*)

Проверить наличие ошибок потока.

Аргументы

stream Поток.

Возвращает

Не-0, если у потока присутствуют ошибки.



Учет ошибок не ведется, фактически всегда возвращается 0.

19.67.4.6. **fflush()**

int fflush (
*FILE * stream*)

Принудительно буферизировать данные, не закрывая поток.

Аргументы

stream Поток.

Возвращает

0 при успехе, EOF иначе.

19.67.4.7. fgetc()

```
int fgetc (
    FILE * stream )
```

Считать символ из потока.

Аргументы

<i>stream</i>	Поток.
---------------	--------

Возвращает

Символ, приведенный к int, или EOF при достижении конца файла или при ошибке.

19.67.4.8. fgetpos()

```
int fgetpos (
    FILE *restrict stream,
    fpos_t *restrict pos )
```

Получить текущую позицию операции ввода-вывода в потоке, сохранив результат в виде типа fpos_t.

Аргументы

<i>stream</i>	Поток.
---------------	--------

<i>pos</i>	Буфер для позиции ввода-вывода.
------------	---------------------------------

Возвращает

0 при успехе, -1 иначе.

19.67.4.9. fgets()

```
char* fgets (
    char *restrict s,
    int n,
    FILE *restrict stream )
```

Считать строку из потока и записать ее в буфер, нуль-терминировать итоговую строку.

Аргументы

<i>s</i>	Буфер для записи.
----------	-------------------

<i>n</i>	Размер буфера.
----------	----------------

<i>stream</i>	Поток.
---------------	--------

Возвращает

s при успехе, NULL при ошибке или достижении конца файла/потока.

19.67.4.10. fopen()

```
FILE* fopen (
    const char *restrict filename,
    const char *restrict mode )
```

Открыть файл и связать его с потоком.

Аргументы

filename Имя файла.

mode Строка, описывающая режим открытия (r, r+, w, w+, a, a+).

Возвращает

Указатель на открытый поток или NULL при ошибке.

19.67.4.11. fprintf()

```
int fprintf (
    FILE *restrict stream,
    const char *restrict format,
    ... )
```

Вывести текст в поток в соответствии с форматной строкой.

Аргументы

stream Поток вывода.

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.12. fputc()

```
int fputc (
    int c,
    FILE * stream )
```

Вывести символ в поток.

Аргументы

c Символ, приведенный к `unsigned char`.

stream Поток.

Возвращает

Символ, обратно приведенный к `int`, или EOF при ошибке.

19.67.4.13. `fputs()`

```
int fputs (  
    const char *restrict s,  
    FILE *restrict stream )
```

Вывести строку в поток, за исключением нуль-терминатора.

Аргументы

s Выводимая строка.

stream Поток.

Возвращает

Неотрицательное число при успехе, EOF иначе.

19.67.4.14. `fread()`

```
size_t fread (  
    void *restrict ptr,  
    size_t size,  
    size_t nmemb,  
    FILE *restrict stream )
```

Считать данные из потока.

Аргументы

ptr Буфер для данных

size Размер элемента данных.

nmemb Кол-во элементов данных.

stream Поток.

Возвращает

Кол-во успешно считанных элементов (не символов/байт!).

19.67.4.15. freopen()

```
FILE* freopen (  
    const char *restrict filename,  
    const char *restrict mode,  
    FILE *restrict stream )
```

Открыть файл и связать его с существующим потоком.

Аргументы

filename Имя файла.

mode Строка, описывающая режим открытия (r, r+, w, w+, a, a+).

stream Существующий поток. Файл, связанный с этим потоком будет закрыт.

Возвращает

Указатель на открытый поток или NULL при ошибке.

19.67.4.16. fscanf()

```
int fscanf (  
    FILE *restrict stream,  
    const char *restrict format,  
    ... )
```

Считать информацию из потока в соответствии с форматной строкой.

Аргументы

stream Поток.

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во считанных элементов или EOF при ошибке.

19.67.4.17. fseek()


```
int fseek (
    FILE * stream,
    long int offset,
    int whence )
```

Установить позицию следующей операции ввода-вывода в потоке.

Аргументы

<i>stream</i>	Поток.
<i>offset</i>	Смещение от начальной позиции.
<i>whence</i>	Начальная позиция: SEEK_SET, SEEK_CUR или SEEK_END.

Возвращает

0 при успехе, -1 иначе.

19.67.4.18. fsetpos()

```
int fsetpos (
    FILE * stream,
    const fpos_t * pos )
```

Установить позицию следующей операции ввода-вывода в потоке, используя результат из функции fgetpos.

Аргументы

<i>stream</i>	Поток.
<i>pos</i>	Позиция ввода-вывода.

Возвращает

0 при успехе, -1 иначе.

19.67.4.19. ftell()

```
long int ftell (
    FILE * stream )
```

Получить текущую позицию операции ввода-вывода в потоке.

Аргументы

<i>stream</i>	Поток.
---------------	--------

Возвращает

Значение текущего смещения или -1 при ошибке.

19.67.4.20. fwrite()

```
size_t fwrite (  
    const void *restrict ptr,  
    size_t size,  
    size_t nmemb,  
    FILE *restrict stream )
```

Записать данные в поток.

Аргументы	
<i>ptr</i>	Буфер для данных
<i>size</i>	Размер элемента данных.
<i>nmemb</i>	Кол-во элементов данных.
<i>stream</i>	Поток.

Возвращает

Кол-во успешно записанных элементов (не символов/байт!).

19.67.4.21. getc()

```
int getc (  
    FILE * stream )
```

Считать символ из потока.

Аргументы	
<i>stream</i>	Поток.

Возвращает

Символ, приведенный к int, или EOF при достижении конца файла или при ошибке.

19.67.4.22. getch()

```
int getch (  
    void )
```

Считать символ из stdin, вернуть ошибку если символ не будет тут же получен.

Возвращает

Символ, приведенный к `int`, или EOF при достижении конца файла или при ошибке.

19.67.4.23. `getchar()`

```
int getchar (  
    void )
```

Считать символ из `stdin`.

Возвращает

Символ, приведенный к `int`, или EOF при достижении конца файла или при ошибке.

19.67.4.24. `gets()`

```
char* gets (  
    char * s )
```

Считать строку из `stdin` и записать ее в буфер.

Аргументы

`s` Буфер для записи, переполнение буфера не проверяется.

Возвращает

`s` при успехе, `NULL` при ошибке.

19.67.4.25. `printerr()`

```
int printerr (  
    const char *restrict format,  
    ... )
```

Вывести текст в `stderr` в соответствии с форматной строкой.

Аргументы

`format` Форматная строка.

`...` Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.26. printf()

```
int printf (
    const char *restrict format,
    ... )
```

Вывести текст в stdout в соответствии с форматной строкой.

Аргументы

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.27. putbytes()

```
int putbytes (
    char * ch,
    int len )
```

Вывести многобайтовый символ в stdout.

Аргументы

ch Указатель на строку с данными.

len Количество символов для вывода.

Возвращает

Количество записанных байт или EOF при ошибке.

19.67.4.28. putc()

```
int putc (
    int c,
    FILE * stream )
```

Вывести символ в поток.

Аргументы

`c` Символ, приведенный к `unsigned char`.

`stream` Поток.

Возвращает

Символ, обратно приведенный к `int`, или EOF при ошибке.

19.67.4.29. putchar()

```
int putchar (  
    int c )
```

Вывести символ в `stdout`.

Аргументы

`c` Символ, приведенный к `unsigned char`.

Возвращает

Символ, обратно приведенный к `int`, или EOF при ошибке.

19.67.4.30. puts()

```
int puts (  
    const char * s )
```

Вывести строку и завершающий перевод строки в `stdout`.

Аргументы

`s` Выводимая строка.

Возвращает

Неотрицательное число при успехе, EOF иначе.

19.67.4.31. remove()

```
int remove (  
    const char * filename )
```

Удалить файл.

Аргументы

filename Имя файла.

Возвращает

0 при успехе, не-0 иначе.

19.67.4.32. rename()

```
int rename (
    const char * oldName,
    const char * newName )
```

Переименовать файл.

Аргументы

oldName Старое имя файла.

newName Новое имя файла.

Возвращает

0 при успехе, не-0 иначе.

19.67.4.33. rewind()

```
void rewind (
    FILE * stream )
```

Установить позицию операции ввода-вывода в потоке в начало.

Аргументы

stream Поток.

19.67.4.34. scanf()

```
int scanf (
    const char *restrict format,
    ... )
```

Считать информацию из stdin в соответствии с форматной строкой.

Аргументы

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во считанных элементов или EOF при ошибке.

19.67.4.35. setbuf()

```
void setbuf (
    FILE *restrict stream,
    char *restrict buf )
```

Не используется: Установить буфер для буферизации потока.

Аргументы

stream Поток.

buf Указатель на буфер размера минимум BUFSIZ для включения блочной буферизации, или NULL для отключения буферизации.



Фактически всегда используется буферизация по-умолчанию (блочная для файлов, отключенная для терминала, по-разному для устройств).

19.67.4.36. setvbuf()

```
int setvbuf (
    FILE *restrict stream,
    char *restrict buf,
    int mode,
    size_t size )
```

Не используется: Изменить тип буферизации для потока и установить новый буфер.

Аргументы

stream Поток.

buf Буфер для буферизации.

mode Режим буферизации.

size Размер буфера.

Возвращает

0 при успехе, не-0 иначе.



Фактически всегда используется буферизация по-умолчанию (блочная для файлов, отключенная для терминала, по-разному для устройств).

19.67.4.37. `snprintf()`

```
int snprintf (  
    char *restrict s,  
    size_t n,  
    const char *restrict format,  
    ... )
```

Вывести текст в буфер в соответствии с форматной строкой.

Аргументы

s Буфер вывода.

n Максимальное количество символов, которое можно вывести в буфер вывода (включая нуль-терминатор).

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.38. `sprintf()`

```
int sprintf (  
    char *restrict s,  
    const char *restrict format,  
    ... )
```

Вывести текст в буфер в соответствии с форматной строкой.

Аргументы

s Буфер вывода.

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.39. `sscanf()`

```
int sscanf (
    const char *restrict s,
    const char *restrict format,
    ... )
```

Считать информацию из текстового буфера в соответствии с форматной строкой.

Аргументы

<code>s</code>	Текстовый буфер.
<code>format</code>	Форматная строка.
<code>...</code>	Аргументы для форматной строки.

Возвращает

Кол-во считанных элементов или EOF при ошибке.

19.67.4.40. `ungetc()`

```
int ungetc (
    int c,
    FILE * stream )
```

Занести символ обратно в поток, сделав его первым к чтению.

Аргументы

<code>c</code>	Символ для занесения.
<code>stream</code>	Поток.

Возвращает

`c` при успехе, EOF при ошибке.

19.67.4.41. `vfprintf()`

```
int vfprintf (
    FILE *restrict stream,
    const char *restrict format,
```

va_list arg)

Вывести текст в поток в соответствии с форматной строкой и используя *va_list* вместо переменного количества аргументов.

Аргументы	
<i>stream</i>	Поток вывода.
<i>format</i>	Форматная строка.
<i>arg</i>	Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.42. **vfscanf()**

```
int vfscanf (  
    FILE *restrict stream,  
    const char *restrict format,  
    va_list arg )
```

Считать информацию из потока в соответствии с форматной строкой и используя *va_list* вместо переменного количества аргументов.

Аргументы	
<i>stream</i>	Поток.
<i>format</i>	Форматная строка.
...	Аргументы для форматной строки.

Возвращает

Кол-во считанных элементов или EOF при ошибке.

19.67.4.43. **vprintf()**

```
int vprintf (  
    const char *restrict format,  
    va_list arg )
```

Вывести текст в `stdout` в соответствии с форматной строкой и используя *va_list* вместо переменного количества аргументов.

Аргументы

format Форматная строка.

arg Аргументы для форматной строки.

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.44. vscanf()

```
int vscanf (
    const char *restrict format,
    va_list arg )
```

Считать информацию из stdin в соответствии с форматной строкой и используя va_list вместо переменного количества аргументов.

Аргументы

format Форматная строка.

... Аргументы для форматной строки.

Возвращает

Кол-во считанных элементов или EOF при ошибке.

19.67.4.45. vsnprintf()

```
int vsnprintf (
    char *restrict s,
    size_t n,
    const char *restrict format,
    va_list arg )
```

Вывести текст в буфер в соответствии с форматной строкой и используя va_list вместо переменного количества аргументов.

Аргументы

s Буфер вывода.

n Максимальное количество символов, которое можно вывести в буфер вывода (включая нуль-терминатор).

format Форматная строка.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>arg</i>	Аргументы для форматной строки.
------------	---------------------------------

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.46. vsprintf()

```
int vsprintf (  
    char *restrict s,  
    const char *restrict format,  
    va_list arg )
```

Вывести текст в буфер в соответствии с форматной строкой и используя *va_list* вместо переменного количества аргументов.

Аргументы

<i>s</i>	Буфер вывода.
----------	---------------

<i>format</i>	Форматная строка.
---------------	-------------------

<i>arg</i>	Аргументы для форматной строки.
------------	---------------------------------

Возвращает

Кол-во фактически напечатанных символов, не включая нуль-терминатор.

19.67.4.47. vsscanf()

```
int vsscanf (  
    const char *restrict s,  
    const char *restrict format,  
    va_list arg )
```

Считать информацию из текстового буфера в соответствии с форматной строкой и используя *va_list* вместо переменного количества аргументов.

Аргументы

<i>s</i>	Текстовый буфер.
----------	------------------

<i>format</i>	Форматная строка.
---------------	-------------------

...	Аргументы для форматной строки.
-----	---------------------------------

Возвращает

Кол-во считанных элементов или EOF при ошибке.

19.67.5. Переменные

19.67.5.1. stderr

*FILE** stderr [extern]

19.67.5.2. stdin

*FILE** stdin [extern]

Стандартный поток ввода.

19.67.5.3. stdout

*FILE** stdout [extern]

19.68. Файл `stdlib.h`

Стандартная библиотека.

Структуры данных

- struct `div_t`
Результат деления с остатком.
- struct `ldiv_t`
*Результат деления чисел типа **long long** с остатком.*

Генерация псевдо-случайных чисел

- int `rand` (void)
- #define `RAND_MAX` (`INT_MAX`)
Максимально возможное значение, возвращаемое функцией `rand()`.
- void `srand` (unsigned int seed)

Коммуникация с окружением

- void `_Exit` (int status)
- void `abort` (void)
- int `atexit` (void(*func)(void))
- void `exit` (int status)
- #define `EXIT_FAILURE` (1)
Значение нештатного завершения для функции `exit`.
- #define `EXIT_SUCCESS` (0)
Значение успешного завершения для функции `exit`.
- `jmp_buf * exitbuf` (void)
- int `exitcode` (void)
- char * `getenv` (const char *name)
- int `setexit` (`jmp_buf` env)
- int `system` (const char *string)

Преобразования чисел

- double `atof` (const char *nptr)
- int `atoi` (const char *nptr)
- long `atol` (const char *nptr)
- double `strtod` (const char *restrict nptr, char **restrict endptr)
- float `strtof` (const char *restrict nptr, char **restrict endptr)
- long `strtol` (const char *nptr, char **endptr, int base)
- long double `strtold` (const char *restrict nptr, char **restrict endptr)
- long long `strtoll` (const char *nptr, char **endptr, int base)
- unsigned long `strtoul` (const char *nptr, char **endptr, int base)
- unsigned long long `strtoull` (const char *nptr, char **endptr, int base)

Поиск и сортировка

- void * `bsearch` (const void *key, const void *base, size_t nmem, size_t size, int(*compar)(const void *keyPtr, const void *arrayElementPtr))
- void `qsort` (void *base, size_t nmem, size_t size, int(*compar)(const void *, const void *))

Целочисленная арифметика

- int `abs` (int i)
- `div_t div` (int numer, int denom)
- long int `labs` (long int i)
- `ldiv_t ldiv` (long int numer, long int denom)
- long long int `llabs` (long long int i)

Нестандартные дополнительные функции

- int *bcd2d* (int d)
- bool *compareStr* (char *S1, char *S2)
- int *d2bcd* (int d)
- char * *gcvt* (double value, int ndig, char *buf)
- int *getWord* (size_t n, const char *src, char *dst)
- int *isdigitex* (char c, int base)
- void *itoa* (int N, char *strptr)
- void *lltoa* (long long N, char *strptr)
- int *setenv* (const char *name, const char *val)
- void *uitoa* (unsigned int N, char *strptr)
- void *ulltoa* (unsigned long long N, char *strptr)

19.68.1. Подробное описание

См. также

C11 standard 7.22.

19.68.2. Макросы

19.68.2.1. EXIT_FAILURE

```
#define EXIT_FAILURE (1)
```

Значение успешного завершения для функции *exit*.

19.68.2.2. EXIT_SUCCESS

```
#define EXIT_SUCCESS (0)
```

Значение успешного завершения для функции *exit*.

19.68.2.3. RAND_MAX

```
#define RAND_MAX (INT_MAX)
```

Максимально возможное значение, возвращаемое функцией *rand()*.

19.68.3. Функции

19.68.3.1. _Exit()

```
void _Exit (  
    int status )
```

Вызывать завершение текущего процесса без подчистки дескрипторов и потоков.

Аргументы

status Статус завершения процесса.

19.68.3.2. abort()

```
void abort (
    void )
```

Вызвать нештатное завершение текущего процесса.

19.68.3.3. abs()

```
int abs (
    int i )
```

Вычислить абсолютное значение целого.

Аргументы

i Целое.

Возвращает

Абсолютное значение целого.

19.68.3.4. atexit()

```
int atexit (
    void*(void) func )
```

Установить функцию, которая должна будет быть вызвана в случае завершения процесса.

Аргументы

func Указатель на функцию.

Возвращает

0 при успешной регистрации функции, иное при ошибке.

Отличия от стандартной реализации: функции будут вызваны даже при нештатном завершении процесса.

19.68.3.5. atof()

```
double atof (
    const char * nptr )
```

Преобразовать начальную часть строки в double-представление.

Аргументы

nptr Указатель на строку.

Возвращает

Преобразованное значение.

19.68.3.6. atoi()

int atoi (
 const char * *nptr*)

Преобразовать начальную часть строки в int-представление.

Аргументы

nptr Указатель на строку.

Возвращает

Преобразованное значение.

19.68.3.7. atol()

long atol (
 const char * *nptr*)

Преобразовать начальную часть строки в long-представление.

Аргументы

nptr Указатель на строку.

Возвращает

Преобразованное значение.

19.68.3.8. bcd2d()

int bcd2d (
 int *d*)

Перевести число из формата BCD в обычный формат.

Аргументы

d Число в формате BCD.

Возвращает

Число в обычном формате.



BCD - binary-coded decimal, двоично-десятичный код.

19.68.3.9. bsearch()

```
void* bsearch (
    const void * key,
    const void * base,
    size_t nmemb,
    size_t size,
    int (*)(const void *keyPtr, const void *arrayElementPtr) compar )
```

Выполнить бинарный поиск по отсортированному массиву.

Аргументы

key Ключ поиска.

base Указатель на первый элемент массива.

nmemb Количество элементов в массиве.

size Размер одиночного элемента в массиве.

compar Функция, сравнивающая ключ поиска с элементами массива.

Функция из аргумента *compar* должна возвращать значение <0, если элемент меньше ключа, ==0, если он равен ключу, и >0, если он больше ключа.

19.68.3.10. compareStr()

```
bool compareStr (
    char * S1,
    char * S2 )
```

Инвертированный strcmp.

Аргументы

S1,S2 Строки для сравнения.

Возвращает

true, если строки равны, false иначе.

19.68.3.11. d2bcd()

```
int d2bcd (  
    int d )
```

Перевести число из обычного формата в формат BCD.

Аргументы

d Число в обычном формате.

Возвращает

Число в формате BCD.



B CD - binary-coded decimal, двоично-десятичный код.

19.68.3.12. div()

```
div_t div (  
    int numer,  
    int denom )
```

Деление целых чисел с остатком.

Аргументы

numer Делимое.

denom Делитель.

Возвращает

Структура из частного и остатка.

19.68.3.13. exit()

```
void exit (  
    int status )
```

Вызывать завершение текущего процесса.

Аргументы

status Статус завершения процесса.

Отличия от стандартной реализации: потоки и дескрипторы, открытые процессом, не будут закрыты/обновлены.

19.68.3.14. exitbuf()

```
jmp_buf* exitbuf (  
    void )
```

Получить текущий буфер возврата для задачи.

Возвращает

Текущий буфер возврата для задачи

19.68.3.15. exitcode()

```
int exitcode (  
    void )
```

Получить код завершения задачи.

Возвращает

Текущий код завершения задачи

19.68.3.16. gcvt()

```
char* gcvt (  
    double value,  
    int ndig,  
    char * buf)
```

Конвертация числа с плавающей точкой в строку заданной длины.

Аргументы

value Конвертируемый параметр.

ndig Количество значащих цифр, которое должно быть сохранено.

buf Буфер для сохранения результата.

Возвращает

Указатель на строку с результатом.

19.68.3.17. `getenv()`

```
char* getenv (  
    const char * name )
```

Получить переменную окружения по её имени.

Аргументы

name Имя переменной окружения.

Возвращает

Указатель на переменную окружения или *NULL*, если переменная не была найдена.

19.68.3.18. `getWord()`

```
int getWord (  
    size_t n,  
    const char * src,  
    char * dst )
```

Выделение слова с номером N из строки Src и помещение его в строку Dst.

Аргументы

n Номер слова, начиная с 1.

src Исходная строка.

dst Буфер под слово.

Возвращает

0 при ошибке, 1, если слово было успешно записано в буфер, 2, если слово было успешно записано в буфер и оно изначально было обрамлено кавычками.

19.68.3.19. `isdigitex()`

```
int isdigitex (  
    char c,  
    int base )
```

Проверить, является ли символ цифрой в заданной системе счисления.

Аргументы

c Символ.

base Разрядность системы счисления (от 1 до 36).

Возвращает

0, если не является, не-0 иначе.

19.68.3.20. itoa()

```
void itoa (  
    int N,  
    char * strptr )
```

Распечатать целое число в строковый буфер.

Аргументы

N Целое.

strptr Буфер.

19.68.3.21. labs()

```
long int labs (  
    long int i )
```

Вычислить абсолютное значение целого.

Аргументы

i Целое.

Возвращает

Абсолютное значение целого.

19.68.3.22. ldiv()

```
ldiv_t ldiv (  
    long int numer,  
    long int denom )
```

Деление целых чисел **long long** с остатком.

Аргументы

numer Делимое.

denom Делитель.

Возвращает

Структура из частного и остатка.

19.68.3.23. llabs()

```
long long int llabs (  
    long long int i )
```

Вычислить абсолютное значение целого.

Аргументы

i Целое.

Возвращает

Абсолютное значение целого.

19.68.3.24. ltoa()

```
void ltoa (  
    long long N,  
    char * strptr )
```

Распечатать целое число в строковый буфер.

Аргументы

N Целое.

strptr Буфер.

19.68.3.25. qsort()

```
void qsort (  
    void * base,  
    size_t nmemb,  
    size_t size,  
    int(*)(const void *, const void *) compar )
```

Отсортировать элементы массива.

Аргументы	
<i>base</i>	Указатель на первый элемент массива.
<i>nmemb</i>	Количество элементов в массиве.
<i>size</i>	Размер одиночного элемента в массиве.
<i>compar</i>	Функция сравнения.

Функция из аргумента *compar* должна возвращать значение <0 , если первый аргумент меньше второго, $=0$, если первый аргумент равен второму, и >0 , если первый аргумент больше второго.

19.68.3.26. rand()

```
int rand (
    void )
```

Получить псевдослучайное число.

Возвращает

Псевдослучайное число в промежутке $[0, \text{RAND_MAX}]$.

19.68.3.27. setenv()

```
int setenv (
    const char * name,
    const char * val )
```

Установить переменную окружения.

Аргументы	
<i>name</i>	Имя переменной окружения.
<i>val</i>	Значение переменной окружения.

Возвращает

0 при успехе, не-0 иначе.

19.68.3.28. setexit()

```
int setexit (
    jmp_buf env )
```

Установить точку сохранения контекста для завершения задачи.

Аргументы

env Буфер для сохранения контекста.

Возвращает

0x80000000, если функция установила точку, иное значение, если возврат функции был вызван вызовом *exit()*.

19.68.3.29. srand()

```
void srand (
    unsigned int seed )
```

Установить 'зерно' для псевдослучайной последовательности.

Аргументы

seed 'Зерно' для псевдослучайной последовательности

19.68.3.30. strtod()

```
double strtod (
    const char *restrict nptr,
    char **restrict endptr )
```

Преобразовать начальную часть строки в double-представление.

Аргументы

nptr Указатель на строку.

out *endptr* Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.

Возвращает

Преобразованное значение.

19.68.3.31. strttof()

```
float strttof (
    const char *restrict nptr,
    char **restrict endptr )
```

Преобразовать начальную часть строки в float-представление.

Аргументы

	<i>nptr</i>	Указатель на строку.
--	-------------	----------------------

out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
-----	---------------	---

Возвращает

Преобразованное значение.

19.68.3.32. strtol()

long strtol (

```
const char * nptr;  
char ** endptr;  
int base )
```

Преобразовать начальную часть строки в long-представление.

Аргументы

	<i>nptr</i>	Указатель на строку.
--	-------------	----------------------

out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
-----	---------------	---

	<i>base</i>	Основание системы исчисления.
--	-------------	-------------------------------

Возвращает

Преобразованное значение.

19.68.3.33. strtolf()

long double strtolf (

```
const char *restrict nptr;  
char **restrict endptr )
```

Преобразовать начальную часть строки в long-double-представление.

Аргументы

	<i>nptr</i>	Указатель на строку.
--	-------------	----------------------

out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
-----	---------------	---

Возвращает

Преобразованное значение.

19.68.3.34. strtoll()

```
long long strtoll (
    const char * nptr,
    char ** endptr,
    int base )
```

Преобразовать начальную часть строки в long-long-представление.

Аргументы		
	<i>nptr</i>	Указатель на строку.
out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
	<i>base</i>	Основание системы исчисления.

Возвращает

Преобразованное значение.

19.68.3.35. strtoul()

```
unsigned long strtoul (
    const char * nptr,
    char ** endptr,
    int base )
```

Преобразовать начальную часть строки в unsigned-long-представление.

Аргументы		
	<i>nptr</i>	Указатель на строку.
out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
	<i>base</i>	Основание системы исчисления.

Возвращает

Преобразованное значение.

19.68.3.36. strtoull()

```
unsigned long long strtoull (  
    const char * nptr,  
    char ** endptr,  
    int base )
```

Преобразовать начальную часть строки в unsigned-long-long-представление.

Аргументы		
	<i>nptr</i>	Указатель на строку.
out	<i>endptr</i>	Опциональный буфер для возврата указателя на символ в строке, следующий после преобразованной строки.
	<i>base</i>	Основание системы исчисления.

Возвращает

Преобразованное значение.

19.68.3.37. system()

```
int system (  
    const char * string )
```

Выполнить команду системе.

Аргументы	
<i>string</i>	Команда системе или <i>NULL</i> .

Возвращает

Если *string* - *NULL*, то не-0, если исполнение команд доступно и 0 иначе. Если *string* - не *NULL*, то функция возвращает возврат команды.



РЕАЛИЗАЦИЯ ФУНКЦИИ ОТСУТСТВУЕТ.

19.68.3.38. uitoa()

```
void uitoa (  
    unsigned int N,  
    char * strptr )
```

Распечатать целое число в строковый буфер.

Аргументы	
-----------	--

<i>N</i>	Целое.
----------	--------

<i>strptr</i>	Буфер.
---------------	--------

19.68.3.39. ulltoa()

```
void ulltoa (  
    unsigned long long N,  
    char * strptr )
```

Распечатать целое число в строковый буфер.

Аргументы	
-----------	--

<i>N</i>	Целое.
----------	--------

<i>strptr</i>	Буфер.
---------------	--------

19.69. Файл `stdnoreturn.h`

Определение макроса `noreturn`.

Макросы

- `#define noreturn`
Не поддерживается в режимах, отличных от C11.

19.69.1. Подробное описание

См. также

[C11 standard 7.23.](#)

19.69.2. Макросы

19.69.2.1. `noreturn`

```
#define noreturn
```

19.70. Файл `string.h`

Работа с массивами символов.

Функции

- `size_t str8len` (const char *s)

Нестандартные функции для работы с массивами символов (расширения POSIX и т.п.)

- `#define bzero(addr, size) memset(addr, 0, size);`
- char `lowercase` (unsigned char c)
- `size_t merge` (char **b, const char *s)
- char * `stpcpy` (char *restrict s1, const char *restrict s2)
- `#define strcmpi(x, y) stricmp(x, y)`
- char * `strdup` (const char *s)
- int `stricmp` (const char *s1, const char *s2)
- char * `stristr` (const char *s1, const char *s2)
- `size_t strlcat` (char *dst, const char *src, size_t dsize)
- `size_t strlcpy` (char *dst, const char *src, size_t dsize)
- char * `strlwr` (char *s)
- `size_t strlen` (const char *s, size_t maxlen)
- char * `strsep` (char **s, const char *ct)
- char * `strtok_r` (char *s, const char *delim, char **lasts)
- char * `strup` (char *s)
- char `uppercase` (unsigned char c)

Стандартные функции для работы с блоками памяти

- void * `memchr` (const void *s, int c, size_t n)
 - int `memcmp` (const void *s1, const void *s2, size_t n)
 - void * `memcpy` (void *restrict s1, const void *restrict s2, size_t n)
 - void * `memmove` (void *s1, const void *s2, size_t n)
 - void * `memset` (void *addr, int c, size_t size)
- Поиск первого вхождения значения в области памяти.*
- void * `memset` (void *s, int c, size_t n)

Стандартные функции для работы с массивами символов

- char * `strcat` (char *restrict s1, const char *restrict s2)
- char * `strchr` (const char *s, int c)
- int `strcmp` (const char *s1, const char *s2)
- int `strcoll` (const char *s1, const char *s2)
- char * `strcpy` (char *restrict s1, const char *restrict s2)
- `size_t strcspn` (const char *s1, const char *s2)
- char * `strerror` (int errnum)
- `size_t strlen` (const char *s)
- char * `strncat` (char *restrict s1, const char *restrict s2, size_t n)
- int `strncmp` (const char *s1, const char *s2, size_t n)
- char * `strncpy` (char *restrict s1, const char *restrict s2, size_t n)
- char * `strpbrk` (const char *s1, const char *s2)
- char * `strrchr` (const char *s, int c)
- `size_t strspn` (const char *s1, const char *s2)
- char * `strstr` (const char *s1, const char *s2)
- char * `strtok` (char *restrict s1, const char *restrict s2)
- `size_t strxfrm` (char *restrict s1, const char *restrict s2, size_t n)

Функции для работы с `char16_t` нуль-терминированными строками

- `char16_t * str16chr` (const `char16_t *str`, `char16_t chr`)
- `int str16cmp` (const `char16_t *s1`, const `char16_t *s2`)
- `char16_t * str16dup` (const `char16_t *str`)
- `size_t str16cat` (`char16_t *dst`, const `char16_t *src`, `size_t dsize`)
- `size_t str16cpy` (`char16_t *dst`, const `char16_t *src`, `size_t dsize`)
- `size_t str16len` (const `char16_t *str`)

Преобразования чисел в строки и обратно

- void `charToHex` (unsigned char D, char *S)
Преобразование 8-разрядного числа в строку в 16-тиричном виде.
- unsigned int `hexToInt` (const char *S)
*Преобразование строки, содержащей 32-разрядное беззнаковое число в 16-тиричном виде, в значение **unsigned int**.*
- void `intToHex` (int D, char *S)
Преобразование 32-разрядного числа в строку в 16-тиричном виде.
- void `intToHexUniversal` (int D, char *S, `size_t len`, char spacer, const char hexSetToUse[static 16])
Преобразование 32-разрядного числа в строку в 16-тиричном виде с форматированием.
- void `longlongToHex` (long long D, char *S)
Преобразование 64-разрядного числа в строку в 16-тиричном виде.
- void `shortToHex` (unsigned short D, char *S)
Преобразование 16-разрядного числа в строку в 16-тиричном виде.

19.70.1. Подробное описание

См. также

[C11 standard 7.24.](#)

19.70.2. Макросы

19.70.2.1. `bzero`

```
#define bzero(  
    addr,  
    size ) memset (addr, 0, size);
```

Альтернатива `memset` из BCD.

19.70.2.2. `strcmpi`

```
#define strcmpi(  
    x,  
    y ) stricmp (x, y)
```

Псевдоним для `stricmp`.

19.70.3. Функции

19.70.3.1. charToHex()

```
void charToHex (  
    unsigned char D,  
    char * S)
```

Функция преобразует число в строку в 16-тиричном представлении. Для строки должно быть отведено достаточное для преобразования место в памяти.

Аргументы

D Исходное число.

S Указатель на строку, в которую будет помещено число в текстовом виде.

19.70.3.2. hexToInt()

```
unsigned int hexToInt (  
    const char * S)
```

Функция преобразует строку в число. Знак игнорируется. Количество символов не должно превышать 8. Принимаются цифры от **0** до **9** и латинские буквы от **a** до **f** и от **A** до **F**.

Аргументы

S Исходная строка.

Возвращает

Результат преобразования.

19.70.3.3. intToHex()

```
void intToHex (  
    int D,  
    char * S)
```

Функция преобразует число в строку в 16-тиричном представлении. Для строки должно быть отведено достаточное для преобразования место в памяти.

Аргументы

D Исходное число.

S Указатель на строку, в которую будет помещено число в текстовом виде.

19.70.3.4. intToHexUniversal()

```
void intToHexUniversal (
    int D,
    char * S,
    size_t len,
    char spacer,
    const char hexSetToUse[static 16])
```

Функция преобразует число в строку в 16-тиричном представлении. В отличие от *intToHex()* функция предоставляет дополнительные возможности по форматированию итоговой строки. Для строки должно быть отведено достаточное для преобразования место в памяти.

Аргументы	
<i>D</i>	Исходное число.
<i>S</i>	Указатель на строку, в которую будет помещено число в текстовом виде.
<i>len</i>	Длина итоговой строки. Если длина значащей части меньше — строка будет дополнена символами spacer .
<i>spacer</i>	Символ, дополняющий строку спереди до указанной длины len .
<i>hexSetToUse</i>	Набор символов для отображения чисел. В функции <i>intToHex()</i> используется набор, содержащий заглавные латинские буквы, такое поведение можно изменить, задав набор символов с прописными буквами.

19.70.3.5. longlongToHex()

```
void longlongToHex (
    long long D,
    char * S)
```

Функция преобразует число в строку в 16-тиричном представлении. Для строки должно быть отведено достаточное для преобразования место в памяти.

Аргументы	
<i>D</i>	Исходное число.
<i>S</i>	Указатель на строку, в которую будет помещено число в текстовом виде.

19.70.3.6. lowercase()

```
char lowercase (
    unsigned char c)
```

Псевдоним для *tolower*.

Аргументы

c Символ для преобразования.

Возвращает

Преобразованный символ.

19.70.3.7. memchr()

```
void* memchr (  
    const void * s,  
    int c,  
    size_t n )
```

Найти первое вхождение символа *c* (приведенного к `unsigned char`) в первых *n* байт объекта по указателю *s*.

Аргументы

s Указатель на объект, в котором проводится поиск.

c Искомый символ.

n Количество байт для поиска.

Возвращает

Указатель на найденный символ в объекте или *NULL*, если символ не был найден.

19.70.3.8. memcmp()

```
int memcmp (  
    const void * s1,  
    const void * s2,  
    size_t n )
```

Сравнить первые *n* байт объекта по указателю *s1* с байтами объекта по указателю *s2*.

Аргументы

s1,s2 Указатели на объекты для сравнения.

n Количество байт для сравнения.

Возвращает

0, если объекты равны, >0, если *s1* > *s2*, <0, если *s2* < *s1*.

19.70.3.9. memcpy()

```
void* memcpy (
    void *restrict s1,
    const void *restrict s2,
    size_t n )
```

Скопировать *n* байт из объекта по указателю *s1* в объект по указателю *s1*.

Аргументы

<i>s1</i>	Указатель на объект, в который производится копирование.
<i>s2</i>	Указатель на объект, из которого производится копирование.
<i>n</i>	Количество байт для копирования.

Возвращает

Значение *s1*.



Объекты не должны пересекаться. В случае пересечения объектов следует использовать функцию `memmove`.

19.70.3.10. memmove()

```
void* memmove (
    void * s1,
    const void * s2,
    size_t n )
```

Скопировать *n* байт из объекта по указателю *s1* в объект по указателю *s1*. Объекты могут пересекаться.

Аргументы

<i>s1</i>	Указатель на объект, в который производится копирование.
<i>s2</i>	Указатель на объект, из которого производится копирование.
<i>n</i>	Количество байт для копирования.

Возвращает

Значение *s1*.

19.70.3.11. memscan()

```
void* memscan (  
    void * addr,  
    int c,  
    size_t size )
```

Функция возвращает адрес первого вхождения *c* или на 1 байт дальше области памяти, если *c* не найден.

Аргументы

addr Начальный адрес.

c Искомое значение.

size Размер области памяти.

Возвращает

void* Адрес – указатель на найденный символ.

19.70.3.12. memset()

```
void* memset (  
    void * s,  
    int c,  
    size_t n )
```

Скопировать символ *c* (приведенный к `unsigned char`) в каждый из первых *n* байтов объекта по указателю *s*.

Аргументы

s Указатель на заполняемый объект.

c Символ для копирования.

n Количество байт для заполнения.

Возвращает
Значение *s*.

19.70.3.13. merge()

```
size_t merge (  
    char ** b,  
    const char * s )
```

Перераспределить строку из динамической памяти, присоединив к ней другую строку.

Аргументы

- | | |
|----------|--|
| <i>b</i> | Указатель на переменную <code>char*</code> с перераспределяемым указателем на исходную строку. |
| <i>s</i> | Копируемая строка. |

Возвращает
Длина новой строки без учета нуль-терминатора.

19.70.3.14. shortToHex()

```
void shortToHex (  
    unsigned short D,  
    char * S )
```

Функция преобразует число в строку в 16-тиричном представлении. Для строки должно быть отведено достаточное для преобразования место в памяти.

Аргументы

- | | |
|----------|---|
| <i>D</i> | Исходное число. |
| <i>S</i> | Указатель на строку, в которую будет помещено число в текстовом виде. |

19.70.3.15. stpcpy()

```
char* stpcpy (  
    char *restrict s1,  
    const char *restrict s2 )
```

Скопировать строку по указателю *s2* в буфер по указателю *s1*.

Аргументы

- s1* Указатель на массив, в который будет производится копирование.
- s2* Копируемая строка.

Возвращает

Конец строки *s1* (т.е. адрес нуль-терминатора).



Объекты не должны пересекаться.

19.70.3.16. str16chr()

```
char16_t* str16chr (  
    const char16_t * str,  
    char16_t chr )
```

strchr для char16_t-строк.

Аргументы

- str* Указатель на строку, в которой проводится поиск.
- chr* Искомый символ.

Возвращает

Указатель на первый найденный символ в строке или *NULL*, если символ не был найден.

19.70.3.17. str16cmp()

```
int str16cmp (  
    const char16_t * s1,  
    const char16_t * s2 )
```

strcmp для char16_t-строк.

Аргументы

- s1,s2* Указатели на строки для сравнения.

Возвращает

0, если строки равны, >0, если $s1 > s2$, <0, если $s2 < s1$.

19.70.3.18. str16dup()

```
char16_t* str16dup (  
    const char16_t * str )
```

strdup для char16_t-строк.

Аргументы

str Указатель на строку для копирования.

Возвращает

Копия строки в динамической памяти.

19.70.3.19. str16lcat()

```
size_t str16lcat (  
    char16_t * dst,  
    const char16_t * src,  
    size_t dsize )
```

strlcat для char16_t-строк.

Аргументы

dst Указатель на массив, в который будет производиться копирование.

src Копируемая строка.

dsize Размер буфера *dst* в char16_t символах.

Возвращает

Размер копируемой строки + MIN(*dsize*, размер начальной строки). Если возврат $\geq dsize$, то строка не была скопирована полностью.

19.70.3.20. str16lcpy()

```
size_t str16lcpy (  
    char16_t * dst,  
    const char16_t * src,  
    size_t dsize )
```


strcpy для char16_t-строк.

Аргументы

<i>dst</i>	Указатель на массив, в который будет производиться копирование.
<i>src</i>	Копируемая строка.
<i>dsize</i>	Размер буфера <i>dst</i> в char16_t символах.

Возвращает

Размер копируемой строки. Если возврат \geq *dsize*, то строка не была скопирована полностью.

19.70.3.21. str16len()

```
size_t str16len (  
    const char16_t * str )
```

strlen для char16_t-строк.

Аргументы

<i>str</i>	Указатель на строку, длина которой должна быть посчитана.
------------	---

Возвращает

Количество символов в строке, без учета нуль-терминатора.

19.70.3.22. str8len()

```
size_t str8len (  
    const char * s )
```

Расчитать длину UTF-8 строки по указателю *s*.

Аргументы

<i>s</i>	Указатель на строку, длина которой должна быть посчитана.
----------	---

Возвращает

Количество символов в строке, без учета нуль-терминатора.

19.70.3.23. strcat()

```
char* strcat (  
    char *restrict s1,  
    const char *restrict s2 )
```

Присоединить копию строки по указателю s2 (включая нулевой символ) в конец строки по указателю s1.

Аргументы

s1 Дополняемая строка.

s2 Копируемая строка.

Возвращает

Значение s1.



Объекты не должны пересекаться.

19.70.3.24. strchr()

```
char* strchr (  
    const char * s,  
    int c )
```

Найти первое вхождение символа c в строке по указателю s.

Аргументы

s Указатель на строку, в которой проводится поиск.

c Искомый символ.

Возвращает

Указатель на первый найденный символ в строке или *NULL*, если символ не был найден.

19.70.3.25. strcmp()

```
int strcmp (  
    const char * s1,  
    const char * s2 )
```

Сравнить строку по указателю s1 со строкой по указателю s2.

Аргументы

s1,s2 Указатели на строки для сравнения.

Возвращает

0 – если строки равны, >0 – если $s1 > s2$, <0 – если $s1 < s2$.

19.70.3.26. strcoll()

```
int strcoll (  
    const char * s1,  
    const char * s2 )
```

Сравнить строку по указателю *s1* со строкой по указателю *s2*, с учетом текущей локали.

Аргументы

s1,s2 Указатели на строки для сравнения.

Возвращает

0, если строки равны, >0, если $s1 > s2$, <0, если $s2 < s1$.



РЕАЛИЗАЦИЯ ФУНКЦИИ ОТСУТСТВУЕТ.

19.70.3.27. strcpy()

```
char* strcpy (  
    char *restrict s1,  
    const char *restrict s2 )
```

Скопировать строку по указателю *s2* в буфер по указателю *s1*.

Аргументы

s1 Указатель на массив, в который будет производится копирование.

s2 Копируемая строка.

Возвращает

Значение *s1*.



Объекты не должны пересекаться.

19.70.3.28. `strcspn()`

```
size_t strcspn (  
    const char * s1,  
    const char * s2 )
```

Найти первое вхождение в строку по указателю *s1* любого из символов из строки по указателю *s2*.

Аргументы

s1 Указатель на строку, в которой выполняется поиск.

s2 Указатель на строку, содержащую символы для поиска.

Возвращает

Количество символов до первого вхождения.

19.70.3.29. `strdup()`

```
char* strdup (  
    const char * s )
```

Сделать копию строки в динамической памяти.

Аргументы

s Указатель на строку для копирования.

Возвращает

Копия строки в динамической памяти.

19.70.3.30. `strerror()`

```
char* strerror (  
    int errnum )
```

Получить сообщение об ошибке, соответствующее коду ошибки.

Аргументы

errno Код ошибки.

Возвращает

Указатель на строку с кодом ошибки.



Текущая реализация тривиальна.

19.70.3.31. stricmp()

```
int stricmp (
    const char * s1,
    const char * s2 )
```

Сравнить строку по указателю *s1* со строкой по указателю *s2* без различия между заглавными и строчными символами.

Аргументы

s1,s2 Указатели на строки для сравнения.

Возвращает

0, если строки равны, >0, если *s1* > *s2*, <0, если *s2* < *s1*.

19.70.3.32. strstr()

```
char* strstr (
    const char * s1,
    const char * s2 )
```

Найти первое вхождение подстроки по указателю *s2* в строке по указателю *s1* без различия между заглавными и строчными символами.

Аргументы

s1 Указатель на строку, в которой выполняется поиск.

s2 Указатель на строку, содержащую подстроку для поиска.

Возвращает

Указатель на первое вхождение подстроки в строке по указателю *s1* или *NULL*, при его отсутствии.

19.70.3.33. `strlcat()`

```
size_t strlcat (  
    char * dst,  
    const char * src,  
    size_t dsize )
```

Безопасный аналог `strcat`. По-сути `strncat`, который ВСЕГДА ставит нуль-терминатор (кроме строки 0го размера).

Аргументы

<i>dst</i>	Указатель на массив, в который будет производится копирование.
<i>src</i>	Копируемая строка.
<i>dsize</i>	Размер буфера <i>dst</i> .

Возвращает

Размер копируемой строки + `MIN(dsize, размер начальной строки)`. Если возврат `>= dsize`, то строка не была скопирована полностью.

19.70.3.34. `strlcpy()`

```
size_t strlcpy (  
    char * dst,  
    const char * src,  
    size_t dsize )
```

Безопасный аналог `strcpy`. По-сути `strncpy` без заполнения нулями, который ВСЕГДА ставит нуль-терминатор (кроме строки 0го размера).

Аргументы

<i>dst</i>	Указатель на массив, в который будет производится копирование.
<i>src</i>	Копируемая строка.
<i>dsize</i>	Размер буфера <i>dst</i> .

Возвращает

Размер копируемой строки. Если возврат `>= dsize`, то строка не была скопирована полностью.

19.70.3.35. strlen()

```
size_t strlen (  
    const char * s )
```

Расчитать длину строки по указателю *s*.

Аргументы

s Указатель на строку, длина которой должна быть посчитана.

Возвращает

Количество символов в строке, без учета нуль-терминатора.

19.70.3.36. strlwr()

```
char* strlwr (  
    char * s )
```

Преобразовать все подходящие символы строки по указателю *s* в строчные.

Аргументы

s Указатель на преобразуемую строку.

Возвращает

Значение *s*.

19.70.3.37. strncat()

```
char* strncat (  
    char *restrict s1,  
    const char *restrict s2,  
    size_t n )
```

Присоединить не более *n* символов из строки по указателю *s2* (включая нулевой символ) в конец строки по указателю *s1*.

Аргументы

s1 Дополняемая строка.

s2 Копируемая строка.

n Количество символов для копирования.

Возвращает

Значение $s1$.



Объекты не должны пересекаться.

19.70.3.38. `strncmp()`

```
int strncmp (  
    const char *  $s1$ ,  
    const char *  $s2$ ,  
    size_t  $n$  )
```

Сравнить не более n символов из строки по указателю $s1$ с символами из строки по указателю $s2$.

Аргументы

$s1, s2$ Указатели на строки для сравнения.

n Количество символов для сравнения.

Возвращает

0, если строки равны, >0 , если $s1 > s2$, <0 , если $s1 < s2$.

19.70.3.39. `strncpy()`

```
char* strncpy (  
    char *restrict  $s1$ ,  
    const char *restrict  $s2$ ,  
    size_t  $n$  )
```

Скопировать не более чем n символов из строки по указателю $s2$ в буфер по указателю $s1$.

Аргументы

$s1$ Указатель на массив, в который будет производится копирование.

$s2$ Копируемая строка.

n Количество символов для копирования.

Возвращает

Значение $s1$.

В случае, если размер копируемой строки меньше n , то остаток байт в буфере-массиве будет заполнен нулевыми байтами.



Объекты не должны пересекаться.

19.70.3.40. `strnlen()`

```
size_t strnlen (  
    const char * s,  
    size_t maxlen )
```

Расчитать длину строки по указателю *s*, но не более чем *maxlen*.

Аргументы

s Указатель на строку, длина которой должна быть посчитана.

maxlen Максимальное количество символов для проверки.

Возвращает

Количество символов в строке, без учета нуль-терминатора, или *maxlen*.

19.70.3.41. `strpbrk()`

```
char* strpbrk (  
    const char * s1,  
    const char * s2 )
```

Найти первое вхождение в строку по указателю *s1* любого из символов из строки по указателю *s2*.

Аргументы

s1 Указатель на строку, в которой выполняется поиск.

s2 Указатель на строку, содержащую символы для поиска.

Возвращает

Указатель на первое вхождение в строке по указателю *s1*, или *NULL*, при его отсутствии.

19.70.3.42. `strrchr()`

```
char* strrchr (  
    const char * s,  
    int c )
```

Найти последнее вхождение символа *c* в строке по указателю *s*.

Аргументы

s Указатель на строку, в которой проводится поиск.

c Искомый символ.

Возвращает

Указатель на последний найденный символ в строке или *NULL*, если символ не был найден.

19.70.3.43. strsep()

```
char* strsep (  
    char ** s,  
    const char * ct )
```

Извлечь элемент из строки и скорректировать указатель на строку.

Аргументы

s Указатель на переменную *char** с указателем на исходную строку.

ct Указатель на строку, содержащую разделители для поиска.

Возвращает

Исходное значение *s*.

19.70.3.44. strspn()

```
size_t strspn (  
    const char * s1,  
    const char * s2 )
```

Найти длину начального участка строки по указателю *s1*, который состоит только из символов из строки по указателю *s2*.

Аргументы

s1 Указатель на строку, в которой проводится поиск.

s2 Указатель на строку, содержащую символы для поиска.

Возвращает

Длина подходящего под условия начального участка строки.

19.70.3.45. strstr()

```
char* strstr (
    const char * s1,
    const char * s2 )
```

Найти первое вхождение подстроки по указателю *s2* в строке по указателю *s1*.

Аргументы

s1 Указатель на строку, в которой выполняется поиск.

s2 Указатель на строку, содержащую подстроку для поиска.

Возвращает

Указатель на первое вхождение подстроки в строке по указателю *s1* или *NULL*, при его отсутствии.

19.70.3.46. strtok()

```
char* strtok (
    char *restrict s1,
    const char *restrict s2 )
```

Найти лексемы в строке по указателю *s1*, разделенные разделителями из строки по указателю *s2*.

Аргументы

s1 Указатель на строку, в которой выполняется поиск. Эта строка будет изменена.

s2 Указатель на строку, содержащую разделители для поиска.

Возвращает

Указатель на последнюю найденную лексему в строке или *NULL*, при её отсутствии.

19.70.3.47. strtok_r()

```
char* strtok_r (
    char * s,
    const char * delim,
    char ** lasts )
```

Реентерабельная версия `strtok`.

Аргументы

<i>s</i>	Указатель на строку, в которой выполняется поиск. Эта строка будет изменена.
<i>delim</i>	Указатель на строку, содержащую разделители для поиска.
<i>lasts</i>	Указатель на переменную <code>char*</code> , которая используется для учета контекста.

Возвращает

Указатель на последнюю найденную лексему в строке или *NULL*, при её отсутствии.

19.70.3.48.strup()

```
char* strup (  
    char * s )
```

Преобразовать все подходящие символы строки по указателю *s* в заглавные.

Аргументы

<i>s</i>	Указатель на преобразуемую строку.
----------	------------------------------------

Возвращает

Значение *s*.

19.70.3.49.strxfrm()

```
size_t strxfrm (  
    char *restrict s1,  
    const char *restrict s2,  
    size_t n )
```

Преобразовать строку с учетом локали.

Преобразовать строку по указателю *s2* таким образом, чтобы ее можно было использовать функцией *strcmp()*, и поместить результат преобразования в строку по указателю *s2*. После преобразования результат вызова функции *strcmp()*, использующей параметр *s1*, будет совпадать с результатом вызова функции *strcoll()*, использующей исходную строку, на которую указывает параметр *s2*. В массив, адресуемый параметром *s1*, записывается не более *n* символов.

Аргументы

<i>s1</i>	Указатель на массив, в который будет производится преобразование.
<i>s2</i>	Преобразуемая строка.
<i>n</i>	Количество символов для преобразования.

Возвращает

Длина преобразованной строки без нулевого символа.



РЕАЛИЗАЦИЯ ФУНКЦИИ ОТСУТСТВУЕТ.

19.70.3.50. `upcase()`

```
char upcase (  
    unsigned char c )
```

Псевдоним для `toupper`.

Аргументы

`c` Символ для преобразования.

Возвращает

Преобразованный символ.

19.71. Файл tasklib.h

Управление задачами.

Структуры данных

- struct *exit_st*
- struct *TCB*
Структура блока управления задачей.

Макросы

- #define *MX_FP_TASK* 0x0008
- #define *TASK_DELAY* 0x02
- #define *TASK_INT_HANDLING* 0x10
- #define *TASK_MARKER* 0x5AA78627
Маркер задачи.
- #define *TASK_PEND* 0x01
- #define *TASK_PS_STACK_SIZE* 32
Размер служебного стека задачи.
- #define *TASK_SUSPEND* 0x04
- #define *TASK_WDT* 0x08
- #define *VX_FP_TASK* *MX_FP_TASK*

Определения типов

- typedef struct *exit_st* *exit_proc*
- typedef struct *TCB* *taskId*
Структура блока управления задачей.

Перечисления

- enum *TASK_SEM_EXIT* { *tseTimeoutOrNone* = 0 , *tseTakenFromAnotherTask* , *tseFlushed* , *tseMutexOrCounterError* }

Функции

- int *cpuUsage* ()
Оценка загрузки процессора в процентах.
- *STATUS deleteWorkTask* (void)
Удалить текущую задачу.
- void *kernelInit* (void(*archInit)(void), *FUNCPTR* rootRtn, size_t rootMemSize, size_t pMemPoolStart, size_t pMemPoolEnd)
Инициализация многозадачного ядра.
- *STATUS kernelTimeSlice* (int ticks)
Задать режим планировщика ядра.
- *STATUS printTasksInfo* (void)
- *STATUS suspendWorkTask* (void)
Приостановить выполнение текущей задачи.
- *taskId* * *taskCreate* (int task, int stacksz, int param)
Создать блок управления задачей.
- *STATUS taskDelay* (int ticks)
Отложить выполнение задачи.
- int *taskDelete* (*TASK_ID* TID)
Удалить задачу.
- *STATUS taskDeleteForce* (*TASK_ID* TID)
Принудительное удаление задачи.
- *TASK_ID taskIdSelf* (void)

- *Идентификатор текущей задачи.*
`STATUS taskIdVerify (TASK_ID TID)`
- *Проверить идентификатор задачи.*
`STATUS taskLock (void)`
- *Запрет переключения текущей задачи.*
`const char * taskName (TASK_ID TID)`
- *Получить имя задачи.*
`TASK_ID taskNameToId (const char *name)`
- *Поиск задачи по имени.*
`int taskPriority (void)`
- *Получить приоритет задачи.*
`STATUS taskPriorityGet (TASK_ID TID, int *pPriority)`
- *Получить приоритет задачи.*
`STATUS taskPrioritySet (TASK_ID TID, int newPriority)`
- *Установить приоритет задачи.*
`STATUS taskResume (TASK_ID TID)`
- *Возобновить выполнение задачи.*
`STATUS taskSafe (void)`
- *Защита задачи.*
`TASK_ID taskSpawn (const char *name, int priority, int options, int stackSize, FUNCPTR entryPt, int arg)`
- *Создать новую задачу.*
`STATUS taskSuspend (TASK_ID TID)`
- *Приостановить выполнение задачи.*
`void taskSwitch (void)`
- *Принудительно переключить текущую задачу на другую активную и более приоритетную.*
`TASK_ID taskTcb (TASK_ID TID)`
- `STATUS taskUnlock (void)`
- *Отмена запрета переключения.*
`STATUS taskUnsafe (void)`
- *Снять защиту задачи.*
`void tickAnnounce (void)`
- *Обработчик прерывания системного таймера.*

19.71.1. Подробное описание

Приведенные в данном файле функции пользователь *MULTEX-ARM* может применять для создания, удаления, приостановки, возобновления и задержки задач в многозадачной среде.

См. также

Многозадачность и межзадачное взаимодействие.

Пример запуска простой задачи:

```
bool stop;

// Задача в отдельной функции
int taskSimple (int dummy) {
    // Инициализация переменных задачи (если есть)
    void *p = malloc (size);

    while (!stop) {
        // ... тело задачи
        taskDelay (1);
    }

    // Освобождение выделенной памяти
```

```
    free (p);  
    // Завершение задачи  
    return 0;  
}  
  
void testSimple () {  
    stop = false;  
    taskSpawn ("{task simple}{}", 10, 0, 0, taskSimple, 0);  
}
```

19.71.2. Макросы

19.71.2.1. MX_FP_TASK

```
#define MX_FP_TASK 0x0008
```

Флаг создания задачи, укзывающий что в задаче используется математический сопроцессор (вещественные числа).

19.71.2.2. TASK_DELAY

```
#define TASK_DELAY 0x02
```

Задача отложена.

19.71.2.3. TASK_INT_HANDLING

```
#define TASK_INT_HANDLING 0x10
```

Задача обрабатывает прерывание.

19.71.2.4. TASK_MARKER

```
#define TASK_MARKER 0x5AA78627
```

19.71.2.5. TASK_PEND

```
#define TASK_PEND 0x01
```

Задача ожидает разблокировки семафора.

19.71.2.6. TASK_PS_STACK_SIZE

```
#define TASK_PS_STACK_SIZE 32
```

Размер служебного стека задачи.

19.71.2.7. TASK_SUSPEND

```
#define TASK_SUSPEND 0x04
```

Задача приостановлена.

19.71.2.8. TASK_WDT

```
#define TASK_WDT 0x08
```

Задача является таймером-вочдогом.

19.71.2.9. VX_FP_TASK

```
#define VX_FP_TASK MX_FP_TASK
```

Алиас MX_FP_TASK для обратной совместимости.

19.71.3. Типы

19.71.3.1. exit_proc

```
typedef struct exit_st exit_proc
```

Структура элемента списка функций-обработчиков, вызываемых при завершении задачи при помощи *exit()* и *abort()*.

19.71.3.2. taskId

```
typedef struct TCB taskId
```

19.71.4. Перечисления

19.71.4.1. TASK_SEM_EXIT

```
enum TASK_SEM_EXIT
```

Как именно задача была отпущена с семафора.

Элементы перечислений

tseTimeoutOrNone	Задача не предпринимала никаких действий с семафорами или семафор не был взят в желаемое время. <i>semTake</i> возвращает <i>ERROR</i> .
tseTakenFromAnotherTask	Семафор был отдан освободившей его задачей. <i>semTake</i> возвращает <i>OK</i> .
tseFlushed	Бинарный семафор был отпущен, но не захвачен. <i>semTake</i> возвращает <i>OK</i> .
tseMutexOrCounterError	Семафор-счетчик или мьютекс был отпущен, но не захвачен. <i>semTake</i> возвращает <i>ERROR</i> .

```
00082 {
00083     tseTimeoutOrNone = 0,
00084     tseTakenFromAnotherTask,
```

```
00085     tseFlushed,  
00086     tseMutexOrCounterError,  
00087 }TASK_SEM_EXIT;
```

19.71.5. Функции

19.71.5.1. cpuUsage()

```
int cpuUsage ( )
```

Функция оценивает значение счётчика простоя процессора и сравнивает его со значением полученным на ненагруженном процессоре.

Возвращает

Процент загрузки процессора: 0 – процессор не нагружен, 100 – процессор загружен полностью.

19.71.5.2. deleteWorkTask()

```
STATUS deleteWorkTask (  
    void )
```

Функция удаляет текущую задачу. Эквивалентно вызову taskDelete(NULL).

Возвращает

Значение *OK* при успешном выполнении, или *ERROR*, если задача защищена от удаления функцией *taskSafe()*.

19.71.5.3. kernelInit()

```
void kernelInit (  
    void*(*)(void) archInit,  
    FUNCPTR rootRtn,  
    size_t rootMemSize,  
    size_t pMemPoolStart,  
    size_t pMemPoolEnd )
```

Функция инициализирует многозадачное ядро *MULTEX-ARM*, создает и запускает служебные системные задачи и задачу пользователя.



Эта функция не завершается до выключения системы и не возвращает управления вызывающей процедуре! Пользовательские процедуры не должны производить ее повторный вызов.

Аргументы	
<i>archInit</i>	Функция инициализации конфигурации аппаратной части.
<i>rootRtn</i>	Корневая задача – задача пользователя или командный интерпретатор Shell .
<i>rootMemSize</i>	Размер стека, отводимого для корневой задачи.
<i>pMemPoolStart</i>	Адрес начала свободной оперативной памяти, выделяемой под HEAP-область .
<i>pMemPoolEnd</i>	Адрес конца HEAP-области .

19.71.5.4. kernelTimeSlice()

STATUS kernelTimeSlice (
int ticks)

Функция задает режим планировщика задач ядра *MULTEX-ARM*.

Аргументы	
<i>ticks</i>	Квант времени в тиках таймера, выделяемый каждой задаче в режиме карусельного планирования. При этом через интервал времени ticks планировщик автоматически переключит задачу на другую активную задачу, приоритет которой выше или равен приоритету текущей задачи. Для задания режима приоритетного планирования следует указать значение параметра ticks равным нулю. При этом текущая задача будет выполняться до тех пор, пока не активизируется более высокоприоритетная задача, или текущая задача не будет приостановлена по собственной инициативе.

Возвращает

Всегда *OK*.

19.71.5.5. printTasksInfo()

STATUS printTasksInfo (
void)

Распечатать в stdout таблицу с информацией о задачах.

Возвращает

Всегда возвращает *OK*.

19.71.5.6. suspendWorkTask()

STATUS suspendWorkTask (

```
void )
```

Функция приостанавливает выполнение указанной задачи. Эквивалентно вызову `taskSuspend(NULL)`

Возвращает

Значение *OK* при успешном выполнении, или *ERROR*, если задача защищена от приостановки функцией *taskSafe()*.

19.71.5.7. taskCreate()

```
taskId* taskCreate (  
    int task,  
    int stacksz,  
    int param )
```

Функция выделяет память и подготавливает структуры для дальнейшего использования.

Аргументы

<i>task</i>	Приведенный к <code>int</code> указатель на функцию, являющуюся точкой входа.
<i>stacksz</i>	Размер стека для задачи.
<i>param</i>	Параметр, передаваемый в точку входа.

Возвращает

Указатель на блок управления задачей.

19.71.5.8. taskDelay()

```
STATUS taskDelay (  
    int ticks )
```

Функция откладывает выполнение текущей задачи на заданный интервал времени.

Аргументы

<i>ticks</i>	Число тиков системного таймера, на которое следует отложить выполнение текущей задачи. Для задания временных интервалов, не связанных с частотой системного таймера, воспользуйтесь функцией <i>sysClkRateGet()</i> , которая вернет количество тиков таймера за одну секунду. Если частота системного таймера не менялась, то это 1000.
--------------	--

Возвращает

STATUS - при вызове в контексте пользовательских задач функция всегда возвращает значение *OK*.

19.71.5.9. taskDelete()

```
int taskDelete (  
    TASK_ID TID )
```

Функция удаляет задачу из системы вне зависимости от ее состояния.

Аргументы

TID Идентификатор удаляемой задачи или *NULL* для удаления текущей задачи.

Возвращает

0 в случае успешного удаления.

Отрицательное число в случае неудачи. Неудача может возникнуть в случае, если идентификатор не является указателем на правильный *TCB* задачи, или если задача защищена от удаления функцией *taskSafe()*.

19.71.5.10. taskDeleteForce()

```
STATUS taskDeleteForce (  
    TASK_ID TID )
```

Удаляет указанную задачу, даже при установленной защите от удаления.

Аргументы

TID Идентификатор удаляемой задачи.

Возвращает

Функция возвращает значение *OK* в случае успешного завершения, или *ERROR* в случае неудачи. Неудача может возникнуть в случае, если идентификатор не является указателем на правильный *TCB* задачи.

19.71.5.11. taskIdSelf()

```
TASK_ID taskIdSelf (  
    void )
```

Функция возвращает идентификатор текущей задачи.

Возвращает

Идентификатор текущей выполняемой задачи.

19.71.5.12. `taskIdVerify()`

```
STATUS taskIdVerify (  
    TASK_ID TID )
```

Функция проверяет правильность указанного идентификатора.

Аргументы

TID Идентификатор задачи.

Возвращает

OK – если идентификатор верен.

ERROR – если нет.

19.71.5.13. `taskLock()`

```
STATUS taskLock (  
    void )
```

Функция запрещает переключение контекста с текущей выполняемой задачи на другую.

Возвращает

Всегда *OK*.

19.71.5.14. `taskName()`

```
const char* taskName (  
    TASK_ID TID )
```

Возвращает символьное имя указанной задачи.

Аргументы

TID Идентификатор задачи.

Возвращает

Указатель на строку символов имени задачи, заканчивающуюся нулем. Если имя не найдено, функция возвращает **NULL**.

19.71.5.15. `taskNameToId()`

```
TASK_ID taskNameToId (  
    const char * name )
```

По заданному символическому имени задачи функция отыскивает ее идентификатор.

Аргументы

<i>name</i>	Указатель на строку с именем задачи.
-------------	--------------------------------------

Возвращает

Идентификатор задачи при успешном выполнении или **NULL** в случае неудачи.

19.71.5.16. `taskPriority()`

```
int taskPriority (  
    void )
```

Возвращает приоритет текущей задачи.

Возвращает

Значение приоритета текущей задачи в диапазоне **0-255**.

- **Важно!** Меньшие значения соответствуют более высокому приоритету!

19.71.5.17. `taskPriorityGet()`

```
STATUS taskPriorityGet (  
    TASK_ID TID,  
    int * pPriority )
```

Функция возвращает приоритет указанной задачи.

Аргументы

<i>TID</i>	Идентификатор задачи, у которой запрашивается приоритет.
<i>pPriority</i>	Возвращаемое значение приоритета.

Возвращает

Значение *OK* в случае успешного завершения, или *ERROR* в случае неудачи. Неудача может возникнуть в случае, если идентификатор не является указателем на правильный *TCB* задачи.

19.71.5.18. taskPrioritySet()

```
STATUS taskPrioritySet (  
    TASK_ID TID,  
    int newPriority )
```

Устанавливает заданный приоритет для указанной задачи.

Аргументы

<i>TID</i>	Идентификатор задачи, которой требуется сменить приоритет.
<i>newPriority</i>	Требуемое значение приоритета.

Возвращает

Функция возвращает значение *OK* в случае успешного завершения, или *ERROR* в случае неудачи. Неудача может возникнуть в случае, если идентификатор не является указателем на правильный *TCB* задачи.

- **Важно!** Изменение приоритета задачи может привести к немедленному переключению на нее.

19.71.5.19. taskResume()

```
STATUS taskResume (  
    TASK_ID TID )
```

Возобновляет выполнение приостановленной задачи.

Аргументы

<i>TID</i>	Идентификатор приостановленной задачи.
------------	--

Возвращает

Значение *OK* при успешном выполнении, или *ERROR* при неудаче (неверный идентификатор задачи или задача не находится в приостановленном состоянии).

19.71.5.20. taskSafe()

```
STATUS taskSafe (  
    void )
```


Функция защищает текущую задачу от случайного удаления.

Возвращает

OK при успешном выполнении.

19.71.5.21. taskSpawn()

```
TASK_ID taskSpawn (
    const char * name,
    int priority,
    int options,
    int stackSize,
    FUNCPTR entryPt,
    int arg )
```

Функция создает и активизирует новую задачу в среде *MULTEX-ARM*. Если создаваемая задача имеет более высокий приоритет, чем та задача, в контексте которой производится вызов этой функции, то производится немедленное переключение на эту задачу. В противном случае выполнение задачи, вызвавшей эту функцию, продолжается.

Аргументы

<i>name</i>	Указатель на имя задачи. Имя задачи – произвольная последовательность символов, заканчивающаяся нулем. Длина имени не ограничена, однако не рекомендуется использовать имена длиннее 10 символов. После вызова функции строка может быть удалена из памяти.
<i>priority</i>	Приоритет задачи. Нулевое значение соответствует самому высокому приоритету. Чем больше значение, тем ниже приоритет. Для совместимости с vxWorks рекомендуется выбирать значения в диапазоне 0 – 255.
<i>options</i>	Опции создаваемой задачи. Для обычной задачи следует задавать значение 0. Если в задаче используются операции с вещественными числами и таких задач несколько, то следует задавать опцию <i>MX_FP_TASK</i> (или <i>VX_FP_TASK</i> , принятую в vxWorks и сохраненную для совместимости). При этом, при каждом переключении с этой задачи на другую, ядро <i>MULTEX-ARM</i> будет сохранять контекст сопроцессора в специально отводимой для этой задачи области. При переключении на эту задачу контекст сопроцессора будет автоматически восстановлен. Однако, следует иметь ввиду, что время переключения задач при установке такой опции несколько увеличивается.
<i>stackSize</i>	<p>Размер стекового кадра для создаваемой задачи. При выборе размера стека следует учитывать глубину вложений всех процедур, вызываемых в контексте задачи. Прерывания в <i>MULTEX-ARM</i> также выполняются с использованием стека прерываемой задачи, поэтому следует рассчитывать наихудший вариант при возможных вложенных прерываниях. Не рекомендуется выбирать размер стека менее 1000, типичный размер 10000 байт. При задании большого размера следует учитывать общий объем оперативной памяти: так, при создании 100 задач с объемом стека в 20000 байт для каждой, только под стек будет отведено около 2Мбайт памяти.</p> <ul style="list-style-type: none"> • Важно! Если указан размер стекового кадра, равный 0, то под него будет выделен размер такой-же, как у задачи, из которой вызывалась эта функция.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>entryPt</i>	Указатель на точку входа в задачу. Задачей в <i>MULTEX-ARM</i> может быть любая функция типа <i>int Func(int,...)</i> . При этом, выход из тела функции (return) приведет к завершению этой задачи и ее автоматическому удалению из памяти.
<i>arg</i>	Аргумент, передаваемый функции, запускаемой ядром <i>MULTEX-ARM</i> как задача.

Возвращает

Идентификатор создаваемой задачи (указатель на **TCB** - блок управления задачи). Если функция не смогла создать задачу (например, система не может выделить требуемый объем памяти), то возвращаемое значение равно **0**.

19.71.5.22. taskSuspend()

```
STATUS taskSuspend (  
    TASK_ID TID )
```

Функция приостанавливает выполнение указанной задачи.

Аргументы

<i>TID</i>	Идентификатор приостанавливаемой задачи или <i>NULL</i> для приостановки текущей задачи. Для возобновления выполнения приостановленной задачи воспользуйтесь функцией <i>taskResume()</i> .
------------	---

Возвращает

Значение *OK* при успешном выполнении, или *ERROR* при неудаче (неверный идентификатор задачи или защита от приостановки функцией *taskSafe()*).

19.71.5.23. taskSwitch()

```
void taskSwitch (  
    void )
```

Текущая задача не будет отложена или задержана. В случае отсутствия более приоритетных активных задач текущая задача продолжит свое выполнение.

19.71.5.24. taskTcb()

```
TASK_ID taskTcb (  
    TASK_ID TID )
```

Проверить корректность идентификатора задачи.

Аргументы

TID Проверяемый идентификатор задачи.

Возвращает

TID, если идентификатор корректный, или NULL, если не корректный.

19.71.5.25. taskUnlock()

STATUS taskUnlock (
void)

Отменяет действие функции *taskLock()*.

Возвращает

Всегда *OK*.

19.71.5.26. taskUnsafe()

STATUS taskUnsafe (
void)

Снимает установленную функцией *taskISafe()* защиту задачи от случайного удаления.

Возвращает

OK при успешном выполнении.

19.71.5.27. tickAnnounce()

void tickAnnounce (
void)

Вызов функции обработчика прерывания от системного таймера. При подключении пользовательской процедуры обработки прерывания от системного таймера для сохранения работоспособности ядра *MULTEX-ARM* в тело этой функции требуется включить вызов *tickAnnounce()*. Это необходимо, так как общая синхронизация ядра производится системным таймером. *

19.72. Файл `terminator.h`

Обработка корректного закрытия драйверов для горячей перезагрузки ОС.

Функции

- void *doTerminate* (void)
- void *registerTerminator* (void(*proc)(void))

19.72.1. Функции

19.72.1.1. `doTerminate()`

```
void doTerminate (  
    void )
```

Вызвать все ранее зарегистрированные процедуры завершения.

19.72.1.2. `registerTerminator()`

```
void registerTerminator (  
    void(*) (void) proc )
```

Зарегистрировать процедуру завершения.

Аргументы

<i>proc</i>	Указатель на функцию завершения, которая должна быть вызвана при вызове <i>doTerminate()</i> .
-------------	--

19.73. Файл time.h

Стандартные манипуляции с датой и временем.

Структуры данных

- struct *timespec*
Тики процессора.
- struct *tm*

Макросы

- #define *CLOCKS_PER_SEC* (1000000)

Временные базисы.

```
{
    • char * asctime (const struct tm *timeptr)
    • char * asctime_r (const struct tm *restrict timeptr, char buf[static 26])
    • clock_t clock (void)
    • typedef size_t clock_t
      Календарное время. Секунды с 1 января 1970 года времени UTC.
    • char * ctime (const time_t *timer)
    • double difftime (time_t time1, time_t time0)
    • struct tm * gmtime (const time_t *timer)
    • struct tm * gmtime_r (const time_t *restrict timer, struct tm *restrict result)
    • struct tm * localtime (const time_t *timer)
    • struct tm * localtime_r (const time_t *restrict timer, struct tm *restrict result)
    • time_t mktime (struct tm *timeptr)
    • size_t strftime (char *restrict s, size_t maxsize, const char *restrict format, const struct tm *restrict timeptr)
    • time_t time (time_t *timer)
    • typedef size_t time_t
      }@
    • #define TIME_UTC (1)
    • int timespec_get (struct timespec *ts, int base)
```

19.73.1. Подробное описание

См. стандарт C11 7.27.

См. также

[C11 standard 7.27.](#)

19.73.2. Макросы

19.73.2.1. CLOCKS_PER_SEC

```
#define CLOCKS_PER_SEC (1000000)
```

19.73.2.2. TIME_UTC

```
#define TIME_UTC (1)
```

19.73.3. Типы

19.73.3.1. `clock_t`

```
typedef size_t clock_t
```

19.73.3.2. `time_t`

```
typedef size_t time_t
```

19.73.4. Функции

19.73.4.1. `asctime()`

```
char* asctime (  
    const struct tm * timeptr )
```

Преобразовать время в календарном представлении в строку.

Аргументы

timeptr Указатель время в календарном представлении.

Возвращает

Указатель на статически выделенную строку с результатом или *NULL* при ошибке.

19.73.4.2. `asctime_r()`

```
char* asctime_r (  
    const struct tm *restrict timeptr,  
    char buf[static 26] )
```

Преобразовать время в календарном представлении в строку и записать ее в предоставленный буфер.

Аргументы

timeptr Указатель время в календарном представлении.

buf Буфер длиной минимум 26 байт под итоговую строку.

Возвращает

Указатель на итоговую строку с результатом или *NULL* при ошибке.

19.73.4.3. clock()

```
clock_t clock (  
    void )
```

Получить суммарное процессорное время, использованное программой.

Возвращает

Суммарное процессорное время, использованное программой.

19.73.4.4. ctime()

```
char* ctime (  
    const time_t * timer )
```

Преобразовать календарное время в строку.

Аргументы

timer Указатель на календарное время.

Возвращает

Указатель на статически выделенную строку с результатом или *NULL* при ошибке.

19.73.4.5. difftime()

```
double difftime (  
    time_t time1,  
    time_t time0 )
```

Получить кол-во секунд, прошедших со времени *time0* и до времени *time1*.

Аргументы

time1 Конечная временная точка.

time0 Начальная временная точка.

Возвращает

Количество секунд разницы.

19.73.4.6. gmtime()

```
struct tm* gmtime (  
    const time_t * timer )
```

Преобразовать календарное время в локальное представление времени.

Аргументы

timer Указатель на календарное время.

Возвращает

Указатель на статически выделенную структуру или NULL при ошибке.

19.73.4.7. gmtime_r()

```
struct tm* gmtime_r (  
    const time_t *restrict timer,  
    struct tm *restrict result )
```

Преобразовать календарное время в календарное представление времени и записать его в предоставленный буфер.

Аргументы

timer Указатель на календарное время.

result Буфер для записи.

Возвращает

Указатель на буфер для записи или NULL при ошибке.

19.73.4.8. localtime()

```
struct tm* localtime (  
    const time_t * timer )
```

Преобразовать календарное время в локальное время в календарном представлении.

Аргументы

timer Указатель на календарное время.

Возвращает

Указатель на статически выделенную структуру или NULL при ошибке.

19.73.4.9. localtime_r()

```
struct tm* localtime_r (  
    const time_t *restrict timer,  
    struct tm *restrict result )
```

Преобразовать календарное время в локальное время в календарном представлении и записать его в предоставленный буфер.

Аргументы

timer Указатель на календарное время.

result Буфер для записи.

Возвращает

Указатель на буфер для записи или NULL при ошибке.

19.73.4.10. mktime()

```
time_t mktime (  
    struct tm * timeptr )
```

Преобразовать время в календарном представлении в абсолютное время.

Аргументы

timeptr Время в календарном представлении.

Возвращает

Календарное время для представленного времени или -1 при ошибке.

19.73.4.11. strftime()

```
size_t strftime (  

```

```
char *restrict s,  
size_t maxsize,  
const char *restrict format,  
const struct tm *restrict timeptr )
```

Преобразовать календарное время в строку в соответствии с форматом.

Аргументы	
<i>s</i>	Буфер для записи.
<i>maxsize</i>	Размер буфера для записи.
<i>format</i>	Формат даты и времени.
<i>timeptr</i>	Дата-время.

Возвращает

Кол-во символов в массиве *s*, не считая нуль-терминатор, при возврате 0 содержимое массива не определено.



Фактически всегда распечатывает результат в формате `asctime`.

19.73.4.12. `time()`

```
time_t time (  
time_t * timer )
```

Вернуть время в секундах, прошедшее с начала эпохи.

Аргументы	
<i>timer</i>	Если не-NULL, то также сохранить результат в буфере.

Возвращает

Кол-во секунд с начала эпохи.

19.73.4.13. `timespec_get()`

```
int timespec_get (  
struct timespec * ts,  
int base )
```

Установить значение интервала времени в соответствии с временным базисом.

Аргументы

ts Указатель на заполняемый интервал.

base Временной базис.

Возвращает

Значение *base* при успехе, 0 при ошибке.

19.74. Файл timer-arm.h

Управление аппаратными таймерами **ARM** процессоров.

Функции

- int *taCount* ()
Количество доступных таймеров.
- *STATUS taSetInterrupt* (unsigned int timer, unsigned int group, unsigned int priority, *usr_int_proc* handler)
Задать обработчик прерывания для таймера.
- *STATUS taStart* (unsigned int timer, unsigned int period_ns, *bool* continuously)
Запуск таймера.
- *STATUS taStop* (unsigned int timer)
Остановка таймера.
- int *taSystemNumber* ()
Номер системного таймера.

Макросы номеров аппаратных таймеров

Синонимы номеров таймеров возможно использовать в качестве параметров функций **timer** для улучшения читаемости кода. Таймер 1 не указан, так как он задействован под нужды системы.

- #define *TIMER_0* 0
Создание блокирующих задержек idelay()
- #define *TIMER_1* 1
Системный таймер
- #define *TIMER_2* 2
- #define *TIMER_3* 3
- #define *TIMER_4* 4
- #define *TIMER_5* 5
- #define *TIMER_SYS* (*TIMER_1*)
Синоним для системного таймера.

19.74.1. Подробное описание

Процессоры **ARM** архитектуры содержат несколько аппаратных таймеров. Один из них ВСЕГДА задействован под нужды **MULTEX-ARM**. Остальные могут быть использованы в пользовательском ПО для задания коротких (менее 1мс) временных интервалов. Так данный модуль нацелен на создание именно коротких временных интервалов, то настройка всех таймеров выполняется соответствующим образом — прочие экзотические способы настройки таймеров не используются. Минимальный временной интервал **42мкс**. Большие временные (от 1мс и выше) интервалы рекомендуется создавать, используя задержку выполнения текущей задачи *taskDelay()*.

19.74.2. Макросы

19.74.2.1. TIMER_0

```
#define TIMER_0 0
```

Аппаратный таймер 0.



Не использовать в пользовательских задачах!

19.74.2.2. TIMER_1

```
#define TIMER_1 1
```

Аппаратный таймер 1.



Не использовать в пользовательских задачах!

19.74.2.3. TIMER_2

```
#define TIMER_2 2
```

Аппаратный таймер 2.

19.74.2.4. TIMER_3

```
#define TIMER_3 3
```

Аппаратный таймер 3.

19.74.2.5. TIMER_4

```
#define TIMER_4 4
```

Аппаратный таймер 4.

19.74.2.6. TIMER_5

```
#define TIMER_5 5
```

Аппаратный таймер 5.

19.74.2.7. TIMER_SYS

```
#define TIMER_SYS (TIMER_1)
```

19.74.3. Функции

19.74.3.1. taCount()

```
int taCount ( )
```

Функция возвращает количество доступных аппаратных таймеров, включая системный.

Возвращает

Количество аппаратных таймеров, либо *ERROR*, если инициализация модуля не выполнена.

19.74.3.2. taSetInterrupt()

```
STATUS taSetInterrupt (
    unsigned int timer,
    unsigned int group,
    unsigned int priority,
    usr_int_proc handler )
```

Функция задаёт обработчик прерывания для указанного таймера.

Аргументы	
<i>timer</i>	Номер таймера в пределах от 0 до <i>taCount()</i> . Таймер с номером <i>taSystemNumber()</i> не может быть использован.
<i>group</i>	Группа прерываний.
<i>priority</i>	Приоритет прерывания в группе.
<i>handler</i>	Обработчик прерывания, назначаемый выбранному таймеру.

Возвращает

OK, если прерывание задано, иначе *ERROR*.

См. также

Порядок обработки прерываний в зависимости от группы и приоритета в разделе *Управление прерываниями*.

19.74.3.3. taStart()

```
STATUS taStart (
    unsigned int timer,
    unsigned int period_ns,
    bool continuously )
```

Таймер может быть запущен как в непрерывном режиме так и в режиме одиночного отсчёта. По истечении заданного интервала времени каждый раз будет формироваться аппаратное прерывание. Обработчик прерывания следует задать с помощью

Аргументы	
<i>timer</i>	Номер таймера в пределах от 0 до <i>taCount()</i> . Таймер с номером <i>taSystemNumber()</i> не может быть запущен из пользовательской задачи.
<i>period_ns</i>	Период работы таймера, либо интервал для одиночного срабатывания. Задаётся в наносекундах.
<i>continuously</i>	<i>true</i> — непрерывный режим работы таймера, иначе — одиночное срабатывание.

Возвращает

OK, если таймер успешно запущен, иначе *ERROR*.

19.74.3.4. taStop()

STATUS taStop (
 unsigned int *timer*)

Остановка таймера, отключение прерываний.

Аргументы

<i>timer</i>	Номер таймера в пределах от 0 до <i>taCount()</i> . Таймер с номером <i>taSystemNumber()</i> не может быть остановлен из пользовательской задачи.
--------------	---

Возвращает

OK, если таймер успешно остановлен, иначе *ERROR*.

19.74.3.5. taSystemNumber()

int taSystemNumber ()

Функция возвращает номер аппаратного таймера, отведённого под нужды операционной системы.

Возвращает

Номер системного таймера, либо *ERROR*, если инициализация модуля не выполнена.

19.75. Файл timer.h

Управление системным таймером.

Функции

- void *hard_reset* ()
Аппаратная перезагрузка.
- int *sysClkRateGet* ()
Количество тиков таймера в секунду.
- int *sysClkRateSet* (int ticks)
Задать частоту системного таймера.
- unsigned long *tickGet* ()
Число тиков таймера с момента включения.
- void *tickSet* (unsigned long val)
Задать значение системного таймера.

19.75.1. Подробное описание

См. также

Подробнее о системном таймере см. в главе *Системный таймер*.

19.75.2. Функции

19.75.2.1. hard_reset()

```
void hard_reset ( )
```

Перезагрузка процессора с помощью аппаратного watchdog таймера через 0,5с.

19.75.2.2. sysClkRateGet()

```
int sysClkRateGet ( )
```

Функция возвращает количество тиков системного таймера за одну секунду. Функция не производит физических замеров частоты таймера, а лишь опрашивает внутреннюю переменную, поэтому выполняется быстро.

Возвращает

Количество тиков системного таймера за одну секунду.

19.75.2.3. sysClkRateSet()

```
int sysClkRateSet (  
    int ticks )
```

Функция устанавливает заданную частоту системного таймера.

Аргументы

ticks Частота в Герцах, на которую будет перестроен таймер.

Возвращает

Функция возвращает **OK**.

19.75.2.4. tickGet()

unsigned long tickGet ()

Функция возвращает число тиков таймера с момента включения системы до момента ее вызова. Счетчик тиков помещается во внутреннюю 32-битную переменную и может быть изменен пользовательской задачей с помощью вызова функции *tickSet()*. В этом случае возвращаемое функцией значение будет определяться интервалом между записью счетчика и ее вызовом, а также записываемой величиной.

Возвращает

Количество тиков таймера.

19.75.2.5. tickSet()

void tickSet (
 unsigned long *val*)

Функция устанавливает значение счетчика тиков системного таймера.

Аргументы

val Записываемое в счетчик значение.

19.76. Файл tv-decoder.h

Драйвер аппаратного TV-декодера.

Перечисления

- enum `tvd_chnl_t` { `TVD_0` = 0 , `TVD_1` , `TVD_2` , `TVD_3` }
Номера каналов (аппаратных входов) TV-декодера.
- enum `tvd_fmt_t` { `TVD_PL_YUV422` , `TVD_PL_YUV420` , `TVD_MB_YUV420` }
Выбор формата выходных данных TV-декодера.
- enum `tvd_interface_t` { `TVD_CVBS` , `TVD_YPBPR_I` , `TVD_YPBPR_P` }
Выбор входного интерфейса TV-декодера.
- enum `tvd_system_t` { `TVD_NTSC` , `TVD_PAL` , `TVD_SECAM` }
Выбор стандарта видео сигнала.

Функции

- void `tvd_capture_off` (`tvd_chnl_t` id)
Остановка захвата сигнала для выбранного канала.
- void `tvd_capture_on` (`tvd_chnl_t` id)
Запуск захвата сигнала для выбранного канала.
- unsigned int `tvd_get_status` (`tvd_chnl_t` id)
Получить статус TV-декодера.
- void `tvd_init` (`tvd_interface_t` interface, `tvd_system_t` system)
Запуск работы TV-декодера.
- void `tvd_irq_all_status_clear` (int status)
Очистить биты прерывания TV-декодера.
- int `tvd_irq_all_status_get` ()
Получить статус прерывания всех каналов TV-декодера.
- void `tvd_irq_disable` (`tvd_chnl_t` id)
Деактивировать прерывания для канала TV-декодера.
- void `tvd_irq_enable` (`tvd_chnl_t` id)
Активировать прерывания для канала TV-декодера.
- void `tvd_irq_status_clear` (`tvd_chnl_t` id)
Очистить бит прерывания выбранного канала TV-декодера.
- signed int `tvd_irq_status_get` (`tvd_chnl_t` id)
Получить статус прерывания выбранного канала TV-декодера.
- void `tvd_set_chroma` (`tvd_chnl_t` id, unsigned int addr)
Задать адрес буфера для выходного буфера яркостного сигнала.
- void `tvd_set_color` (`tvd_chnl_t` id, unsigned int luma, unsigned int contrast, unsigned int saturation, unsigned int hue)
Настройка параметров передачи цвета TV-декодера.
- void `tvd_set_fmt` (`tvd_chnl_t` id, `tvd_fmt_t` fmt)
Задать формат декодированного кадра.
- void `tvd_set_height` (`tvd_chnl_t` id, unsigned int h)
Задать высоту кадра входного видео сигнала.
- void `tvd_set_luma` (`tvd_chnl_t` id, unsigned int addr)
Задать адрес буфера для выходного буфера цветоразностного сигнала.
- void `tvd_set_width` (`tvd_chnl_t` id, unsigned int w)
Задать ширину кадра входного видео сигнала.

19.76.1. Подробное описание

Подключение:

```
#include <multimedia/tv-decoder.h>
```

Makefile:

```
LIBRARIES += -l_tvd
```

Настройка аппаратной части TV-декодера. Рекомендуется использовать TV-декодер в составе Устройства приёма телевизионного сигнала, описанного в [tv-receiver.h](#).

19.76.2. Перечисления**19.76.2.1. tvd_chnl_t**

enum [tvd_chnl_t](#)

Элементы перечислений

TVD_0

TVD_1

TVD_2

TVD_3

```
00057         {
00058         \mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aa1c7dceaca29deb6bede1e0b
= 0,
00059         \mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aab14da9f2b7637dd52381705
00060         \mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aab87d8b2d9037bf8cfaeb0dd
00061         \mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aa7e6ac9a141b4fa1468f507f
00062 }
         \mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358a}{tvd_chnl_t}};
```

19.76.2.2. tvd_fmt_t

enum [tvd_fmt_t](#)

Элементы перечислений

TVD_PL_YUV422

TVD_PL_YUV420

TVD_MB_YUV420

```
00048      {
00049
00050      \mbox{\hyperlink{tv-decoder_8h_ab5bdfca6290854631e6c87695e128a08a79d12df7307179ef0d0adb90}}
00051      \mbox{\hyperlink{tv-decoder_8h_ab5bdfca6290854631e6c87695e128a08a1515d3c73d3f64dd6cb01285}}
00052 }
00053 \mbox{\hyperlink{tv-decoder_8h_ab5bdfca6290854631e6c87695e128a08}{tvd_fmt_t}};
```

19.76.2.3. tvd_interface_t

enum *tvd_interface_t*

Элементы перечислений

TVD_CVBS

TVD_YPBPR_I

TVD_YPBPR_P

```
00030      {
00031
00032      \mbox{\hyperlink{tv-decoder_8h_aa5c6c5e878cb3b4c87b4c672835bb1ecad3d13579dc3332e7c5af2ec7}}
00033      \mbox{\hyperlink{tv-decoder_8h_aa5c6c5e878cb3b4c87b4c672835bb1eca484e4be1951261ef149948c6}}
00034 }
00035 \mbox{\hyperlink{tv-decoder_8h_aa5c6c5e878cb3b4c87b4c672835bb1eca15036649536f317f7819554}}{tvd_interface_t}};
```

19.76.2.4. tvd_system_t

enum *tvd_system_t*

Элементы перечислений

TVD_NTSC

TVD_PAL

TVD_SECAM

```
00039      {
```

```
00040 \mbox{\hyperlink{tv-decoder_8h_a063bf68278699240e8088dad0e469b3caae26a5b34f7511be1dcb773e}}
00041 \mbox{\hyperlink{tv-decoder_8h_a063bf68278699240e8088dad0e469b3ca64b4bbc9cfa4b54b49b3e43b}}
00042 \mbox{\hyperlink{tv-decoder_8h_a063bf68278699240e8088dad0e469b3caa80c358bb174be2c4c3cb70b}}
00043 }
\mbox{\hyperlink{tv-decoder_8h_a063bf68278699240e8088dad0e469b3c}{tvd_system_t}};
```

19.76.3. Функции

19.76.3.1. tvd_capture_off()

```
void tvd_capture_off (
    tvd_chnl_t id )
```

Функция останавливает захват входного сигнала для выбранного канала TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления [tvd_chnl_t](#)).

19.76.3.2. tvd_capture_on()

```
void tvd_capture_on (
    tvd_chnl_t id )
```

Функция запускает захват входного сигнала для выбранного канала TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления [tvd_chnl_t](#)).

19.76.3.3. tvd_get_status()

```
unsigned int tvd_get_status (
    tvd_chnl_t id )
```

Функция возвращает статус аппаратного TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления [tvd_chnl_t](#)).

Возвращает

Битовую маску. Используются следующие биты: бит 0 – detect (признак обнаружения входного сигнала); бит 4 – system.

19.76.3.4. tvd_init()

```
void tvd_init (  
    tvd_interface_t interface,  
    tvd_system_t system )
```

Инициализация аппаратного модуля захвата телевизионного сигнала. Модуль запускается сразу на все 4 входа.

Аргументы

interface Выбор входного интерфейса декодера.

system Выбор стандарта сигнала для **CVBS** интерфейса.

19.76.3.5. tvd_irq_all_status_clear()

```
void tvd_irq_all_status_clear (  
    int status )
```

Функция позволяет очистить группу бит регистра статуса прерывания TV-декодера.

Аргументы

status Значение группы бит регистра статуса, полученное с помощью *tvd_irq_all_status_get()*.

19.76.3.6. tvd_irq_all_status_get()

```
int tvd_irq_all_status_get ( )
```

Функция позволяет получить значение группы бит регистра статуса прерывания TV-декодера.

Возвращает

Значение группы бит регистра статуса. Значение – первые 4 бита.

19.76.3.7. tvd_irq_disable()

```
void tvd_irq_disable (  
    tvd_chnl_t id )
```

Функция деактивирует прерывания для выбранного канала TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tv_d_chnl_t*).

19.76.3.8. tvd_irq_enable()

```
void tvd_irq_enable (  
    tvd_chnl_t id )
```

Функция активирует прерывания для выбранного канала TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tv_d_chnl_t*).

19.76.3.9. tvd_irq_status_clear()

```
void tvd_irq_status_clear (  
    tvd_chnl_t id )
```

Функция позволяет очистить один бит регистра статуса прерывания TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tv_d_chnl_t*).

19.76.3.10. tvd_irq_status_get()

```
signed int tvd_irq_status_get (  
    tvd_chnl_t id )
```

Функция позволяет получить значение одного бита регистра статуса прерывания TV-декодера.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tv_d_chnl_t*).

Возвращает

1 – если бит статуса выбранного канала взведён, иначе **0**.

19.76.3.11. tvd_set_chroma()

```
void tvd_set_chroma (
```

```
tvd_chnl_t id,  
unsigned int addr )
```

Функция позволяет задать адрес области памяти в которую будут помещены данные яркостного сигнала декодированного кадра.

Аргументы

<i>id</i>	Номер канала декодера (рекомендуется выбирать из перечисления <i>tvd_chnl_t</i>).
<i>addr</i>	Адрес области памяти для яркостного сигнала.

19.76.3.12. *tvd_set_color()*

```
void tvd_set_color (  
    tvd_chnl_t id,  
    unsigned int luma,  
    unsigned int contrast,  
    unsigned int saturation,  
    unsigned int hue )
```

Настройка параметров передачи цвета TV-декодера при отличии от значений по умолчанию.

Аргументы

<i>id</i>	Номер канала декодера (рекомендуется выбирать из перечисления <i>tvd_chnl_t</i>).
<i>luma</i>	
<i>contrast</i>	
<i>saturation</i>	
<i>hue</i>	

19.76.3.13. *tvd_set_fmt()*

```
void tvd_set_fmt (  
    tvd_chnl_t id,  
    tvd_fmt_t fmt )
```

Функция позволяет выбрать формат выходных данных TV-декодера.

Аргументы

<i>id</i>	Номер канала декодера (рекомендуется выбирать из перечисления <i>tvd_chnl_t</i>).
<i>fmt</i>	Формат данных из перечисления <i>tvd_fmt_t</i> .

19.76.3.14. tvd_set_height()

```
void tvd_set_height (  
    tvd_chnl_t id,  
    unsigned int h )
```

Функция задаёт высоту кадра входного видео сигнала.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tvd_chnl_t*).

w Высота кадра в пикселях.

19.76.3.15. tvd_set_luma()

```
void tvd_set_luma (  
    tvd_chnl_t id,  
    unsigned int addr )
```

Функция позволяет задать адрес области памяти в которую будут помещены данные цветоразностного сигнала декодированного кадра.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tvd_chnl_t*).

addr Адрес области памяти для цветоразностного сигнала.

19.76.3.16. tvd_set_width()

```
void tvd_set_width (  
    tvd_chnl_t id,  
    unsigned int w )
```

Функция задаёт ширину кадра входного видео сигнала.

Аргументы

id Номер канала декодера (рекомендуется выбирать из перечисления *tvd_chnl_t*).

w Ширина кадра в пикселях.

19.77. Файл tv-receiver.h

Устройство захвата телевизионного сигнала.

Функции

- `bool tvrDevCreate` (int n, `tvd_fmt_t` fmt, unsigned int width, unsigned int height)
Создание устройства захвата декодированного видео потока.
- `void tvrDevDelete` (int n)
Удаление устройства захвата декодированного видео потока.
- `bool tvrGetData` (int n, HDC surf)
Получить данные последнего декодированного кадра.
- `bool tvrInit` (`tvd_interface_t` interface, `tvd_system_t` system)
Настройка и запуск аппаратного декодера.
- `bool tvrWaitFrame` (int n, int timeout)
Ожидание захвата одного кадра в выбранном канале.

19.77.1. Подробное описание

Подключение:

```
#include <multimedia/tv-receiver.h>
```

Makefile:

```
LIBRARIES += -l_tvd
```

Устройство приёма телевизионного сигнала на базе драйвера аппаратного декодера `tv-decoder.h`. Для начала приёма сигнала необходимо запустить аппаратную часть с помощью функции инициализации и создать устройство приёма данных. При создании устройства ему выделяется тройной кольцевой буфер данные из которого можно брать в любой момент времени. Для синхронизации с обновлением данных от камеры можно использовать функцию `tvrDataChanged()`. Приём данных от всех источников входного сигнала ведётся независимо.

Пример запуска устройства приёма телевизионного сигнала, получения кадров и вывода изображения на конструируемую поверхность приведён ниже:

```
HDC surf = newSurface (720, 576, G2D_FMT_PYUV420UVC);
\mbox{\hyperlink{tv-receiver_8h_aae16e65299359919e4241489ec9ff97d}{tvrInit}}
(\mbox{\hyperlink{tv-decoder_8h_aa5c6c5e878cb3b4c87b4c672835bb1ecad3d13579dc3332e7c5af2e}
\mbox{\hyperlink{tv-decoder_8h_a063bf68278699240e8088dad0e469b3ca64b4bbc9cfa4b54b49b3e43b}
\mbox{\hyperlink{tv-receiver_8h_a927f49cee00a3d533781830222d64f72}{tvrDevCreate}}
(\mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aa1c7dceaca29deb6bede1e0b}
\mbox{\hyperlink{tv-decoder_8h_ab5bdfca6290854631e6c87695e128a08a1515d3c73d3f64dd6cb01285}
720, 576);
while (true) {

\mbox{\hyperlink{tv-receiver_8h_a8834261a0d544dbf19a055fc39ca70e7}{tvrWaitFrame}}
(\mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aa1c7dceaca29deb6bede1e0b}
WAIT_FOREVER);
```

```
\mbox{\hyperlink{tv-receiver_8h_a29948e841d190dd1695a8b2c344f9603}{tvrGetData}}  
(\mbox{\hyperlink{tv-decoder_8h_adfa467276b1408ea14a2a2d7a613358aa1c7dceaca29deb6bede1e0b}  
surf);  
bitBlit (CONSTR, 0, 0, FRAME_W, FRAME_H, surf, 0, 0, 0, 0);  
}
```

19.77.2. Функции

19.77.2.1. tvrDevCreate()

```
bool tvrDevCreate (  
    int n,  
    tvd_fmt_t fmt,  
    unsigned int width,  
    unsigned int height )
```

Функция создаёт устройство захвата и выделяет память для его работы.

Аргументы

<i>n</i>	Номер канала от 0 до 4. Рекомендуется выбирать значение из перечисления <i>tvd_chnl_t</i> .
<i>fmt</i>	Формат данных из перечисления <i>tvd_fmt_t</i> .
<i>width</i>	Ширина захватываемого кадра.
<i>height</i>	Высота захватываемого кадра.

Возвращает

true при удачном создании устройства, иначе *false*.

19.77.2.2. tvrDevDelete()

```
void tvrDevDelete (  
    int n )
```

Функция удаляет устройство захвата, если таковое было создано.

Аргументы

<i>n</i>	Номер канала от 0 до 4. Рекомендуется выбирать значение из перечисления <i>tvd_chnl_t</i> .
----------	---

19.77.2.3. tvrGetData()

```
bool tvrGetData (
    int n,
    HDC surf )
```

Функция подключает буфер с последним декодированным кадром к указанной поверхности. Физического копирования данных не происходит – функция только заменяет адреса областей памяти поверхности. Рекомендуется получать указатели на последний декодированный кадр сразу после окончания декодирования, о котором можно узнать с помощью [tvrWaitFrame\(\)](#). Декодированные данные следует сразу скопировать, например, в конструируемую поверхность с помощью одной из функций копирования из раздела [Работа с поверхностями](#).

Аргументы

<i>n</i>	Номер канала от 0 до 4. Рекомендуется выбирать значение из перечисления tvd_chnl_t .
<i>surf</i>	Поверхность, к которой будут подключены буферы декодированного кадра.

Возвращает

true при удачном получении данных, иначе *false*.

19.77.2.4. tvrInit()

```
bool tvrInit (
    tvd_interface_t interface,
    tvd_system_t system )
```

Функция настраивает и запускает аппаратную часть TV-декодера.

Аргументы

<i>interface</i>	Выбор входного интерфейса декодера.
<i>system</i>	Выбор стандарта сигнала для CVBS интерфейса.

Возвращает

true при удачной настройке, иначе *false*.

19.77.2.5. tvrWaitFrame()

```
bool tvrWaitFrame (
    int n,
    int timeout )
```

Функция ожидает окончания захвата и декодирования одного кадра в выбранном канале в течение заданного промежутка времени. Ожидание реализовано с использованием функции [taskDelay\(\)](#) и предназначено для использования в отдельной задаче.

Аргументы

<i>n</i>	Номер канала от 0 до 4. Рекомендуется выбирать значение из перечисления <i>tvd_chnl_t</i> .
<i>timeout</i>	Максимальное время ожидания в тиках системного таймера. Возможны так же значения <i>NO_WAIT</i> и <i>WAIT_FOREVER</i> .

Возвращает

true если кадр захвачен раньше указанного времени, иначе *false*.

19.78. Файл `uart.h`

Драйвер UART. Поточковая передача данных.

Макросы

- `#define com_port (debugUartBaseAddress())`
*Базовый адрес порта **debug-uart**.*

Перечисления

- `enum eUartParity {
uartNoParity = 0 , uartOddParity , uartEvenParity , uartMarkParity ,
uartSpaceParity }`
Выбор режима контроля чётности.

Макросы выбора каналов UART

- `#define UART_0 0`
- `#define UART_1 1`
- `#define UART_2 2`
- `#define UART_3 3`
- `#define UART_4 4`
- `#define UART_5 5`
- `#define UART_6 6`
- `#define UART_7 7`

Макросы выбора набора выходных линий UART

- `#define UART_GPIO_ADDITIONAL 1`
- `#define UART_GPIO_DEFAULT 0`

Макросы выбора частоты передаваемого сигнала

- `#define UART_BAUD_115200 115200`
- `#define UART_BAUD_19200 19200`
- `#define UART_BAUD_230400 230400`
- `#define UART_BAUD_2400 2400`
- `#define UART_BAUD_38400 38400`
- `#define UART_BAUD_460800 460800`
- `#define UART_BAUD_4800 4800`
- `#define UART_BAUD_57600 57600`
- `#define UART_BAUD_7200 7200`
- `#define UART_BAUD_921600 921600`
- `#define UART_BAUD_9600 9600`

Запуск аппаратного модуля

- `STATUS uartInit (unsigned int n, unsigned int gpion)`
*Настройка аппаратного модуля **UART**.*
- `const char * uartName (unsigned int n)`
Имя устройства в текстовом виде.

Функции настройки параметров передачи данных

Функции настройки параметров можно использовать как при инициализации аппаратного модуля `uartInit()`, так и в паузах между отправляемыми пакетами данных.

- `STATUS uartSetBaudRate` (unsigned int n, unsigned int baud)
Задать частоту передачи данных.
- `STATUS uartSetBitsNumber` (unsigned int n, unsigned int bits)
Задать количество передаваемых бит.
- `STATUS uartSetParity` (unsigned int n, `eUartParity` parity)
Управлением битом контроля чётности.
- `STATUS uartSetReadBufferSize` (unsigned int n, unsigned int size)
Настройка размера приёмного буфера.
- `STATUS uartSetReadTimeout` (unsigned int n, int timeout)
Настройка таймаута приёма данных.
- `STATUS uartSetStopBitsNumber` (unsigned int n, unsigned int bits)
Задать количество стоп-бит.
- `STATUS uartSetTextMode` (unsigned int n, `bool` enable)
Включение текстового режима (режим консоли).
- `STATUS uartSetWriteBufferSize` (unsigned int n, unsigned int size)
Настройка размера буфера отправки данных.
- `STATUS uartSetWriteTimeout` (unsigned int n, int timeout)
Настройка таймаута передачи данных.

Контроль работы UART

- `STATUS uartFlush` (unsigned int n)
Очистка входного и выходного буферов.
- int `uartReadBufferCounter` (unsigned int n)
Количество байт в приёмном буфере данных.
- unsigned int `uartReadBufferOverflowCounter` (unsigned int n)
Количество ошибок получения данных.
- `STATUS uartSetWaitTxComplete` (unsigned int n, `bool` enable)
Задать режим отправки с ожиданием реальной передачи данных по линии.
- int `uartWriteBufferCounter` (unsigned int n)
Количество байт в передающем буфере данных.

19.78.1. Подробное описание

Настройка аппаратного модуля **UART**.

Подключение:

```
#include <uart.h>
```

См. также

Общее описание работы с последовательным портом **UART** в разделе [UART](#).

19.78.2. Макросы

19.78.2.1. com_port

```
#define com_port (debugUartBaseAddress())
```

Макрос возвращает базовый адрес **UART0** и добавлен для совместимости с предыдущей версией драйвера.

19.78.2.2. UART_0

```
#define UART_0 0
```

Индекс для **UART0**.

19.78.2.3. UART_1

```
#define UART_1 1
```

Индекс для **UART1**.

19.78.2.4. UART_2

```
#define UART_2 2
```

Индекс для **UART2**.

19.78.2.5. UART_3

```
#define UART_3 3
```

Индекс для **UART3**.

19.78.2.6. UART_4

```
#define UART_4 4
```

Индекс для **UART4**.

19.78.2.7. UART_5

```
#define UART_5 5
```

Индекс для **UART5**.

19.78.2.8. UART_6

```
#define UART_6 6
```

Индекс для **UART6**.

19.78.2.9. UART_7

```
#define UART_7 7
```

Индекс для **UART7**.

19.78.2.10. UART_BAUD_115200

```
#define UART_BAUD_115200 115200
```

Частота передачи данных 115200 бит/с.

19.78.2.11. UART_BAUD_19200

```
#define UART_BAUD_19200 19200
```

Частота передачи данных 19200 бит/с.

19.78.2.12. UART_BAUD_230400

```
#define UART_BAUD_230400 230400
```

Частота передачи данных 230400 бит/с.

19.78.2.13. UART_BAUD_2400

```
#define UART_BAUD_2400 2400
```

Частота передачи данных 2400 бит/с.

19.78.2.14. UART_BAUD_38400

```
#define UART_BAUD_38400 38400
```

Частота передачи данных 38400 бит/с.

19.78.2.15. UART_BAUD_460800

```
#define UART_BAUD_460800 460800
```

Частота передачи данных 460800 бит/с.

19.78.2.16. UART_BAUD_4800

```
#define UART_BAUD_4800 4800
```

Частота передачи данных 4800 бит/с.

19.78.2.17. UART_BAUD_57600

```
#define UART_BAUD_57600 57600
```

Частота передачи данных 57600 бит/с.

19.78.2.18. UART_BAUD_7200

```
#define UART_BAUD_7200 7200
```

Частота передачи данных 7200 бит/с.

19.78.2.19. UART_BAUD_921600

```
#define UART_BAUD_921600 921600
```

Частота передачи данных 921600 бит/с.

19.78.2.20. UART_BAUD_9600

```
#define UART_BAUD_9600 9600
```

Частота передачи данных 9600 бит/с.

19.78.2.21. UART_GPIO_ADDITIONAL

```
#define UART_GPIO_ADDITIONAL 1
```

Выбор дополнительного набора линий модуля UART (обычно расположен в старших портах **GPIO**).

19.78.2.22. UART_GPIO_DEFAULT

```
#define UART_GPIO_DEFAULT 0
```

Выбор основного набора линий модуля UART (обычно расположен в младших портах **GPIO**).

19.78.3. Перечисления

19.78.3.1. eUartParity

enum *eUartParity*

Элементы перечислений

uartNoParity	Бит контроля чётности отсутствует.
uartOddParity	Бит контроля нечётности – выставляется так, чтобы дополнить количество единиц в байте до нечётного числа.
uartEvenParity	Бит контроля чётности – выставляется так, чтобы дополнить количество единиц в байте до чётного числа.
uartMarkParity	Бит контроля всегда выставляется в единицу.
uartSpaceParity	Бит контроля всегда выставляется в ноль.

```
00140      {
00141      uartNoParity = 0,
00142      uartOddParity,
00143      uartEvenParity,
00144      uartMarkParity,
00145      uartSpaceParity
00146 } eUartParity;
```

19.78.4. Функции

19.78.4.1. `uartFlush()`

STATUS `uartFlush` (
 unsigned int *n*)

Функция очищает входной и выходной буферы заданного аппаратного модуля **UART**.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

Возвращает

OK если очистка прошла успешно, иначе *ERROR*.

19.78.4.2. `uartInit()`

STATUS `uartInit` (
 unsigned int *n*,
 unsigned int *gpion*)

Функция настраивает регистры выбранного аппаратного модуля и настраивает драйвер потокового ввода / вывода. По умолчанию драйвер **UART** имеет следующие настройки передачи данных:

- скорость 115200 бит/с;
- 8 бит данных;
- 1 старт бит;
- 1 стоп бит;
- без проверки чётности;
- размер входного буфера – 4096 байт;
- размер выходного буфера – 4096 байт;
- таймаут ожидания приёма данных – ждать до появления данных в приёмном буфере;
- таймаут ожидания отправки данных – ждать до успешного помещения данных в приёмный буфер;
- режим передачи данных – бинарный;
- режим ожидания окончания отправки по линии – отключен (см. *`uartSetWaitTxComplete()`*).

Для изменения настроек по умолчанию можно воспользоваться *функциями* настройки параметров передачи.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

Продолжение на следующей странице

Аргументы (Продолжение.)

griop Номер набора портов ввода/вывода. Рекомендуется использовать значение из группы *макросов*.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.3. uartName()

```
const char* uartName (  
    unsigned int n )
```

Имя устройство используется для открытия с помощью стандартной функции *open()*.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

Возвращает

Имя устройства.

19.78.4.4. uartReadBufferCounter()

```
int uartReadBufferCounter (  
    unsigned int n )
```

Функция возвращает количество непочитанных байт из приёмного буфера данных. Возвращаемое значение должно быть строго не отрицательным. Отрицательное число в результате говорит о некорректной работе буфера.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

Возвращает

Количество байт в буфере.

19.78.4.5. uartReadBufferOverflowCounter()

```
unsigned int uartReadBufferOverflowCounter (  
    unsigned int n )
```

Функция возвращает количество ошибок получения данных связанных с переполнением принимающего буфера. Такие ошибки возникают при приёме большого потока данных, когда задача разбора данных не успевает читать все приходящие данные. В этом случае рекомендуется увеличить размер приёмного буфера, либо повысить приоритет такой задачи.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

Возвращает

Количество байт отброшенных в следствие переполнения приёмного буфера.

19.78.4.6. uartSetBaudRate()

```
STATUS uartSetBaudRate (  
    unsigned int n,  
    unsigned int baud )
```

Функция выставляет частоту передачи данных аппаратного модуля **UART**.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

baud Частота передачи в битах в секунду. Рекомендуется выбрать значение из соответствующей группы *макросов*.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.7. uartSetBitsNumber()

```
STATUS uartSetBitsNumber (  
    unsigned int n,  
    unsigned int bits )
```

Функция задаёт количество значащих бит между старт и стоп битами.

Аргументы

n Номер модуля **UART** рекомендуется брать из соответствующей группы *макросов*.

bits Количество значащих бит от 5 до 8 включительно.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.8. `uartSetParity()`

```
STATUS uartSetParity (  
    unsigned int n,  
    eUartParity parity )
```

Функция позволяет задать поведение бита контроля чётности, либо вообще не передавать данный бит.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>parity</i>	Поведение бита контроля чётности.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.9. `uartSetReadBufferSize()`

```
STATUS uartSetReadBufferSize (  
    unsigned int n,  
    unsigned int size )
```

Функция настраивает размер входного буфера. По умолчанию размер буфера **4096** байт. Рекомендуется устанавливать размер входного буфера сопоставимым с максимальным ожидаемым размером единовременно принимаемых данных.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>size</i>	Размер приёмного буфера в байтах.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.10. `uartSetReadTimeout()`

```
STATUS uartSetReadTimeout (  
    unsigned int n,
```

int *timeout*)

Функция настраивает таймаут ожидания приёма данных. Таймаут задается в тиках системного таймера. Один тик по умолчанию равен одной миллисекунде. Допустимы значения *NO_WAIT* и *WAIT_FOREVER*.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>timeout</i>	Таймаут на чтение данных в тиках системного таймера.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.11. uartSetStopBitsNumber()

STATUS uartSetStopBitsNumber (
 unsigned int *n*,
 unsigned int *bits*)

Функция задаёт количество стоп-бит при передаче данных.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>bits</i>	Количество стоп бит от 1 до 2 включительно. При посылках из 5-ти значащих бит, вместо 2 стоп битов будет генерироваться стоп бит длительностью 1,5.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.12. uartSetTextMode()

STATUS uartSetTextMode (
 unsigned int *n*,
 bool *enable*)

Включение текстового режима (режим Консоли).

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>enable</i>	Переключение в режим текстовой консоли. По умолчанию порты работают в бинарном режиме.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.13. `uartSetWaitTxComplete()`

STATUS `uartSetWaitTxComplete` (
 unsigned int *n*,
 bool *enable*)

Функция управляет режимом ожидания реальной отправки данных по линии передачи. По умолчанию функция `write()` ожидает записи данных в приёмный буфер и возвращает результат ещё до того как данные были переданы по линии. С помощью данной функции можно установить режим ожидания окончания отправки данных по линии передачи, в этом случае функция `write()` вернёт управление только после отправки последнего записываемого байта.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>enable</i>	<i>true</i> – режим ожидания отправки всех байт по линии передачи, <i>false</i> – режим ожидания записи данных в приёмный буфер.

Возвращает

OK в случае успешной настройки, иначе *ERROR*.

19.78.4.14. `uartSetWriteBufferSize()`

STATUS `uartSetWriteBufferSize` (
 unsigned int *n*,
 unsigned int *size*)

Функция настраивает размер выходного буфера. По умолчанию размер буфера **4096** байт. Рекомендуется устанавливать размер выходного буфера сопоставимым с максимальным размером одновременно отправляемых данных.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>size</i>	Размер буфера отправки данных в байтах.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.15. `uartSetWriteTimeout()`

```
STATUS uartSetWriteTimeout (  
    unsigned int n,  
    int timeout )
```

Функция настраивает таймаут ожидания освобождения буфера для записи новых данных. Таймаут задается в тиках системного таймера. Один тик по умолчанию равен одной миллисекунде. Допустимы значения *NO_WAIT* и *WAIT_FOREVER*.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
<i>timeout</i>	Таймаут на запись данных в тиках системного таймера.

Возвращает

OK если настройка прошла успешно, иначе *ERROR*.

19.78.4.16. `uartWriteBufferCounter()`

```
int uartWriteBufferCounter (  
    unsigned int n )
```

Функция возвращает количество не отправленных байт в передающем буфере данных. Возвращаемое значение должно быть строго не отрицательным. Отрицательное число в результате говорит о некорректной работе буфера.

Аргументы

<i>n</i>	Номер модуля UART рекомендуется брать из соответствующей группы <i>макросов</i> .
----------	--

Возвращает

Количество байт в буфере.

19.79. Файл uchar.h

Unicode utilities from C11.

Определения типов

- typedef *uint_least16_t* *char16_t*
- typedef *uint_least32_t* *char32_t*

19.79.1. Подробное описание

См. также

C11 standard 7.18.

19.79.2. Типы

19.79.2.1. char16_t

```
typedef uint_least16_t char16_t
```

19.79.2.2. char32_t

```
typedef uint_least32_t char32_t
```

19.80. Файл `udp.h`

Сетевой протокол UDP.

Структуры данных

- struct `ip_packet`
Заголовок пакета IP.
- struct `udp_hdr`
Заголовок пакета UDP.
- struct `udp_service`

Описание заголовков пакетов.

- typedef struct `ip_packet` `IP_PACKET`
Заголовок пакета IP.
- #define `SIZEOF_UDP_HDR` 8
Длина заголовка пакета UDP.
- typedef struct `udp_hdr` `UDP_HDR`
Заголовок пакета UDP.

Служба приёма пакетов

- typedef struct `udp_service` `UDP_SERVICE`
- void `udpKillServices` (void)
Отключить службу приёма пакетов UDP.
- void `udpRegisterService` (unsigned short port, `udpServRout` service)
Зарегистрировать службу приёма пакетов UDP.
- typedef void(* `udpServRout`) (`IP_PACKET` *ipp)
Функция приёма пакета по UDP.

Настройка параметров устройства

- unsigned int `getPrimaryIpAddress` (void)
Получить IP адрес устройства.
- void `iptostr` (unsigned int IP, char *ipstr)
Перевод целого числа в строку IP адреса.
- void `setPrimaryIpAddress` (unsigned int ip)
Установить IP адрес устройства.
- unsigned int `strtoip` (const char *IPAddr)
Перевод строки IP адреса в целое число.

Отправка пакетов

- `STATUS udpSendPacket` (const void *buf, unsigned short len, unsigned short sport, unsigned short dport, unsigned int host)
Отправка пакета UDP.

Обработка полученных данных

- void * `udpGetDataPointer` (`IP_PACKET` *ipp)
Получить указатель на данные UDP.
- `UDP_HDR` * `udpGetUdpHeader` (`IP_PACKET` *ipp)
Получить указатель на заголовок UDP.

19.80.1. Подробное описание

Работа с сетью по протоколу **UDP**.

Подключение:

```
#include <udp.h>
```

См. также

Настройка *Makefile* и файла конфигурации *config.h* в разделе *Подключение сетевой подсистемы*.

Пример работы с сетью по UDP:

```
\#include <udp.h>

static char my_ip[] = "{10.0.3.40}{}";
static char target_ip[] = "{10.0.3.41}{}";
static unsigned int port = 1243;

// Функция приёма пакетов
void udpReceiveService (IP_PACKET *ipp) {
    if (ipp->s_addr != strtoint (my_ip))
        return;
    void *data = udpGetDataPointer (ipp);
    // ... Некие действия с принятыми данными data
}

void startUdp () {
    // Задать адрес устройства
    setPrimaryIpAddress (strtoint (my_ip));
    // Регистрация функции приёма данных
    udpRegisterService (port, udpReceiveService);
    int dataLen = 512;
    char data[dataLen];
    // ... Заполнение отправляемого массива data
    udpSendPacket (data, dataLen, port, port, strtoint (target_ip));
}
```

19.80.2. Макросы

19.80.2.1. SIZEOF_UDPHDR

```
#define SIZEOF_UDPHDR 8
```

19.80.3. Типы

19.80.3.1. IP_PACKET

```
typedef struct ip_packet IP_PACKET
```

Заголовок пакета **IP**. Порядок следования байт – младшим байтом вперёд.

19.80.3.2. UDP_HDR

```
typedef struct udp_hdr UDP_HDR
```

Заголовок пакета **UDP**. Порядок следования байт – старшим байтом вперёд. Для чтения значений байты следует менять местами.

19.80.3.3. UDP_SERVICE

```
typedef struct udp_service UDP_SERVICE
```

19.80.3.4. udpServRout

```
typedef void(* udpServRout) (IP_PACKET *ipp)
```

Функцию приём следует зарегистрировать с помощью *udpRegisterService()*.

19.80.4. Функции

19.80.4.1. getPrimaryIpAddress()

```
unsigned int getPrimaryIpAddress (  
    void )
```

Функция позволяет получить текущий **IP** адрес устройства.

Возвращает

ip Первичный **IP** адрес.

19.80.4.2. iptostr()

```
void iptostr (  
    unsigned int IP,  
    char * ipstr )
```

Функция переводит целое число в строку **IP** адреса.

Аргументы

IP Беззнаковое целое.

ipstr Строка, в которую будет помещён результат перевода. Размер предоставляемого массива должен быть не менее 16 байт.

19.80.4.3. setPrimaryIpAddress()

```
void setPrimaryIpAddress (  
    unsigned int ip )
```

Функция позволяет изменить **IP** адрес устройства.

Аргументы

ip Первичный **IP** адрес.

19.80.4.4. strtoint()

```
unsigned int strtoint (  
    const char * IPAddr )
```

Перевод строки **IP** адреса в целое число.

Аргументы

IPAddr Указатель на строку, содержащую **IP** адрес.

Возвращает

Целое число.

19.80.4.5. udpGetDataPointer()

```
void* udpGetDataPointer (  
    IP_PACKET * ipp )
```

Функция возвращает указатель на данные **UDP** пакета внутри пакета **IP**.

Аргументы

ipp Пакет **IP**.

Возвращает

Указатель на данные, которые следуют сразу после заголовка *UDP_HDR*.

19.80.4.6. udpGetUdpHeader()

```
UDP_HDR* udpGetUdpHeader (
```

*IP_PACKET * ipp*)

Функция возвращает указатель на заголовок **UDP** пакета внутри пакета **IP**.

Аргументы

ipp Пакет **IP**.

Возвращает

Указатель на заголовок **UDP** в пакете **IP**.

19.80.4.7. udpKillServices()

```
void udpKillServices (  
    void )
```

Отключение всех подключенных функций приёма пакетов от всех портов.

19.80.4.8. udpRegisterService()

```
void udpRegisterService (  
    unsigned short port,  
    udpServRout service )
```

Подключение функции разбора входящих пакетов к заданному порту. Функция будет вызываться в системной **задаче** разбора принимаемых пакетов.

Аргументы

port Порт, на который подключается функция.

service Подключаемая функция приёма пакетов.

19.80.4.9. udpSendPacket()

```
STATUS udpSendPacket (  
    const void * buf,  
    unsigned short len,  
    unsigned short sport,  
    unsigned short dport,  
    unsigned int host )
```

Функция отправляет пакет и дожидается конца отправки.

Аргументы

buf Указатель на передаваемые данные.

Продолжение на следующей странице

Аргументы (Продолжение.)	
<i>len</i>	Размер передаваемых данных в байтах.
<i>sport</i>	Номер порта отправителя.
<i>dport</i>	Номер порта получателя.
<i>host</i>	Адрес порта, получаемый с помощью <i>strtoip()</i> .

Возвращает

OK при удачной отправке пакета, иначе *ERROR*.

19.81. Файл `unicode.h`

Преобразование строк и символов между различными кодировками (UTF-16, UTF-8, Cp1251, Cp866, Ascii).

Функции

- `size_t convert_AsciiToUtf16` (`const char *pAsciiString`, `char16_t *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Cp1251ToUtf16` (`const char *pCp1251String`, `char16_t *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Cp1251ToUtf8` (`const char *pCp1251String`, `char *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Utf16ToAscii` (`const char16_t *pUtf16String`, `char *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Utf16ToCp1251` (`const char16_t *pUtf16String`, `char *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Utf16ToUtf8` (`const char16_t *pUtf16String`, `char *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Utf8ToCp1251` (`const char *pUtf8String`, `char *pOutputBuffer`, `size_t outputBufferLength`)
- `size_t convert_Utf8ToUtf16` (`const char *pUtf8String`, `char16_t *pOutputBuffer`, `size_t outputBufferLength`)
- `unsigned char dos2win` (`unsigned char c`)
- `unsigned int uni2char` (`unsigned short *uni`, `unsigned char *res`)
- `unsigned char win2dos` (`unsigned char c`)

19.81.1. Функции

19.81.1.1. `convert_AsciiToUtf16()`

```
size_t convert_AsciiToUtf16 (
    const char * pAsciiString,
    char16_t * pOutputBuffer,
    size_t outputBufferLength )
```

Преобразовать строку в кодировке ASCII в строку в кодировке UTF-16.

Аргументы	
<code>pAsciiString</code>	Исходная строка в кодировке ASCII.
<code>pOutputBuffer</code>	Буфер для вывода.
<code>outputBufferLength</code>	Размер буфера для вывода (в символах, не байтах).

Возвращает

Кол-во записанных в буфер для вывода символов (не байтов).

19.81.1.2. `convert_Cp1251ToUtf16()`

```
size_t convert_Cp1251ToUtf16 (  
    const char * pCp1251String,  
    char16_t * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке CP-1251 в строку в кодировке UTF-16.

Аргументы	
<i>pCp1251String</i>	Исходная строка в кодировке CP-1251.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода (в символах, не байтах).

Возвращает

Кол-во записанных в буфер для вывода символов (не байтов).

19.81.1.3. convert_Cp1251ToUtf8()

```
size_t convert_Cp1251ToUtf8 (  
    const char * pCp1251String,  
    char * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке CP-1251 в строку в кодировке UTF-8.

Аргументы	
<i>pCp1251String</i>	Исходная строка в кодировке CP-1251.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода.

Возвращает

Кол-во записанных в буфер для вывода байтов.

19.81.1.4. convert_Utf16ToAscii()

```
size_t convert_Utf16ToAscii (  
    const char16_t * pUtf16String,  
    char * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке UTF-16 в строку в кодировке ASCII.

Аргументы

<i>pUtf16String</i>	Исходная строка в кодировке UTF-16.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода.

Возвращает

Кол-во записанных в буфер для вывода символов.



Символы, не подходящие под целевую кодировку, будут преобразованы в пробелы.

19.81.1.5. convert_Utf16ToCp1251()

```
size_t convert_Utf16ToCp1251 (  
    const char16_t * pUtf16String,  
    char * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке UTF-16 в строку в кодировке CP-1251.

Аргументы

<i>pUtf16String</i>	Исходная строка в кодировке UTF-16.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода.

Возвращает

Кол-во записанных в буфер для вывода символов.



Символы, не подходящие под целевую кодировку, будут преобразованы в пробелы.

19.81.1.6. convert_Utf16ToUtf8()

```
size_t convert_Utf16ToUtf8 (  
    const char16_t * pUtf16String,  
    char * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке UTF-16 в строку в кодировке UTF-8.

Аргументы

<i>pUtf16String</i>	Исходная строка в кодировке UTF-16.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода.

Возвращает

Кол-во записанных в буфер для вывода байтов.



Поддерживаются только 1-но символьные UTF-16 символы. При обнаружении 2х-символьного символа обработка строки будет прекращена (итоговая строка все равно будет нуль-терминирована).

19.81.1.7. convert_Utf8ToCp1251()

```
size_t convert_Utf8ToCp1251 (  
    const char * pUtf8String,  
    char * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке UTF-8 в строку в кодировке CP-1251.

Аргументы

<i>pUtf8String</i>	Исходная строка в кодировке UTF-8.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода.

Возвращает

Кол-во записанных в буфер для вывода байтов.



4х-байтные UTF-8 символы не поддерживаются. При обнаружении 4х-байтного символа обработка строки будет прекращена (итоговая строка все равно будет нуль-терминирована). Символы, не подходящие под целевую кодировку, будут преобразованы в пробелы.

19.81.1.8. convert_Utf8ToUtf16()

```
size_t convert_Utf8ToUtf16 (  
    const char * pUtf8String,  
    char16_t * pOutputBuffer,  
    size_t outputBufferLength )
```

Преобразовать строку в кодировке UTF-8 в строку в кодировке UTF-16.

Аргументы	
<i>pUtf8String</i>	Исходная строка в кодировке UTF-8.
<i>pOutputBuffer</i>	Буфер для вывода.
<i>outputBufferLength</i>	Размер буфера для вывода (в символах, не байтах).



4х-байтные UTF-8 символы не поддерживаются. При обнаружении 4х-байтного символа обработка строки будет прекращена (итоговая строка все равно будет нуль-терминирована).

Возвращает

Кол-во записанных в буфер для вывода символов (не байтов).

19.81.1.9. dos2win()

```
unsigned char dos2win (  
    unsigned char c )
```

Преобразовать символ из кодировки CP-866 в кодировку CP-1251.

Аргументы	
<i>c</i>	Символ в кодировке CP-866.

Возвращает

Соответствующий ему символ в кодировке CP-1251.

19.81.1.10. uni2char()

```
unsigned int uni2char (  
    unsigned short * uni,  
    unsigned char * res )
```

Преобразовать нуль-терминированную строку в кодировке UTF-16 в нуль-терминированную строку в кодировке CP-1251.

Аргументы	
<i>uni</i>	Исходная строка в кодировке UTF-16.
<i>res</i>	Буфер под строку в кодировке CP-1251.

Возвращает

Количество символов в итоговой строке.

19.81.1.11. win2dos()

```
unsigned char win2dos (  
    unsigned char c )
```

Преобразовать символ из кодировки CP-1251 в кодировку CP-866.

Аргументы

c Символ в кодировке CP-1251.

Возвращает

Соответствующий ему символ в кодировке CP-866.

19.82. Файл usb.h

Методы работы с **USB**.

Структуры данных

- struct `usb_config`
Структура дескриптора конфигурации.
- struct `usb_device`
Структура данных о подключении USB-устройства.
- struct `usb_interface`
Структура дескриптора интерфейса.

Перечисления

- enum { `PACKET_SIZE_8` = 0 , `PACKET_SIZE_16` = 1 , `PACKET_SIZE_32` = 2 , `PACKET_SIZE_64` = 3 }
- Размер пакета для обмена с USB-устройством.*

Конфигурация для OMAP3

- #define `_LITTLE_ENDIAN`
- #define `ARCH_DMA_MINALIGN` 64
- #define `CONFIG_USB_EHCI`

Настройки USB

- #define `USB_ALTSETTINGALLOC` 4
- #define `USB_CNTL_TIMEOUT` 100
100 ms timeout
- #define `USB_DMA_MINALIGN` `ARCH_DMA_MINALIGN`
- #define `USB_MAX_DEVICE` 32
- #define `USB_MAX_HUB` 16
- #define `USB_MAXALTSETTING` 128
Hard limit.
- #define `USB_MAXCHILDREN` 8
This is arbitrary.
- #define `USB_MAXCONFIG` 8
- #define `USB_MAXENDPOINTS` 16
- #define `USB_MAXINTERFACES` 8
- #define `USB_TIMEOUT_MS(pipe)` (`usb_pipebulk(pipe)` ? 5000 : 1000)

Настройка идентификации устройства.

- #define `USB_UHCI_DEV_ID` 0x7112
- #define `USB_UHCI_VEND_ID` 0x8086

Макросы сдвига переменных

Конвертация переменных **big endian** -> **little endian**. Некоторые процессоры, например **ARM920T**, исходно работают со значениями **little endian**.

- #define `_swap_16(x)`
- #define `_swap_32(x)`
- #define `swap_16(x)` (x)
- #define `swap_32(x)` (x)

Создание интегрального параметра pipe

Подробнее см в разделе *Использование интегрального параметра pipe*.

- #define *create_pipe*(dev, endpoint)
- #define *default_pipe*(dev) ((dev)->speed << 26)
- #define *usb_rcvbulkpipe*(dev, endpoint)
- #define *usb_rcvctrlpipe*(dev, endpoint)
- #define *usb_rcvdefctrl*(dev)
- #define *usb_rcvintpipe*(dev, endpoint)
- #define *usb_rcvisocpipe*(dev, endpoint)
- #define *usb_sndbulkpipe*(dev, endpoint)
- #define *usb_sndctrlpipe*(dev, endpoint)
- #define *usb_snddefctrl*(dev)
- #define *usb_sndintpipe*(dev, endpoint)
- #define *usb_sndisocpipe*(dev, endpoint)

Работа с интегральным параметром pipe

Подробнее см в разделе *Использование интегрального параметра pipe*.

- #define *usb_dotoggle*(dev, ep, out) ((dev)->toggle[out] ^= (1 << ep))
- #define *usb_endpoint_halt*(dev, ep, out) ((dev)->halted[out] |= (1 << (ep)))
- #define *usb_endpoint_halted*(dev, ep, out) ((dev)->halted[out] & (1 << (ep)))
- #define *usb_endpoint_out*(ep_dir) (((ep_dir >> 7) & 1) ^ 1)
- #define *usb_endpoint_running*(dev, ep, out) ((dev)->halted[out] &= ~(1 << (ep)))
- #define *usb_gettoggle*(dev, ep, out) (((dev)->toggle[out] >> ep) & 1)
- #define *usb_packetid*(pipe)
- #define *usb_pipe_endpdev*(pipe) (((pipe) >> 8) & 0x7ff)
- #define *usb_pipebulk*(pipe) (*usb_pipetype*((pipe)) == PIPE_BULK)
- #define *usb_pipecontrol*(pipe) (*usb_pipetype*((pipe)) == PIPE_CONTROL)
- #define *usb_pipedata*(pipe) (((pipe) >> 19) & 1)
- #define *usb_pipedevice*(pipe) (((pipe) >> 8) & 0x7f)
- #define *usb_pipeendpoint*(pipe) (((pipe) >> 15) & 0xf)
- #define *usb_pipein*(pipe) (((pipe) >> 7) & 1)
- #define *usb_pipeint*(pipe) (*usb_pipetype*((pipe)) == PIPE_INTERRUPT)
- #define *usb_pipeisoc*(pipe) (*usb_pipetype*((pipe)) == PIPE_ISOCHRONOUS)
- #define *usb_pipeout*(pipe) (((pipe) >> 7) & 1) ^ 1)
- #define *usb_pipeslow*(pipe) (*usb_pipespeed*(pipe) == USB_SPEED_LOW)
- #define *usb_pipespeed*(pipe) (((pipe) >> 26) & 3)
- #define *usb_pipetype*(pipe) (((pipe) >> 30) & 3)
- #define *usb_settoggle*(dev, ep, out, bit)

Проверка подключения устройства.

- #define *UCS_FLASHDRIVE_IS_CONNECTED* (0x02)
- #define *UCS_KEYBOARD_IS_CONNECTED* (0x04)
- #define *UCS_SOMETHING_IS_CONNECTED* (0x01)
- int *usb_connection_status* ()

Проверка подключения устройства.

Работа с конечными точками

- int *submit_int_msg* (struct *usb_device* *dev, unsigned long pipe, void *buffer, int transfer_len, int interval)
Подключить обработчик прерывания к конечной точке типа interrupt.
- int *usb_bulk_msg* (struct *usb_device* *dev, unsigned int pipe, void *data, int len, int *actual_length, int timeout)
Передача информации к конечной точке типа bulk и от неё.

Функции работы с управляющей конечной точкой

Для работы с управляющей конечной точкой используются **control message**.

- int *usb_set_configuration* (struct *usb_device* *dev, int configuration)
Задать конфигурацию устройства.
- int *usb_set_interface* (struct *usb_device* *dev, int interface, int alternate)
Задать интерфейс устройства.

19.82.1. Подробное описание

Файл содержит описание методов работы с **USB** в ОС *MULTEX-ARM*.

Подключение:

```
#include <usb.h>
```

config.h:

```
#define INCLUDE_USB
```

Makefile:

```
LIBRARIES += -l_usb_sunxi
```

См. также

Общее описание работы с **USB** в главе *Подсистема USB*.

19.82.2. Макросы

19.82.2.1. `__LITTLE_ENDIAN`

```
#define __LITTLE_ENDIAN
```

19.82.2.2. `__swap_16`

```
#define __swap_16(  
    x)
```

Макроопределение:

```
({ unsigned short x_ = (unsigned short)x; \  
  (unsigned short)( \  
    ((x_ \& 0x00FFU) << 8) | ((x_ \& 0xFF00U) >> 8)); \  
})
```

19.82.2.3. __swap_32

```
#define __swap_32(  
    x)
```

Макроопределение:

```
({ unsigned long x_ = (unsigned long)x; \  
  (unsigned long)( \  
    ((x_ \& 0x000000FFUL) << 24) | \  
    ((x_ \& 0x0000FF00UL) << 8) | \  
    ((x_ \& 0x00FF0000UL) >> 8) | \  
    ((x_ \& 0xFF000000UL) >> 24)); \  
})
```

19.82.2.4. ARCH_DMA_MINALIGN

```
#define ARCH_DMA_MINALIGN 64
```

19.82.2.5. CONFIG_USB_EHCI

```
#define CONFIG_USB_EHCI
```

19.82.2.6. create_pipe

```
#define create_pipe(  
    dev,  
    endpoint)
```

Макроопределение:

```
((dev)->devnum << 8) | ((endpoint) << 15) | \  
((dev)->speed << 26) | (dev)->maxpacketize)
```

19.82.2.7. default_pipe

```
#define default_pipe(  
    dev ) ((dev)->speed << 26)
```

19.82.2.8. swap_16

```
#define swap_16(  
    x ) (x)
```

19.82.2.9. swap_32

```
#define swap_32(  
    x ) (x)
```

19.82.2.10. UCS_FLASHDRIVE_IS_CONNECTED

```
#define UCS_FLASHDRIVE_IS_CONNECTED (0x02)
```

Подключена флешка, отключение не отслеживается.

19.82.2.11. UCS_KEYBOARD_IS_CONNECTED

```
#define UCS_KEYBOARD_IS_CONNECTED (0x04)
```

Подключена клавиатура.

19.82.2.12. UCS_SOMETHING_IS_CONNECTED

```
#define UCS_SOMETHING_IS_CONNECTED (0x01)
```

Подключено какое-то устройство, отключение не отслеживается.

19.82.2.13. USB_ALTSETTINGALLOC

```
#define USB_ALTSETTINGALLOC 4
```

19.82.2.14. USB_CNTL_TIMEOUT

```
#define USB_CNTL_TIMEOUT 100
```

19.82.2.15. USB_DMA_MINALIGN

```
#define USB_DMA_MINALIGN ARCH_DMA_MINALIGN
```

19.82.2.16. usb_dotoggle

```
#define usb_dotoggle(  
    dev,  
    ep,  
    out ) ((dev)->toggle[out] ^= (1 << ep))
```

19.82.2.17. usb_endpoint_halt

```
#define usb_endpoint_halt(  
    dev,  
    ep,  
    out ) ((dev)->halted[out] |= (1 << (ep)))
```

19.82.2.18. usb_endpoint_halted

```
#define usb_endpoint_halted(  
    dev,  
    ep,  
    out ) ((dev)->halted[out] & (1 << (ep)))
```

19.82.2.19. usb_endpoint_out

```
#define usb_endpoint_out(  
    ep_dir ) (((ep_dir >> 7) & 1) ^ 1)
```

19.82.2.20. usb_endpoint_running

```
#define usb_endpoint_running(  
    dev,  
    ep,  
    out ) ((dev)->halted[out] &= ~(1 << (ep)))
```

19.82.2.21. usb_gettoggle

```
#define usb_gettoggle(  
    dev,  
    ep,  
    out ) (((dev)->toggle[out] >> ep) & 1)
```

19.82.2.22. USB_MAX_DEVICE

```
#define USB_MAX_DEVICE 32
```

19.82.2.23. USB_MAX_HUB

```
#define USB_MAX_HUB 16
```

19.82.2.24. USB_MAXALTSETTING

```
#define USB_MAXALTSETTING 128
```

19.82.2.25. USB_MAXCHILDREN

```
#define USB_MAXCHILDREN 8
```

19.82.2.26. USB_MAXCONFIG

```
#define USB_MAXCONFIG 8
```

19.82.2.27. USB_MAXENDPOINTS

```
#define USB_MAXENDPOINTS 16
```

19.82.2.28. USB_MAXINTERFACES

```
#define USB_MAXINTERFACES 8
```

19.82.2.29. usb_packetid

```
#define usb_packetid(  
    pipe )
```

Макроопределение:

```
((pipe) & USB_DIR_IN) ? USB_PID_IN : \  
USB_PID_OUT)
```

19.82.2.30. usb_pipe_endpdev

```
#define usb_pipe_endpdev(  
    pipe ) (((pipe) >> 8) & 0x7ff)
```

19.82.2.31. usb_pipebulk

```
#define usb_pipebulk(  
    pipe ) (usb_pipetype((pipe)) == PIPE_BULK)
```

19.82.2.32. usb_pipecontrol

```
#define usb_pipecontrol(  
    pipe ) (usb_pipetype((pipe)) == PIPE_CONTROL)
```

19.82.2.33. usb_pipedata

```
#define usb_pipedata(  
    pipe ) (((pipe) >> 19) & 1)
```

19.82.2.34. usb_pipedevice

```
#define usb_pipedevice(  
    pipe ) (((pipe) >> 8) & 0x7f)
```

19.82.2.35. usb_pipeendpoint

```
#define usb_pipeendpoint(  
    pipe ) (((pipe) >> 15) & 0xf)
```

19.82.2.36. usb_pipein

```
#define usb_pipein(  
    pipe ) (((pipe) >> 7) & 1)
```

19.82.2.37. usb_pipeint

```
#define usb_pipeint(  
    pipe ) (usb_pipetype((pipe)) == PIPE_INTERRUPT)
```

19.82.2.38. usb_pipeisoc

```
#define usb_pipeisoc(  
    pipe ) (usb_pipetype((pipe)) == PIPE_ISOCHRONOUS)
```

19.82.2.39. usb_pipeout

```
#define usb_pipeout(  
    pipe ) (((pipe) >> 7) & 1) ^ 1)
```

19.82.2.40. usb_pipeslow

```
#define usb_pipeslow(  
    pipe ) (usb_pipespeed(pipe) == USB_SPEED_LOW)
```

19.82.2.41. usb_pipespeed

```
#define usb_pipespeed(  
    pipe ) (((pipe) >> 26) & 3)
```

19.82.2.42. usb_pipetype

```
#define usb_pipetype(  
    pipe ) (((pipe) >> 30) & 3)
```

19.82.2.43. usb_rcvbulkpipe

```
#define usb_rcvbulkpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_BULK << 30) | \  
create_pipe(dev, endpoint) | \  
USB_DIR_IN)
```

19.82.2.44. usb_rcvctrlpipe

```
#define usb_rcvctrlpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_CONTROL << 30) | \  
create_pipe(dev, endpoint) | \  
USB_DIR_IN)
```

19.82.2.45. usb_rcvdefctrl

```
#define usb_rcvdefctrl(  
    dev )
```

Макроопределение:

```
((PIPE_CONTROL << 30) | \  
default_pipe(dev) | \  
USB_DIR_IN)
```

19.82.2.46. usb_rcvintpipe

```
#define usb_rcvintpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_INTERRUPT << 30) | \  
create_pipe(dev, endpoint) | \  
USB_DIR_IN)
```

19.82.2.47. usb_rcvisocpipe

```
#define usb_rcvisocpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_ISOCHRONOUS << 30) | \  
create_pipe(dev, endpoint) | \  
USB_DIR_IN)
```

19.82.2.48. usb_settoggle

```
#define usb_settoggle(  
    dev,  
    ep,  
    out,
```


bit)

Макроопределение:

```
((dev)->toggle[out] = \  
((dev)->toggle[out] \& \  
~(1 << ep)) | ((bit) << ep))
```

19.82.2.49. usb_sndbulkpipe

```
#define usb_sndbulkpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_BULK << 30) | \  
create_pipe(dev, endpoint))
```

19.82.2.50. usb_sndctrlpipe

```
#define usb_sndctrlpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_CONTROL << 30) | \  
create_pipe(dev, endpoint))
```

19.82.2.51. usb_snddefctrl

```
#define usb_snddefctrl(  
    dev )
```

Макроопределение:

```
((PIPE_CONTROL << 30) | \  
default_pipe(dev))
```

19.82.2.52. `usb_sndintpipe`

```
#define usb_sndintpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_INTERRUPT << 30) | \  
create_pipe(dev, endpoint))
```

19.82.2.53. `usb_sndisocpipe`

```
#define usb_sndisocpipe(  
    dev,  
    endpoint )
```

Макроопределение:

```
((PIPE_ISOCHRONOUS << 30) | \  
create_pipe(dev, endpoint))
```

19.82.2.54. `USB_TIMEOUT_MS`

```
#define USB_TIMEOUT_MS(  
    pipe ) (usb_pipebulk(pipe) ? 5000 : 1000)
```

19.82.2.55. `USB_UHCI_DEV_ID`

```
#define USB_UHCI_DEV_ID 0x7112
```

Определение индекса строкового дескриптора.

19.82.2.56. `USB_UHCI_VEND_ID`

```
#define USB_UHCI_VEND_ID 0x8086
```

Определение идентификатора поставщика для устройства.

19.82.3. Перечисления

19.82.3.1. anonymous enum

anonymous enum

Используется для определения размера пакета в структуре *usb_device*.

Элементы перечислений

PACKET_SIZE_8 8 байт.

PACKET_SIZE_16 16 байт.

PACKET_SIZE_32 32 байта.

PACKET_SIZE_64 64 байта.

```
00146     {
00147     PACKET_SIZE_8    = 0,
00148     PACKET_SIZE_16   = 1,
00149     PACKET_SIZE_32   = 2,
00150     PACKET_SIZE_64   = 3,
00151   };
```

19.82.4. Функции

19.82.4.1. submit_int_msg()

```
int submit_int_msg (
    struct usb_device * dev,
    unsigned long pipe,
    void * buffer,
    int transfer_len,
    int interval)
```

Функция позволяет подключать обработчик прерывания к конечной точке типа **interrupt**. Чтобы подключить обработчик на прерывание от конечной точки, нужно перед этой командой записать следующую строку:

```
dev->irq_handle = usb_my_irq;
```

Где **usb_my_irq** – функция вида:

```
int usb_my_irq(struct usb_device *dev);
```

Для многократного входа в прерывание эта функция должна вернуть ненулевое значение. Так, например, для драйвера указателя типа *мышь* необходимо записать следующие строки:

```
// --- Установка обработчика ---  
// Обработчик прерывания мыши  
dev->irq_handle = usb_mouse_irq;  
  
// Сформировать pipe  
unsigned int pipe = usb_rcvintpipe(dev, ep->bEndpointAddress);  
  
// Определить размер пакета  
int maxp = usb_maxpacket(dev, pipe);  
  
// Подключить обработчик  
usb_submit_int_msg(dev, pipe, data, maxp, ep->bInterval);
```

Аргументы

<i>dev</i>	Устройство USB.
<i>pipe</i>	Интегральный параметр <i>pipe</i> .
<i>buffer</i>	Указатель на буфер.
<i>transfer_len</i>	Длина в байтах
<i>interval</i>	Период обменов в микрофреймах (~1мс).

Возвращает

OK при успешном завершении, или код ошибки.

19.82.4.2. usb_bulk_msg()

```
int usb_bulk_msg (  
    struct usb_device * dev,  
    unsigned int pipe,  
    void * data,  
    int len,  
    int * actual_length,  
    int timeout )
```

Эта функция позволяет передавать информацию на конечную точку и принимать информацию от конечной точки типа **bulk**.

Аргументы

<i>dev</i>	Устройство USB.
<i>pipe</i>	Интегральный параметр <i>pipe</i> .
<i>data</i>	Указатель на буфер данных.
<i>len</i>	Длина в байтах.

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>actual_length</i>	Фактическая длина данных.
<i>timeout</i>	Таймаут в миллисекундах.

Возвращает

OK при успешном завершении, или код ошибки.

19.82.4.3. `usb_connection_status()`

```
int usb_connection_status ( )
```

Функция проверки подключения устройства. Определение подключения осуществляется проверкой, взведен ли соответствующий бит из набора *UCS_SOMETHING_IS_CONNECTED*, *UCS_FLASHDRIVE_IS_CONNECTED*, *UCS_KEYBOARD_IS_CONNECTED*.

19.82.4.4. `usb_set_configuration()`

```
int usb_set_configuration (
    struct usb_device * dev,
    int configuration )
```

Функция устанавливает заданную конфигурацию устройства.

Аргументы

<i>dev</i>	Устройство USB.
<i>configuration</i>	Выбранный номер конфигурации.

Возвращает

OK при успешном завершении, или код ошибки.

19.82.4.5. `usb_set_interface()`

```
int usb_set_interface (
    struct usb_device * dev,
    int interface,
    int alternate )
```

Функция устанавливает требуемый интерфейс для устройства по номеру.

Аргументы

<i>dev</i>	Устройство USB.
------------	-----------------

Продолжение на следующей странице

Аргументы (Продолжение.)

<i>interface</i>	Интерфейс, который будет обновлён.
<i>alternate</i>	Выбранная настройка.

Возвращает

OK при успешном завершении, или код ошибки.

19.83. Файл `usb_driver.h`

Регистрация USB драйвера в *MULTEX-ARM*.

Функции

- void `usb_register_driver` (char *name, int(*probe)(struct `usb_device` *dev), void(*disconnect)(struct `usb_device` *dev))
Регистрация драйвера в системе.

19.83.1. Подробное описание

Файл содержит функции регистрации драйверов **USB** в ОС *MULTEX-ARM*.

Подключение:

```
#include <usb_driver.h>
```

config.h:

```
#define INCLUDE_USB
```

Makefile:

```
LIBRARIES += -l_usb_sunxi
```

См. также

Общее описание работы с **USB** в главе *Подсистема USB*.

19.83.2. Функции

19.83.2.1. `usb_register_driver()`

```
void usb_register_driver (  
    char * name,  
    int(*)(struct usb_device *dev) probe,  
    void(*)(struct usb_device *dev) disconnect )
```

Функция позволяет зарегистрировать драйвер устройства USB в системе. Для этого драйвер должен содержать две функции: **probe()** и **disconnect()**. Функция **probe()** должна убедиться (исследуя дескрипторы устройства), что драйвер подходит для данного устройства и настроить устройство для работы (выбрать интерфейс, конфигурацию и протокол). Функция **disconnect()** должна обеспечить освобождение всех ресурсов, созданных функцией **probe()**.

Аргументы

<i>name</i>	Имя устройства USB.
<i>probe</i>	Функция probe .
<i>disconnect</i>	Функция отсоединения устройства.

19.84. Файл `usbdescriptors.h`

Структуры дескрипторов **USB**.

Структуры данных

- struct `usb_class_abstract_control_descriptor`
- struct `usb_class_atm_networking_descriptor`
- struct `usb_class_call_management_descriptor`
- struct `usb_class_capi_control_descriptor`
- struct `usb_class_country_selection_descriptor`
- struct `usb_class_descriptor`
- struct `usb_class_direct_line_descriptor`
- struct `usb_class_ethernet_networking_descriptor`
- struct `usb_class_extension_unit_descriptor`
- struct `usb_class_function_descriptor`
- struct `usb_class_function_descriptor_generic`
- struct `usb_class_header_function_descriptor`
- struct `usb_class_hid_descriptor`
- struct `usb_class_mdln_descriptor`
- struct `usb_class_mdlnmd_descriptor`
- struct `usb_class_multi_channel_descriptor`
- struct `usb_class_network_channel_descriptor`
- struct `usb_class_protocol_unit_function_descriptor`
- struct `usb_class_report_descriptor`
- struct `usb_class_telephone_call_descriptor`
- struct `usb_class_telephone_operational_descriptor`
- struct `usb_class_telephone_ringer_descriptor`
- struct `usb_class_union_function_descriptor`
- struct `usb_class_usb_terminal_descriptor`
- struct `usb_configuration_descriptor`
Структура дескриптора конфигурации.
- struct `usb_descriptor`
- struct `usb_device_descriptor`
*Структура дескриптора устройства **USB**.*
- struct `usb_endpoint_descriptor`
Структура дескриптора конечной точки.
- struct `usb_generic_descriptor`
- struct `usb_interface_descriptor`
Структура дескриптора интерфейса.
- struct `usb_string_descriptor`
Структура дескриптора строки.

Макросы

- #define `BMATTRIBUTE_RESERVED` 0x80
- #define `BMATTRIBUTE_SELF_POWERED` 0x40
- #define `BULK` 0x02
- #define `CLASS_BCD_VERSION` 0x0110
- #define `COMMUNICATIONS_ACM_SUBCLASS` 0x02
- #define `COMMUNICATIONS_ANCM_SUBCLASS` 0x07
- #define `COMMUNICATIONS_CCM_SUBCLASS` 0x05
- #define `COMMUNICATIONS_DEVICE_CLASS` 0x02
- #define `COMMUNICATIONS_DLDM_SUBCLASS` 0x01
- #define `COMMUNICATIONS_DMM_SUBCLASS` 0x09
- #define `COMMUNICATIONS_ENCM_SUBCLASS` 0x06
- #define `COMMUNICATIONS_INTERFACE_CLASS_CONTROL` 0x02
- #define `COMMUNICATIONS_INTERFACE_CLASS_DATA` 0x0A
- #define `COMMUNICATIONS_INTERFACE_CLASS_VENDOR` 0xFF
- #define `COMMUNICATIONS_MCCM_SUBCLASS` 0x04
- #define `COMMUNICATIONS_MDLM_SUBCLASS` 0x0A

- #define *COMMUNICATIONS_NO_PROTOCOL* 0x00
- #define *COMMUNICATIONS_NO_SUBCLASS* 0x00
- #define *COMMUNICATIONS_OBEX_SUBCLASS* 0x0b
- #define *COMMUNICATIONS_TCM_SUBCLASS* 0x03
- #define *COMMUNICATIONS_V25TER_PROTOCOL* 0x01 /*Common AT Hayes compatible*/
- #define *COMMUNICATIONS_WHCM_SUBCLASS* 0x08
- #define *CONTROL* 0x00
- #define *CS_ENDPOINT* 0x25
- #define *CS_INTERFACE* 0x24
- #define *DATA_INTERFACE_CLASS* 0x0a
- #define *DATA_INTERFACE_PROTOCOL_NONE* 0x00 /* No class protcol required */
- #define *DATA_INTERFACE_SUBCLASS_NONE* 0x00 /* No subclass pertinent */
- #define *IN* 0x80
- #define *INTERRUPT* 0x03
- #define *ISOCHRONOUS* 0x01
- #define *OUT* 0x00
- #define *print_device_descriptor*(d)
- #define *USB_ST_ACMF* 0x02
- #define *USB_ST_ATMNF* 0x10
- #define *USB_ST_CCMF* 0x0e
- #define *USB_ST_CMF* 0x01
- #define *USB_ST_CS* 0x16
- #define *USB_ST_CSD* 0x17
- #define *USB_ST_CSF* 0x07
- #define *USB_ST_DLMF* 0x03
- #define *USB_ST_DMM* 0x14
- #define *USB_ST_ENF* 0x0f
- #define *USB_ST_EUF* 0x0c
- #define *USB_ST_HEADER* 0x00
- #define *USB_ST_MCMF* 0x0d
- #define *USB_ST_MDLM* 0x12
- #define *USB_ST_MDLMMD* 0x13
- #define *USB_ST_NCT* 0x0a
- #define *USB_ST_OBEX* 0x15
- #define *USB_ST_PUF* 0x0b
- #define *USB_ST_TCLF* 0x05
- #define *USB_ST_TCM* 0x18
- #define *USB_ST_TOMF* 0x08
- #define *USB_ST_TRF* 0x04
- #define *USB_ST_UF* 0x06
- #define *USB_ST_USBTF* 0x09
- #define *USB_ST_WHCM* 0x11

19.84.1. Подробное описание

Файл содержит описание структур дескрипторов **USB** в ОС *MULTEX-ARM*.

См. также

Общее описание работы с **USB** в главе *Подсистема USB*.

19.84.2. Макросы

19.84.2.1. BMATTRIBUTE_RESERVED

```
#define BMATTRIBUTE_RESERVED 0x80
```

19.84.2.2. BMATTRIBUTE_SELF_POWERED

```
#define BMATTRIBUTE_SELF_POWERED 0x40
```

19.84.2.3. BULK

```
#define BULK 0x02
```

19.84.2.4. CLASS_BCD_VERSION

```
#define CLASS_BCD_VERSION 0x0110
```

19.84.2.5. COMMUNICATIONS_ACM_SUBCLASS

```
#define COMMUNICATIONS_ACM_SUBCLASS 0x02
```

19.84.2.6. COMMUNICATIONS_ANCM_SUBCLASS

```
#define COMMUNICATIONS_ANCM_SUBCLASS 0x07
```

19.84.2.7. COMMUNICATIONS_CCM_SUBCLASS

```
#define COMMUNICATIONS_CCM_SUBCLASS 0x05
```

19.84.2.8. COMMUNICATIONS_DEVICE_CLASS

```
#define COMMUNICATIONS_DEVICE_CLASS 0x02
```

19.84.2.9. COMMUNICATIONS_DLCM_SUBCLASS

```
#define COMMUNICATIONS_DLCM_SUBCLASS 0x01
```

19.84.2.10. COMMUNICATIONS_DMM_SUBCLASS

```
#define COMMUNICATIONS_DMM_SUBCLASS 0x09
```

19.84.2.11. COMMUNICATIONS_ENCM_SUBCLASS

```
#define COMMUNICATIONS_ENCM_SUBCLASS 0x06
```

19.84.2.12. COMMUNICATIONS_INTERFACE_CLASS_CONTROL

```
#define COMMUNICATIONS_INTERFACE_CLASS_CONTROL 0x02
```

19.84.2.13. COMMUNICATIONS_INTERFACE_CLASS_DATA

```
#define COMMUNICATIONS_INTERFACE_CLASS_DATA 0x0A
```

19.84.2.14. COMMUNICATIONS_INTERFACE_CLASS_VENDOR

```
#define COMMUNICATIONS_INTERFACE_CLASS_VENDOR 0xFF
```

19.84.2.15. COMMUNICATIONS_MCCM_SUBCLASS

```
#define COMMUNICATIONS_MCCM_SUBCLASS 0x04
```

19.84.2.16. COMMUNICATIONS_MDLM_SUBCLASS

```
#define COMMUNICATIONS_MDLM_SUBCLASS 0x0a
```

19.84.2.17. COMMUNICATIONS_NO_PROTOCOL

```
#define COMMUNICATIONS_NO_PROTOCOL 0x00
```

19.84.2.18. COMMUNICATIONS_NO_SUBCLASS

```
#define COMMUNICATIONS_NO_SUBCLASS 0x00
```

19.84.2.19. COMMUNICATIONS_OBEX_SUBCLASS

```
#define COMMUNICATIONS_OBEX_SUBCLASS 0x0b
```

19.84.2.20. COMMUNICATIONS_TCM_SUBCLASS

```
#define COMMUNICATIONS_TCM_SUBCLASS 0x03
```

19.84.2.21. COMMUNICATIONS_V25TER_PROTOCOL

```
#define COMMUNICATIONS_V25TER_PROTOCOL 0x01 /*Common AT Hayes compatible*/
```

19.84.2.22. COMMUNICATIONS_WHCM_SUBCLASS

```
#define COMMUNICATIONS_WHCM_SUBCLASS 0x08
```

19.84.2.23. CONTROL

```
#define CONTROL 0x00
```

19.84.2.24. CS_ENDPOINT

```
#define CS_ENDPOINT 0x25
```

19.84.2.25. CS_INTERFACE

```
#define CS_INTERFACE 0x24
```

19.84.2.26. DATA_INTERFACE_CLASS

```
#define DATA_INTERFACE_CLASS 0x0a
```

19.84.2.27. DATA_INTERFACE_PROTOCOL_NONE

```
#define DATA_INTERFACE_PROTOCOL_NONE 0x00 /* No class protcol required */
```

19.84.2.28. DATA_INTERFACE_SUBCLASS_NONE

```
#define DATA_INTERFACE_SUBCLASS_NONE 0x00 /* No subclass pertinent */
```

19.84.2.29. IN

```
#define IN 0x80
```

19.84.2.30. INTERRUPT

```
#define INTERRUPT 0x03
```

19.84.2.31. ISOCHRONOUS

```
#define ISOCHRONOUS 0x01
```

19.84.2.32. OUT

```
#define OUT 0x00
```

19.84.2.33. print_device_descriptor

```
#define print_device_descriptor(  
    d )
```

19.84.2.34. USB_ST_ACMF

```
#define USB_ST_ACMF 0x02
```

19.84.2.35. USB_ST_ATMNF

```
#define USB_ST_ATMNF 0x10
```

19.84.2.36. USB_ST_CCMF

```
#define USB_ST_CCMF 0x0e
```

19.84.2.37. USB_ST_CMF

```
#define USB_ST_CMF 0x01
```

19.84.2.38. USB_ST_CS

```
#define USB_ST_CS 0x16
```

19.84.2.39. USB_ST_CSD

```
#define USB_ST_CSD 0x17
```

19.84.2.40. USB_ST_CSF

```
#define USB_ST_CSF 0x07
```

19.84.2.41. USB_ST_DLMF

```
#define USB_ST_DLMF 0x03
```

19.84.2.42. USB_ST_DMM

```
#define USB_ST_DMM 0x14
```

19.84.2.43. USB_ST_ENF

```
#define USB_ST_ENF 0x0f
```

19.84.2.44. USB_ST_EUF

```
#define USB_ST_EUF 0x0c
```

19.84.2.45. USB_ST_HEADER

```
#define USB_ST_HEADER 0x00
```

19.84.2.46. USB_ST_MCMF

```
#define USB_ST_MCMF 0x0d
```

19.84.2.47. USB_ST_MDLM

```
#define USB_ST_MDLM 0x12
```

19.84.2.48. USB_ST_MDLMD

```
#define USB_ST_MDLMD 0x13
```

19.84.2.49. USB_ST_NCT

```
#define USB_ST_NCT 0x0a
```

19.84.2.50. USB_ST_OBEX

```
#define USB_ST_OBEX 0x15
```

19.84.2.51. USB_ST_PUF

```
#define USB_ST_PUF 0x0b
```

19.84.2.52. USB_ST_TCLF

```
#define USB_ST_TCLF 0x05
```

19.84.2.53. USB_ST_TCM

```
#define USB_ST_TCM 0x18
```

19.84.2.54. USB_ST_TOMF

```
#define USB_ST_TOMF 0x08
```

19.84.2.55. USB_ST_TRF

```
#define USB_ST_TRF 0x04
```


19.84.2.56. USB_ST_UF

```
#define USB_ST_UF 0x06
```

19.84.2.57. USB_ST_USBTF

```
#define USB_ST_USBTF 0x09
```

19.84.2.58. USB_ST_WHCM

```
#define USB_ST_WHCM 0x11
```

19.85. Файл `usbman.dox`

19.86. Файл `vdisk.h`

Механизмы монтирования виртуального тома.

Функции

- `STATUS mountVDisk` (const char *devName, const char *imgFile)

19.86.1. Подробное описание

Файл содержит описание методов работы с виртуальными томами в ОС `MULTEX-ARM`.

19.86.2. Функции

19.86.2.1. `mountVDisk()`

```
STATUS mountVDisk (  
    const char * devName,  
    const char * imgFile )
```

Смонтировать виртуальный том.

Аргументы

`devName` Имя устройства, которое будет иметь смонтированный том.

`imgFile` Путь к файлу тома.

Возвращает

`OK` при успешном монтировании тома, `ERROR` иначе.



Для смонтированного тома доступны только операции чтения, но не записи.

19.87. Файл `vector.h`

Работа с массивами данных типа **Вектор**.

Структуры данных

- `struct sVector`

Функции

- `STATUS vector_Add (sVector *vPtr, const void *element)`
- `STATUS vector_AddEmpty (sVector *vPtr)`
- `STATUS vector_Copy (sVector *dstVPtr, const sVector *srcVPtr)`
- `sVector * vector_Create (size_t elementSize, size_t elementsAmount)`
- `sVector * vector_Duplicate (const sVector *vPtr)`
- `void vector_Free (sVector *vPtr)`
- `void * vector_Get (sVector *vPtr, size_t elementIndex)`
- `void * vector_GetFirst (sVector *vPtr)`
- `void * vector_GetLast (sVector *vPtr)`
- `STATUS vector_Remove (sVector *vPtr, size_t elementIndex)`
- `STATUS vector_RemoveLast (sVector *vPtr)`

19.87.1. Подробное описание

Вектор — динамический массив, обеспечивающий быстрое добавление новых элементов в конец и меняющий свой размер при необходимости.

19.87.2. Функции

19.87.2.1. `vector_Add()`

```
STATUS vector_Add (
    sVector * vPtr,
    const void * element )
```

Добавить новый элемент в вектор.

Аргументы

<code>vPtr</code>	Указатель на структуру вектора.
<code>element</code>	Адрес элемента, который должен быть добавлен в вектор.

Возвращает

ОК при успехе, ERROR иначе.



При надобности и невозможности расширения массива будет возвращен ERROR. Тем не менее, массивом можно будет продолжать пользоваться.

19.87.2.2. vector_AddEmpty()

```
STATUS vector_AddEmpty (  
    sVector * vPtr )
```

Add new element to the vector. Initialize it with zeros.

Аргументы

<i>vPtr</i>	Pointer to vector.
-------------	--------------------

Возвращает

OK if succeeded, ERROR otherwise.

19.87.2.3. vector_Copy()

```
STATUS vector_Copy (  
    sVector * dstVPtr,  
    const sVector * srcVPtr )
```

Скопировать содержимое одного вектора в другой.

Аргументы

<i>dstVPtr</i>	Вектор, в который проводится копирование.
----------------	---

<i>srcVPtr</i>	Копируемый вектор.
----------------	--------------------

Возвращает

OK при успехе, ERROR иначе.

Содержимое *dstVPtr* будет утеряно!

19.87.2.4. vector_Create()

```
sVector* vector_Create (  
    size_t elementSize,  
    size_t elementsAmount )
```

Создать вектор - динамически расширяющийся массив.

Аргументы

<i>elementSize</i>	Размер одного элемента в векторе.
--------------------	-----------------------------------

<i>elementsAmount</i>	Примерное кол-во элементов в векторе, которое нужно будет туда добавить.
-----------------------	--

Возвращает

Указатель на структуру вектора или NULL при ошибке.

19.87.2.5. `vector_Duplicate()`

```
sVector* vector_Duplicate (  
    const sVector * vPtr )
```

Создать копию вектора. Копию нужно будет освободить отдельно от исходного вектора.

Аргументы

vPtr Указатель на исходный вектор.

Возвращает

Указатель на структуру вектора или NULL при ошибке.

19.87.2.6. `vector_Free()`

```
void vector_Free (  
    sVector * vPtr )
```

Освободить вектор и хранящиеся в нем данные.

Аргументы

vPtr Указатель на структуру вектора.

19.87.2.7. `vector_Get()`

```
void* vector_Get (  
    sVector * vPtr,  
    size_t elementIndex )
```

Получить элемент из вектора.

Аргументы

vPtr Указатель на структуру вектора.

elementIndex Индекс элемента, который нужно извлечь.

Возвращает

Указатель на элемент или NULL, при ошибках.



Если будет запрошен 10 элемент из массива на 5 элементов, то функция вернет NULL.

Если в векторе хранится тип `Type`, то при отправляться туда должны `&TypeVar`, а полученные указатели кастоваться в `Type*`.

19.87.2.8. `vector_GetFirst()`

```
void* vector_GetFirst (  
    sVector * vPtr )
```

Получить первый элемент в векторе.

Аргументы

vPtr Указатель на элемент или NULL.

19.87.2.9. `vector_GetLast()`

```
void* vector_GetLast (  
    sVector * vPtr )
```

Получить последний элемент в векторе.

Аргументы

vPtr Указатель на элемент или NULL.

19.87.2.10. `vector_Remove()`

```
STATUS vector_Remove (  
    sVector * vPtr,  
    size_t elementIndex )
```

Delete element form vector (memory will not be reallocated!).

Аргументы

vPtr Pointer to vector.

elementIndex Index of target element.

Возвращает

OK if succeeded, ERROR otherwise.

19.87.2.11. vector_RemoveLast()

STATUS vector_RemoveLast (
 sVector * *vPtr*)

Удалить последний добавленный элемент из вектора (размер при этом НЕ будет пересчитан в меньшую сторону).

Аргументы

vPtr Указатель на структуру вектора.

Возвращает

OK при успехе, ERROR иначе.

Предметный указатель

- `_Exit`
 - `stdlib.h`, 654
- `_IOFBF`
 - `stdio.h`, 632
- `_IOLBF`
 - `stdio.h`, 632
- `_IONBF`
 - `stdio.h`, 632
- `_IS_BLN`
 - `ctype.h`, 313
- `_IS_CTL`
 - `ctype.h`, 313
- `_IS_DIG`
 - `ctype.h`, 313
- `_IS_HEX`
 - `ctype.h`, 314
- `_IS_LOW`
 - `ctype.h`, 314
- `_IS_PUN`
 - `ctype.h`, 314
- `_IS_SP`
 - `ctype.h`, 314
- `_IS_UPP`
 - `ctype.h`, 314
- `_MULTEX_`
 - `multex.h`, 511
- `__ALIGN_MASK`
 - `multex.h`, 511
- `__ASSERT_NOOP`
 - `assert.h`, 265
- `__LITTLE_ENDIAN`
 - `multex.h`, 511
 - `usb.h`, 760
- `__assert_fail`
 - `assert.h`, 265
- `__be32_to_cpu`
 - `multex.h`, 511
- `__bool_true_false_are_defined`
 - `stdbool.h`, 615
- `__free`
 - `memlib.h`, 495
- `__swap_16`
 - `usb.h`, 760
- `__swap_32`
 - `usb.h`, 761
- `__typeof`
 - `stddef.h`, 617
- `_findSTName`
 - `names.h`, 521
- `_tolower`
 - `ctype.h`, 314
- `_toupper`
 - `ctype.h`, 314
- `a20graph.h`, 226
 - `ACONSTR`, 229
 - `ASCREEN`, 230
 - `bitBlt`, 240
 - `CONSTR`, 230
 - `deleteSurface`, 241
 - `deleteSurfaceDataArray`, 241
 - `Display`, 251
 - `fillRect`, 241
 - `FlipScreenAndConstr`, 242
 - `G2D_BLT_DST_COLORKEY`, 233
 - `g2d_blt_flags`, 232
 - `G2D_BLT_FLIP_HORIZONTAL`, 233
 - `G2D_BLT_FLIP_VERTICAL`, 233
 - `G2D_BLT_MIRROR135`, 233
 - `G2D_BLT_MIRROR45`, 233
 - `G2D_BLT_MULTI_ALPHA`, 232
 - `G2D_BLT_NONE`, 232
 - `G2D_BLT_PIXEL_ALPHA`, 232
 - `G2D_BLT_PLANE_ALPHA`, 232
 - `G2D_BLT_ROTATE180`, 233
 - `G2D_BLT_ROTATE270`, 233
 - `G2D_BLT_ROTATE90`, 233
 - `G2D_BLT_SRC_COLORKEY`, 233
 - `g2d_data_fmt`, 234
 - `G2D_FIL_MULTI_ALPHA`, 236
 - `G2D_FIL_NONE`, 236
 - `G2D_FIL_PIXEL_ALPHA`, 236
 - `G2D_FIL_PLANE_ALPHA`, 236
 - `g2d_fillrect_flags`, 236
 - `G2D_FMT_1BPP_MONO`, 235
 - `G2D_FMT_1BPP_PALETTE`, 235
 - `G2D_FMT_2BPP_MONO`, 235
 - `G2D_FMT_2BPP_PALETTE`, 235
 - `G2D_FMT_4BPP_MONO`, 235
 - `G2D_FMT_4BPP_PALETTE`, 235
 - `G2D_FMT_8BPP_MONO`, 235
 - `G2D_FMT_8BPP_PALETTE`, 235
 - `G2D_FMT_ABGR1555`, 234
 - `G2D_FMT_ABGR4444`, 234
 - `G2D_FMT_ABGR8888`, 234
 - `G2D_FMT_ABGR_AVUY8888`, 234
 - `G2D_FMT_ARGB1555`, 234
 - `G2D_FMT_ARGB4444`, 234
 - `G2D_FMT_ARGB8888`, 234
 - `G2D_FMT_ARGB_AYUV8888`, 234
 - `G2D_FMT_BGR565`, 235
 - `G2D_FMT_BGRA4444`, 234
 - `G2D_FMT_BGRA5551`, 234
 - `G2D_FMT_BGRA8888`, 234
 - `G2D_FMT_BGRA_VUYA8888`, 234
 - `G2D_FMT_BGRX8888`, 234
 - `G2D_FMT_IYUV422`, 235
 - `G2D_FMT_PYUV411`, 235

G2D_FMT_PYUV411UVC, 235
 G2D_FMT_PYUV420, 235
 G2D_FMT_PYUV420UVC, 235
 G2D_FMT_PYUV422, 235
 G2D_FMT_PYUV422UVC, 235
 G2D_FMT_RGB565, 234
 G2D_FMT_RGBA4444, 234
 G2D_FMT_RGBA5551, 234
 G2D_FMT_RGBA8888, 234
 G2D_FMT_RGBA_YUVA8888, 234
 G2D_FMT_RGBX8888, 234
 G2D_FMT_XBGR8888, 234
 G2D_FMT_XRGB8888, 234
 g2d_pixel_seq, 237
 G2D_SEQ_1BPP_BIG_BIG, 237
 G2D_SEQ_1BPP_BIG_LITTER, 237
 G2D_SEQ_1BPP_LITTER_BIG, 237
 G2D_SEQ_1BPP_LITTER_LITTER, 237
 G2D_SEQ_2BPP_BIG_BIG, 237
 G2D_SEQ_2BPP_BIG_LITTER, 237
 G2D_SEQ_2BPP_LITTER_BIG, 237
 G2D_SEQ_2BPP_LITTER_LITTER, 237
 G2D_SEQ_NORMAL, 237
 G2D_SEQ_P01, 237
 G2D_SEQ_P0123, 237
 G2D_SEQ_P01234567, 237
 G2D_SEQ_P10, 237
 G2D_SEQ_P10325476, 237
 G2D_SEQ_P3210, 237
 G2D_SEQ_P67452301, 237
 G2D_SEQ_P76543210, 237
 G2D_SEQ_VUVU, 237
 G2D_SEQ_VYUY, 237
 G2D_SEQ_YVYU, 237
 getConstr2Buffer, 242
 getConstrAlpSurface, 242
 getConstrBuffer, 243
 getConstrSurface, 243
 getLFB, 243
 getScreen2Buffer, 243
 getScreenAlpSurface, 243
 getScreenBitsPerPixel, 244
 getScreenBuffer, 244
 getScreenHeight, 244
 getScreenPitch, 244
 getScreenSurface, 244
 getScreenWidth, 245
 HDC, 232
 init_2D_engine, 245
 initLvdsDisplay, 245
 loadBMPSurface, 245
 loadJPEGSurface, 246
 loadPNGSurface, 246
 loadRawSurface, 247
 LVDS_1024x768, 238
 LVDS_1280x800, 238
 LVDS_1400x1050, 239
 LVDS_158x1920, 239
 LVDS_1920x1080, 238
 LVDS_1920x165, 239
 LVDS_1920x360, 238
 LVDS_800x480, 238
 LVDS_800x600, 238
 lvds_param_t, 238
 MODE_1024x768, 239
 MODE_1280x1024, 239
 MODE_1280x720, 239
 MODE_1280x768, 239
 MODE_1280x800, 239
 MODE_1368x768, 239
 MODE_1920x1080, 239
 MODE_640x480, 239
 MODE_800x480, 239
 MODE_800x600, 239
 MODE_TV, 239
 newSurface, 247
 OT_INRGB888_PLANAR, 230
 OT_MB_INTERLEAVED, 230
 OT_MB_PLANAR, 230
 OT_MB_UV_COMBINED, 230
 OT_PLANAR, 230
 OT_UV_COMBINED, 230
 OT_YUV420_COMBINED, 230
 OT_YUV420_INTERLEAVED, 231
 OT_YUV420_MB_COMBINED, 231
 OT_YUV420_MB_PLANAR, 231
 OT_YUV420_PLANAR, 231
 OT_YUV422_COMBINED, 231
 OT_YUV422_INTERLEAVED, 231
 OT_YUV422_MB_COMBINED, 231
 OT_YUV422_MB_PLANAR, 231
 OT_YUV422_PLANAR, 231
 OverlayClose, 248
 OverlayInit, 248
 OverlayOpen, 248
 OverlayOpenDI, 248
 OverlaySetAddr, 249
 SCREEN, 232
 set_lvds_mode, 249
 set_lvds_param, 249
 setOverlayPriority, 250
 setVideoMode, 250
 stretchBlit, 250
 surface_t, 232
 VideoModes, 239
 waitVerticalRetrace, 251
 abort
 stdlib.h, 655
 abs
 stdlib.h, 655
 abstract_control
 usb_class_descriptor, 183
 accept
 socket.h, 576

ACONSTR
 a20graph.h, 229
 acos
 math.h, 488
 act_altsetting
 usb_interface, 222
 act_len
 usb_device, 211
 addr
 g2d_image, 129
 tDrvBit, 164
 tDrvBitGroup, 165
 AF_INET
 socket.h, 574
 AHB_1
 pll.h, 526
 AHB_2
 pll.h, 526
 ALIGN
 multex.h, 511
 alignHCenter
 fontdefines.h, 377
 alignHLeft
 fontdefines.h, 377
 alignHRight
 fontdefines.h, 377
 alignType
 fontdefines.h, 377
 alignVBottom
 fontdefines.h, 378
 alignVMiddle
 fontdefines.h, 378
 alignVTop
 fontdefines.h, 378
 ALLOC_ALIGN_BUFFER
 multex.h, 512
 ALLOC_CACHE_ALIGN_BUFFER
 multex.h, 512
 alpha
 g2d_blt, 126
 g2d_fillrect, 128
 g2d_stretchblt, 131
 ALT_B
 inputstr.h, 410
 ALT_D
 inputstr.h, 410
 ALT_F
 inputstr.h, 410
 and
 iso646.h, 466
 and_eq
 iso646.h, 466
 APB_1
 pll.h, 526
 APB_2
 pll.h, 526
 appendSymbol
 names.h, 521
 arch.h, 253
 archAddInt, 254
 archAddIntArray, 254
 archAddString, 255
 archCheckString, 255
 archFree, 256
 archGetGpio, 256
 archGetInt, 256
 archGetIntArray, 256
 archGetString, 257
 archPrint, 257
 drvGetBit, 257
 drvGetBitGroup, 258
 drvInitBit, 258
 drvInitBitGroup, 258
 drvInitGpio, 259
 drvResetBit, 259
 drvSetBit, 260
 drvSetBitGroup, 260
 drvSetGpio, 260
 ARCH_BACKLIGHT_FREQUENCY
 archdef.h, 262
 ARCH_BACKLIGHT_GPIO
 archdef.h, 263
 ARCH_BOARD
 archdef.h, 263
 ARCH_BOARD_SE8350_00
 archdef.h, 263
 ARCH_BOARD_SE8351_00
 archdef.h, 263
 ARCH_BOARD_SE_DB_A20_B254
 archdef.h, 263
 ARCH_CPU
 archdef.h, 263
 ARCH_DMA_MINALIGN
 multex.h, 512
 usb.h, 761
 ARCH_LED_DISK
 archdef.h, 263
 ARCH_LED_SYSTEM
 archdef.h, 264
 ARCH_LEDLINE_PWM
 archdef.h, 264
 ARCH_LEDLINE_TIMER
 archdef.h, 264
 ARCH_PROC_A20
 archdef.h, 264
 ARCH_PROC_A40
 archdef.h, 264
 ARCH_PROC_H3
 archdef.h, 264
 ARCH_PROC_V3S
 archdef.h, 264
 archAddInt
 arch.h, 254
 archAddIntArray

- arch.h, 254
- archAddString
 - arch.h, 255
- archCheckString
 - arch.h, 255
- archdef.h, 262
 - ARCH_BACKLIGHT_FREQUENCY, 262
 - ARCH_BACKLIGHT_GPIO, 263
 - ARCH_BOARD, 263
 - ARCH_BOARD_SE8350_00, 263
 - ARCH_BOARD_SE8351_00, 263
 - ARCH_BOARD_SE_DB_A20_B254, 263
 - ARCH_CPU, 263
 - ARCH_LED_DISK, 263
 - ARCH_LED_SYSTEM, 264
 - ARCH_LEDLINE_PWM, 264
 - ARCH_LEDLINE_TIMER, 264
 - ARCH_PROC_A20, 264
 - ARCH_PROC_A40, 264
 - ARCH_PROC_H3, 264
 - ARCH_PROC_V3S, 264
- archFree
 - arch.h, 256
- archGetGpio
 - arch.h, 256
- archGetInt
 - arch.h, 256
- archGetIntArray
 - arch.h, 256
- archGetString
 - arch.h, 257
- archPrint
 - arch.h, 257
- arg
 - blk_dev, 108
- Array
 - iniBinaryArray, 134
 - iniIntArray, 136
- ARRAY_INDEX_TO_SYMBOL
 - fontsdefines.h, 376
- ARRAY_SIZE
 - multex.h, 512
- ArrayLength
 - iniBinaryArray, 134
 - iniIntArray, 136
- ASCII_PRINTED_SYMBOLS_AMOUNT
 - fontsdefines.h, 376
- ASCREEN
 - a20graph.h, 230
- asctime
 - time.h, 709
- asctime_r
 - time.h, 709
- asin
 - math.h, 488
- assert
 - assert.h, 265
- assert.h, 265
 - __ASSERT_NOOP, 265
 - __assert_fail, 265
 - assert, 265
 - static_assert, 265
- atan
 - math.h, 488
- atan2
 - math.h, 488
- atan2f
 - math.h, 488
- atanf
 - math.h, 489
- atexit
 - stdlib.h, 655
- atm_networking
 - usb_class_descriptor, 183
- atof
 - stdlib.h, 655
- atoi
 - stdlib.h, 656
- atol
 - stdlib.h, 656
- available
 - tDrvGpio, 166
- avi.dox, 267
- AVI_HANDLE
 - avilib.h, 269
- AVI_RETURN_EOF
 - avilib.h, 269
- AVI_RETURN_ERROR
 - avilib.h, 269
- avilib.h, 268
 - AVI_HANDLE, 269
 - AVI_RETURN_EOF, 269
 - AVI_RETURN_ERROR, 269
 - closeAVIFile, 269
 - newAVIFile, 269
 - newAVIFileSnd, 270
 - openAVIFile, 270
 - readAVIAudio, 271
 - readAVIFrame, 271
 - seekToFirstVideoFrame, 271
 - setAVIType, 272
 - writeAVIFrame, 272
 - writeSNDFrame, 273
- bAlternateSetting
 - usb_interface_descriptor, 223
- bcd2d
 - stdlib.h, 656
- bcdCDC
 - usb_class_header_function_descriptor, 192
 - usb_class_hid_descriptor, 193
- bcdDevice
 - usb_device_descriptor, 215
- bcdUSB

- usb_device_descriptor, 215
- bcdVersion
 - usb_class_mdldm_descriptor, 194
- bChannelIndex
 - usb_class_network_channel_descriptor, 197
- bChild0
 - usb_class_extension_unit_descriptor, 189
 - usb_class_protocol_unit_function_descriptor, 198
 - usb_class_usb_terminal_descriptor, 204
- bConfigurationValue
 - usb_configuration_descriptor, 207
- bCountryCode
 - usb_class_hid_descriptor, 193
- bd_blkTotal
 - blk_dev, 108
- bd_bytesPerBlk
 - blk_dev, 108
- bd_catSaved
 - blk_dev, 108
- bd_rootCat
 - blk_dev, 108
- bd_signature
 - blk_dev, 108
- bd_startBlk
 - blk_dev, 108
- BD_STD_SIGNATURE
 - iolib.h, 441
- bd_volume
 - blk_dev, 109
- bData
 - usb_class_report_descriptor, 199
- bDataInterface
 - usb_class_call_management_descriptor, 180
- bDescriptorSubtype
 - usb_class_abstract_control_descriptor, 177
 - usb_class_atm_networking_descriptor, 178
 - usb_class_call_management_descriptor, 180
 - usb_class_capi_control_descriptor, 181
 - usb_class_country_selection_descriptor, 182
 - usb_class_direct_line_descriptor, 186
 - usb_class_ethernet_networking_descriptor, 187
 - usb_class_extension_unit_descriptor, 189
 - usb_class_function_descriptor, 190
 - usb_class_function_descriptor_generic, 191
 - usb_class_header_function_descriptor, 192
 - usb_class_mdldm_descriptor, 194
 - usb_class_mdldmd_descriptor, 195
 - usb_class_multi_channel_descriptor, 196
 - usb_class_network_channel_descriptor, 197
 - usb_class_protocol_unit_function_descriptor, 198
 - usb_class_telephone_call_descriptor, 200
 - usb_class_telephone_operational_descriptor, 201
 - usb_class_telephone_ringer_descriptor, 202
 - usb_class_union_function_descriptor, 203
 - usb_class_usb_terminal_descriptor, 204
 - usb_generic_descriptor, 221
- bDescriptorType
- usb_class_abstract_control_descriptor, 177
- usb_class_atm_networking_descriptor, 178
- usb_class_call_management_descriptor, 180
- usb_class_capi_control_descriptor, 181
- usb_class_country_selection_descriptor, 182
- usb_class_direct_line_descriptor, 186
- usb_class_ethernet_networking_descriptor, 187
- usb_class_extension_unit_descriptor, 189
- usb_class_function_descriptor, 190
- usb_class_function_descriptor_generic, 191
- usb_class_header_function_descriptor, 192
- usb_class_hid_descriptor, 193
- usb_class_mdldm_descriptor, 194
- usb_class_mdldmd_descriptor, 195
- usb_class_multi_channel_descriptor, 196
- usb_class_network_channel_descriptor, 197
- usb_class_protocol_unit_function_descriptor, 198
- usb_class_report_descriptor, 199
- usb_class_telephone_call_descriptor, 200
- usb_class_telephone_operational_descriptor, 201
- usb_class_telephone_ringer_descriptor, 202
- usb_class_union_function_descriptor, 203
- usb_class_usb_terminal_descriptor, 204
- usb_configuration_descriptor, 207
- usb_device_descriptor, 215
- usb_endpoint_descriptor, 218
- usb_generic_descriptor, 221
- usb_interface_descriptor, 223
- usb_string_descriptor, 225
- bDescriptorType0
 - usb_class_hid_descriptor, 193
- bDetailData
 - usb_class_mdldmd_descriptor, 195
- bDeviceClass
 - usb_device_descriptor, 215
- bDeviceProtocol
 - usb_device_descriptor, 215
- bDeviceSubClass
 - usb_device_descriptor, 216
- bEndpointAddress
 - usb_endpoint_descriptor, 218
- bEntityId
 - usb_class_extension_unit_descriptor, 189
 - usb_class_network_channel_descriptor, 197
 - usb_class_protocol_unit_function_descriptor, 198
 - usb_class_usb_terminal_descriptor, 204
- bExtensionCode
 - usb_class_extension_unit_descriptor, 189
- bFunctionLength
 - usb_class_abstract_control_descriptor, 177
 - usb_class_atm_networking_descriptor, 178
 - usb_class_call_management_descriptor, 180
 - usb_class_capi_control_descriptor, 181
 - usb_class_country_selection_descriptor, 182
 - usb_class_direct_line_descriptor, 186
 - usb_class_ethernet_networking_descriptor, 187
 - usb_class_extension_unit_descriptor, 189

- usb_class_function_descriptor, 190
- usb_class_function_descriptor_generic, 191
- usb_class_header_function_descriptor, 192
- usb_class_mdIm_descriptor, 194
- usb_class_mdImd_descriptor, 195
- usb_class_multi_channel_descriptor, 196
- usb_class_network_channel_descriptor, 197
- usb_class_protocol_unit_function_descriptor, 198
- usb_class_telephone_call_descriptor, 200
- usb_class_telephone_operational_descriptor, 201
- usb_class_telephone_ringer_descriptor, 202
- usb_class_union_function_descriptor, 203
- usb_class_usb_terminal_descriptor, 204
- bgBrightBlack
 - crt.h, 295
- bgBrightBlue
 - crt.h, 295
- bgBrightCyan
 - crt.h, 295
- bgBrightGreen
 - crt.h, 295
- bgBrightMagenta
 - crt.h, 295
- bgBrightRed
 - crt.h, 295
- bgBrightWhite
 - crt.h, 295
- bgBrightYellow
 - crt.h, 295
- bGUID
 - usb_class_mdIm_descriptor, 194
- bGuidDescriptorType
 - usb_class_mdImd_descriptor, 195
- bind
 - socket.h, 577
- bInterfaceClass
 - usb_interface_descriptor, 223
- bInterfaceNo
 - usb_class_usb_terminal_descriptor, 204
- bInterfaceNumber
 - usb_interface_descriptor, 223
- bInterfaceProtocol
 - usb_interface_descriptor, 223
- bInterfaceSubClass
 - usb_interface_descriptor, 224
- bInterval
 - usb_endpoint_descriptor, 218
- bitand
 - iso646.h, 466
- bitBlt
 - a20graph.h, 240
- bitor
 - iso646.h, 466
- bLength
 - usb_class_hid_descriptor, 193
 - usb_class_report_descriptor, 199
 - usb_configuration_descriptor, 207
- usb_device_descriptor, 216
- usb_endpoint_descriptor, 218
- usb_generic_descriptor, 221
- usb_interface_descriptor, 224
- usb_string_descriptor, 225
- blk_cache, 106
 - BlkSize, 106
 - BlkStat, 106
 - Cache, 106
 - Cache_na, 106
 - CacheIdx, 106
 - cacheOff, 106
 - CacheSize, 107
 - hDrv, 107
 - read, 107
 - write, 107
- blk_cache_callback
 - blkcache.h, 274
- blk_cache_create
 - blkcache.h, 275
- blk_cache_flush
 - blkcache.h, 275
- blk_cache_free
 - blkcache.h, 275
- blk_cache_load
 - blkcache.h, 276
- blk_cache_proc
 - blkcache.h, 274
- blk_cache_read
 - blkcache.h, 276
- blk_cache_write
 - blkcache.h, 277
- BLK_DEV
 - iolib.h, 445
- blk_dev, 108
 - arg, 108
 - bd_blkTotal, 108
 - bd_bytesPerBlk, 108
 - bd_catSaved, 108
 - bd_rootCat, 108
 - bd_signature, 108
 - bd_startBlk, 108
 - bd_volume, 109
 - fsData, 109
 - funcCode, 109
 - numBlks, 109
 - pBuf, 109
 - pDev, 109
 - startBlk, 109
 - volConfig, 109
- blkBuf
 - file_fcb, 124
- blkcache.h, 274
 - blk_cache_callback, 274
 - blk_cache_create, 275
 - blk_cache_flush, 275
 - blk_cache_free, 275

- blk_cache_load, 276
- blk_cache_proc, 274
- blk_cache_read, 276
- blk_cache_write, 277
- blkNum
 - file_fcb, 124
- BlkSize
 - blk_cache, 106
- BlkStat
 - blk_cache, 106
- bMasterInterface
 - usb_class_union_function_descriptor, 203
- bmATMDeviceStatistics
 - usb_class_atm_networking_descriptor, 178
- BMATTRIBUTE_RESERVED
 - usbdescriptors.h, 777
- BMATTRIBUTE_SELF_POWERED
 - usbdescriptors.h, 777
- bmAttributes
 - usb_configuration_descriptor, 207
 - usb_endpoint_descriptor, 218
- bMaxPacketSize0
 - usb_device_descriptor, 216
- bMaxPower
 - usb_configuration_descriptor, 208
- bmCapabilities
 - usb_class_abstract_control_descriptor, 177
 - usb_class_call_management_descriptor, 180
 - usb_class_capi_control_descriptor, 181
 - usb_class_function_descriptor_generic, 191
 - usb_class_multi_channel_descriptor, 196
 - usb_class_telephone_call_descriptor, 200
 - usb_class_telephone_operational_descriptor, 201
- bmDataCapabilities
 - usb_class_atm_networking_descriptor, 178
- bmEthernetStatistics
 - usb_class_ethernet_networking_descriptor, 187
- bmOptions
 - usb_class_usb_terminal_descriptor, 204
- bNumberPowerFilters
 - usb_class_ethernet_networking_descriptor, 187
- bNumConfigurations
 - usb_device_descriptor, 216
- bNumDescriptors
 - usb_class_hid_descriptor, 193
- bNumEndpoints
 - usb_interface_descriptor, 224
- bNumInterfaces
 - usb_configuration_descriptor, 208
- bNumRingerPatterns
 - usb_class_telephone_ringer_descriptor, 202
- bool
 - stdbool.h, 615
- bOutInterfaceNo
 - usb_class_usb_terminal_descriptor, 204
- bPhysicalInterface
 - usb_class_network_channel_descriptor, 197
- BPP
 - Display, 114
- bpp
 - sDisplayInfo, 146
 - tScreenDeviceMode, 173
- bProtocol
 - usb_class_protocol_unit_function_descriptor, 198
- bRingerVolSeps
 - usb_class_telephone_ringer_descriptor, 202
- bsearch
 - stdlib.h, 657
- bSlaveInterface0
 - usb_class_union_function_descriptor, 203
- BUFSIZ
 - stdio.h, 632
- buildFileName
 - fnames.h, 358
- BULK
 - usbdescriptors.h, 778
- bzero
 - string.h, 671
- cabs
 - math.h, 489
- Cache
 - blk_cache, 106
- cache.h, 278
 - dcache_disable, 278
 - dcache_enable, 278
 - dcache_status, 278
 - flush_dcache_all, 279
 - flush_dcache_range, 279
 - icache_status, 279
 - invalidate_dcache_all, 279
 - invalidate_dcache_range, 279
 - invalidate_icache_all, 280
- Cache_na
 - blk_cache, 106
- CacheIdx
 - blk_cache, 106
- cacheOff
 - blk_cache, 106
- CacheSize
 - blk_cache, 107
- call_management
 - usb_class_descriptor, 183
- calloc
 - memlib.h, 496
- capi_control
 - usb_class_descriptor, 183
- catIndex
 - file_fcb, 124
- cbrt
 - math.h, 489
- cbtrf
 - math.h, 489
- cd

- ioolib.h, 447
- cedrus.h, 281
 - color_format, 282
 - h264_decode, 282
 - h264_decode_part, 282
 - h264_decoder_free, 283
 - h264_decoder_init, 283
 - h264_decoder_set_mk, 284
 - h264_decoder_set_qp, 284
 - h264_decoder_set_wm, 284
 - h264_encode, 285
 - h264_encoder_free, 285
 - h264_encoder_init, 285
 - h264_encoder_set_qp, 286
 - H264_FMT_NV12, 282
 - H264_FMT_NV16, 282
 - h264_get_out, 286
 - h264_is_keyframe, 286
 - h264_set_src_format, 287
 - ve_close, 287
 - ve_open, 287
- ceil
 - math.h, 489
- ceilf
 - math.h, 489
- char16_t
 - uchar.h, 745
- char32_t
 - uchar.h, 745
- CHAR_BIT
 - limits.h, 470
- CHAR_MAX
 - limits.h, 470
- CHAR_MIN
 - limits.h, 470
- charToHex
 - string.h, 671
- child
 - TCB, 160
- children
 - usb_device, 211
- childstatus
 - TCB, 160
- chkDsk
 - ioolib.h, 447
- clamp
 - multex.h, 512
- CLASS_BCD_VERSION
 - usbdescriptors.h, 778
- clBlack
 - crt.h, 296
- clBlue
 - crt.h, 296
- clBrightBlack
 - crt.h, 296
- clBrightBlue
 - crt.h, 296
- clBrightCyan
 - crt.h, 296
- clBrightGreen
 - crt.h, 296
- clBrightMagenta
 - crt.h, 296
- clBrightRed
 - crt.h, 296
- clBrightWhite
 - crt.h, 296
- clBrightYellow
 - crt.h, 297
- clCyan
 - crt.h, 297
- clearerr
 - stdio.h, 633
- clearSurface
 - softgraph.h, 584
- clGreen
 - crt.h, 297
- clMagenta
 - crt.h, 297
- clNone
 - crt.h, 297
- clock
 - time.h, 710
- clock_t
 - time.h, 709
- CLOCKS_PER_SEC
 - time.h, 708
- close
 - ioolib.h, 448
- closeAVIFile
 - avilib.h, 269
- clRed
 - crt.h, 297
- clrscr
 - crt.h, 298
- clWhite
 - crt.h, 297
- clYellow
 - crt.h, 297
- color
 - g2d_blt, 126
 - g2d_fillrect, 128
 - g2d_stretchblt, 131
- color_format
 - cedrus.h, 282
- com_port
 - uart.h, 734
- COMMUNICATIONS_ACM_SUBCLASS
 - usbdescriptors.h, 778
- COMMUNICATIONS_ANCM_SUBCLASS
 - usbdescriptors.h, 778
- COMMUNICATIONS_CCM_SUBCLASS
 - usbdescriptors.h, 778
- COMMUNICATIONS_DEVICE_CLASS

- usbdescriptors.h, 778
- COMMUNICATIONS_DLCM_SUBCLASS
 - usbdescriptors.h, 778
- COMMUNICATIONS_DMM_SUBCLASS
 - usbdescriptors.h, 778
- COMMUNICATIONS_ENCM_SUBCLASS
 - usbdescriptors.h, 778
- COMMUNICATIONS_INTERFACE_CLASS_CONTROL
 - usbdescriptors.h, 779
- COMMUNICATIONS_INTERFACE_CLASS_DATA
 - usbdescriptors.h, 779
- COMMUNICATIONS_INTERFACE_CLASS_VENDOR
 - usbdescriptors.h, 779
- COMMUNICATIONS_MCCM_SUBCLASS
 - usbdescriptors.h, 779
- COMMUNICATIONS_MDLM_SUBCLASS
 - usbdescriptors.h, 779
- COMMUNICATIONS_NO_PROTOCOL
 - usbdescriptors.h, 779
- COMMUNICATIONS_NO_SUBCLASS
 - usbdescriptors.h, 779
- COMMUNICATIONS_OBEX_SUBCLASS
 - usbdescriptors.h, 779
- COMMUNICATIONS_TCM_SUBCLASS
 - usbdescriptors.h, 779
- COMMUNICATIONS_V25TER_PROTOCOL
 - usbdescriptors.h, 780
- COMMUNICATIONS_WHCM_SUBCLASS
 - usbdescriptors.h, 780
- compareNames
 - fnames.h, 358
- compareStr
 - stdlib.h, 657
- compl
 - iso646.h, 466
- complex, 110
 - x, 110
 - y, 110
- config
 - usb_device, 211
- CONFIG_USB_EHCI
 - usb.h, 761
- configno
 - usb_device, 211
- configuration
 - usb_descriptor, 209
- connect
 - socket.h, 577
- console.h, 288
 - get_c, 288
 - geth, 288
 - putb, 288
 - puti, 289
 - putl, 289
 - putw, 289
 - strtoh, 289
- CONSTR
 - a20graph.h, 230
 - Constr
 - Display, 114
 - CONTROL
 - usbdescriptors.h, 780
 - convert_AsciiToUtf16
 - unicode.h, 752
 - convert_Cp1251ToUtf16
 - unicode.h, 752
 - convert_Cp1251ToUtf8
 - unicode.h, 753
 - convert_Utf16ToAscii
 - unicode.h, 753
 - convert_Utf16ToCp1251
 - unicode.h, 754
 - convert_Utf16ToUtf8
 - unicode.h, 754
 - convert_Utf8ToCp1251
 - unicode.h, 755
 - convert_Utf8ToUtf16
 - unicode.h, 755
 - convertDateTimeTime
 - datetime.h, 321
 - copy
 - filesystem.h, 350
 - copyFile
 - filesystem.h, 350
 - cos
 - math.h, 489
 - cosf
 - math.h, 489
 - cosh
 - math.h, 490
 - count
 - msgQID, 142
 - Sem_Id, 148
 - country_selection
 - usb_class_descriptor, 183
 - cpuUsage
 - tasklib.h, 697
 - crc32.h, 291
 - crc32_compute, 291
 - crc32_compute
 - crc32.h, 291
 - crc8.h, 292
 - crc8_1step, 292
 - crc8_compute, 292
 - crc8_1step
 - crc8.h, 292
 - crc8_compute
 - crc8.h, 292
 - creat
 - iolib.h, 448
 - creat_empty
 - iolib.h, 448
 - create_pipe
 - usb.h, 761

- createDynSymTbl
 - names.h, 522
- createScreenSurface
 - softgraph.h, 584
- createSHdr
 - softgraph.h, 585
- createSurface
 - softgraph.h, 585
- crt.h, 294
 - bgBrightBlack, 295
 - bgBrightBlue, 295
 - bgBrightCyan, 295
 - bgBrightGreen, 295
 - bgBrightMagenta, 295
 - bgBrightRed, 295
 - bgBrightWhite, 295
 - bgBrightYellow, 295
 - clBlack, 296
 - clBlue, 296
 - clBrightBlack, 296
 - clBrightBlue, 296
 - clBrightCyan, 296
 - clBrightGreen, 296
 - clBrightMagenta, 296
 - clBrightRed, 296
 - clBrightWhite, 296
 - clBrightYellow, 297
 - clCyan, 297
 - clGreen, 297
 - clMagenta, 297
 - clNone, 297
 - clRed, 297
 - clrscr, 298
 - clWhite, 297
 - clYellow, 297
 - CSI, 298
 - cursorMove, 298
 - cursorOff, 299
 - cursorOn, 299
 - cursorRestore, 299
 - cursorStore, 299
 - keyPressed, 299
 - nosound, 299
 - readKey, 300
 - readKey_Timeout, 300
 - sound, 301
 - soundt, 301
 - taBgLight, 297
 - taInverse, 298
 - taLight, 298
 - taNormal, 298
 - taUnderlined, 298
 - textAttr, 302
 - textBackground, 302
 - textColor, 302
- CS_ENDPOINT
 - usbdescriptors.h, 780
- CS_INTERFACE
 - usbdescriptors.h, 780
- CSI
 - crt.h, 298
- csi.h, 303
 - CSI_0, 304
 - CSI_1, 304
 - csiActiveHigh, 308
 - csiActiveLow, 308
 - csiDevCreate, 309
 - csiDevDelete, 309
 - csiFallingEdge, 305
 - csiField_either, 305
 - csiField_even, 305
 - csiField_odd, 305
 - csiFlush, 309
 - csiGetChroma, 310
 - csiGetLuma, 310
 - csiInFmt_CCIR656_1ch, 305
 - csiInFmt_CCIR656_2ch, 306
 - csiInFmt_CCIR656_4ch, 306
 - csiInFmt_RAW, 305
 - csiInFmt_YUV422, 305
 - csiInFmt_YUV422_16bit, 306
 - csiOutFmt_PathThrough, 306
 - csiOutFmt_YCbCr_420_fldp, 306
 - csiOutFmt_YCbCr_420_fldp_UVc, 306
 - csiOutFmt_YCbCr_420_fldt, 306
 - csiOutFmt_YCbCr_420_frp, 306
 - csiOutFmt_YCbCr_420_frp_UVc, 306
 - csiOutFmt_YCbCr_420_frt, 307
 - csiOutFmt_YCbCr_422_fldp, 306
 - csiOutFmt_YCbCr_422_fldp_UVc, 306
 - csiOutFmt_YCbCr_422_fldt, 306
 - csiOutFmt_YCbCr_422_frp, 306
 - csiOutFmt_YCbCr_422_frp_UVc, 306
 - csiOutFmt_YCbCr_422_frt, 307
 - csiOutFmt_YCbCr_422_frt_UVc, 306
 - csiOutFmt_YCbCr_422_inter, 306
 - csiOutFmt_YUV_420_p, 307
 - csiOutFmt_YUV_420_pc, 307
 - csiOutFmt_YUV_420_t, 307
 - csiOutFmt_YUV_422_p, 307
 - csiOutFmt_YUV_422_pc, 307
 - csiOutFmt_YUV_422_t, 307
 - csiRisingEdge, 305
 - csiRun, 310
 - csiSequence_UYVY, 308
 - csiSequence_VYUY, 308
 - csiSequence_YUYV, 308
 - csiSequence_YVYU, 308
 - csiSetMode, 311
 - csiSetPhase, 311
 - csiSetSeq, 311
 - csiStop, 312
 - csiWaitFrame, 312
 - eCsiClockEdge, 304
 - eCsiFieldSelection, 305

eCsiInputFormat, [305](#)
 eCsiOutputFormat, [306](#)
 eCsiPolarity, [307](#)
 eSciInputSequence, [308](#)
 sciSequence_BGBG, [308](#)
 sciSequence_GBGB, [308](#)
 sciSequence_GRGR, [308](#)
 sciSequence_RGRG, [308](#)
 CSI_0
 csi.h, [304](#)
 CSI_1
 csi.h, [304](#)
 csiActiveHigh
 csi.h, [308](#)
 csiActiveLow
 csi.h, [308](#)
 csiDevCreate
 csi.h, [309](#)
 csiDevDelete
 csi.h, [309](#)
 csiFallingEdge
 csi.h, [305](#)
 csiField_either
 csi.h, [305](#)
 csiField_even
 csi.h, [305](#)
 csiField_odd
 csi.h, [305](#)
 csiFlush
 csi.h, [309](#)
 csiGetChroma
 csi.h, [310](#)
 csiGetLuma
 csi.h, [310](#)
 csiInFmt_CCIR656_1ch
 csi.h, [305](#)
 csiInFmt_CCIR656_2ch
 csi.h, [306](#)
 csiInFmt_CCIR656_4ch
 csi.h, [306](#)
 csiInFmt_RAW
 csi.h, [305](#)
 csiInFmt_YUV422
 csi.h, [305](#)
 csiInFmt_YUV422_16bit
 csi.h, [306](#)
 csiOutFmt_PathThrough
 csi.h, [306](#)
 csiOutFmt_YCbCr_420_fldp
 csi.h, [306](#)
 csiOutFmt_YCbCr_420_fldp_UVc
 csi.h, [306](#)
 csiOutFmt_YCbCr_420_fldt
 csi.h, [306](#)
 csiOutFmt_YCbCr_420_frp
 csi.h, [306](#)
 csiOutFmt_YCbCr_420_frp_UVc
 csi.h, [306](#)
 csiOutFmt_YCbCr_422_frt
 csi.h, [307](#)
 csiOutFmt_YCbCr_422_fldp
 csi.h, [306](#)
 csiOutFmt_YCbCr_422_fldp_UVc
 csi.h, [306](#)
 csiOutFmt_YCbCr_422_fldt
 csi.h, [306](#)
 csiOutFmt_YCbCr_422_frp
 csi.h, [306](#)
 csiOutFmt_YCbCr_422_frp_UVc
 csi.h, [306](#)
 csiOutFmt_YCbCr_422_frt
 csi.h, [307](#)
 csiOutFmt_YCbCr_422_inter
 csi.h, [306](#)
 csiOutFmt_YUV_420_p
 csi.h, [307](#)
 csiOutFmt_YUV_420_pc
 csi.h, [307](#)
 csiOutFmt_YUV_420_t
 csi.h, [307](#)
 csiOutFmt_YUV_422_p
 csi.h, [307](#)
 csiOutFmt_YUV_422_pc
 csi.h, [307](#)
 csiOutFmt_YUV_422_t
 csi.h, [307](#)
 csiRisingEdge
 csi.h, [305](#)
 csiRun
 csi.h, [310](#)
 csiSequence_UYVY
 csi.h, [308](#)
 csiSequence_VYUY
 csi.h, [308](#)
 csiSequence_YUYV
 csi.h, [308](#)
 csiSequence_YVYU
 csi.h, [308](#)
 csiSetMode
 csi.h, [311](#)
 csiSetPhase
 csi.h, [311](#)
 csiSetSeq
 csi.h, [311](#)
 csiStop
 csi.h, [312](#)
 csiWaitFrame
 csi.h, [312](#)
 ctime
 time.h, [710](#)
 CTL_DEL
 inputstr.h, [410](#)
 CTL_DWN
 inputstr.h, [410](#)

- CTL_END
 - inputstr.h, 410
- CTL_HOME
 - inputstr.h, 410
- CTL_INS
 - inputstr.h, 410
- CTL_KEY
 - inputstr.h, 410
- CTL_LEFT
 - inputstr.h, 410
- CTL_NONE
 - inputstr.h, 410
- CTL_PGDWN
 - inputstr.h, 410
- CTL_PGUP
 - inputstr.h, 410
- CTL_RIGHT
 - inputstr.h, 410
- CTL_UP
 - inputstr.h, 410
- ctype.h, 313
 - _IS_BLN, 313
 - _IS_CTL, 313
 - _IS_DIG, 313
 - _IS_HEX, 314
 - _IS_LOW, 314
 - _IS_PUN, 314
 - _IS_SP, 314
 - _IS_UPP, 314
 - _tolower, 314
 - _toupper, 314
 - isalnum, 314
 - isalpha, 315
 - isascii, 315
 - isblank, 315
 - iscntrl, 316
 - isdigit, 316
 - isgraph, 316
 - islower, 317
 - isprint, 317
 - ispunct, 317
 - isspace, 318
 - isupper, 318
 - isxdigit, 318
 - toascii, 319
 - tolower, 319
 - toupper, 319
- CurrentElementsAmount
 - sVector, 157
- cursorMove
 - crt.h, 298
- cursorOff
 - crt.h, 299
- cursorOn
 - crt.h, 299
- cursorRestore
 - crt.h, 299
- cursorStore
 - crt.h, 299
- curstp
 - TCB, 160
- d2bcd
 - stdlib.h, 658
- d_addr
 - ip_packet, 138
- Data
 - sVector, 157
- data
 - ip_packet, 138
 - tRingBuffer, 172
- DATA_INTERFACE_CLASS
 - usbdescriptors.h, 780
- DATA_INTERFACE_PROTOCOL_NONE
 - usbdescriptors.h, 780
- DATA_INTERFACE_SUBCLASS_NONE
 - usbdescriptors.h, 780
- date
 - datetime.h, 322
- DATE_TIME
 - datetime.h, 321
- date_time, 111
 - Day, 111
 - Hour, 111
 - Minute, 111
 - Month, 111
 - Second, 111
 - Year, 111
- datetime.h, 321
 - convertDateTimeTime, 321
 - date, 322
 - DATE_TIME, 321
 - diffdate, 322
 - DT_COMPACT, 321
 - increaseDateTimeBySecond, 322
 - setTime, 323
- Day
 - date_time, 111
 - dtcompact, 117
- dcache_disable
 - cache.h, 278
- dcache_enable
 - cache.h, 278
- dcache_status
 - cache.h, 278
- DCB
 - iolib.h, 445
- de2.h, 324
 - de2FlipScreenAndConstr, 326
 - de2GetDefaultScreenMode, 327
 - de2Init, 327
 - de2SetBacklight, 327
 - de2WaitVerticalRetrace, 328
 - eOverlayDataFormat, 325

- eScreenDeviceType, 326
- ovDataFormat_ABGR_1555, 325
- ovDataFormat_ABGR_4444, 325
- ovDataFormat_ABGR_8888, 325
- ovDataFormat_ARGB_1555, 325
- ovDataFormat_ARGB_4444, 325
- ovDataFormat_ARGB_8888, 325
- ovDataFormat_BGR_565, 325
- ovDataFormat_BGR_888, 325
- ovDataFormat_BGRA_4444, 325
- ovDataFormat_BGRA_5551, 325
- ovDataFormat_BGRA_8888, 325
- ovDataFormat_BGRX_8888, 325
- ovDataFormat_RGB_565, 325
- ovDataFormat_RGB_888, 325
- ovDataFormat_RGBA_4444, 325
- ovDataFormat_RGBA_5551, 325
- ovDataFormat_RGBA_8888, 325
- ovDataFormat_RGBX_8888, 325
- ovDataFormat_XBGR_8888, 325
- ovDataFormat_XRGB_8888, 325
- screenType_Lcd_480x272, 326
- screenType_Lcd_800x480, 326
- screenType_TM043NBH02, 326
- screenType_TM070RxH10, 326
- de2FlipScreenAndConstr
 - de2.h, 326
- de2GetDefaultScreenMode
 - de2.h, 327
- de2Init
 - de2.h, 327
- de2SetBacklight
 - de2.h, 327
- de2WaitVerticalRetrace
 - de2.h, 328
- debug_cond
 - multex.h, 513
- default_pipe
 - usb.h, 761
- DEFINE_ALIGN_BUFFER
 - multex.h, 513
- DEFINE_CACHE_ALIGN_BUFFER
 - multex.h, 513
- del
 - filesyst.h, 351
- delay
 - TCB, 160
- deleteRingBuffer
 - ringbuffer.h, 536
- deleteSurface
 - a20graph.h, 241
- deleteSurfaceDataArray
 - a20graph.h, 241
- deleteWorkTask
 - tasklib.h, 697
- delta
 - tRingBuffer, 172
- desc
 - usb_config, 206
 - usb_interface, 222
- descriptor
 - usb_class_descriptor, 184
 - usb_descriptor, 209
 - usb_device, 212
- DEV_HDR
 - iolib.h, 445
- devDCB
 - device_header, 113
- devHdr
 - file_fcb, 124
- DEVICE
 - iolib.h, 446
- device
 - usb_descriptor, 209
- device_header, 113
 - devDCB, 113
 - devName, 113
 - drvNumber, 113
 - next, 113
- devName
 - device_header, 113
- devname
 - usb_device, 212
- devnum
 - usb_device, 212
- diffdate
 - datetime.h, 322
- difftime
 - time.h, 710
- dir
 - filesyst.h, 351
- dircopy
 - filesyst.h, 352
- dirCopy_Delay
 - filesyst.h, 352
- direct_line
 - usb_class_descriptor, 184
- dirExists
 - iolib.h, 449
- dirIsEmpty
 - filesyst.h, 353
- Display, 114
 - a20graph.h, 251
 - BPP, 114
 - Constr, 114
 - DSize, 114
 - Height, 114
 - interface, 114
 - LFB, 115
 - mode, 115
 - modeline, 115
 - Screen, 115
 - VideoMode, 115
 - Width, 115

div
 stdlib.h, 658
 DIV_ROUND
 multex.h, 514
 DIV_ROUND_CLOSEST
 multex.h, 514
 DIV_ROUND_UP
 multex.h, 514
 div_t, 116
 quot, 116
 rem, 116
 dma_addr_t
 multex.h, 518
 dos2win
 unicode.h, 756
 doTerminate
 terminator.h, 707
 driver
 usb_device, 212
 drivers.dox, 329
 drvGetBit
 arch.h, 257
 drvGetBitGroup
 arch.h, 258
 drvInitBit
 arch.h, 258
 drvInitBitGroup
 arch.h, 258
 drvInitGpio
 arch.h, 259
 drvNumber
 device_header, 113
 drvResetBit
 arch.h, 259
 drvSetBit
 arch.h, 260
 drvSetBitGroup
 arch.h, 260
 drvSetGpio
 arch.h, 260
 DSize
 Display, 114
 dst_image
 g2d_blt, 126
 g2d_fillrect, 128
 g2d_stretchblt, 131
 dst_rect
 g2d_fillrect, 128
 g2d_stretchblt, 131
 dst_x
 g2d_blt, 126
 dst_y
 g2d_blt, 126
 DT_COMPACT
 datetime.h, 321
 dtcompact, 117
 Day, 117
 Hour, 117
 Minute, 117
 Month, 117
 Sec2, 117
 Year, 117
 E2BIG
 errno.h, 334
 EACCES
 errno.h, 334
 EADDRINUSE
 errno.h, 334
 EADDRNOTAVAIL
 errno.h, 334
 EADV
 errno.h, 334
 EAFNOSUPPORT
 errno.h, 335
 EAGAIN
 errno.h, 335
 eAlignType
 fontsdefines.h, 377
 EALREADY
 errno.h, 335
 EBADE
 errno.h, 335
 EBADF
 errno.h, 335
 EBADFD
 errno.h, 335
 EBADMSG
 errno.h, 335
 EBADR
 errno.h, 335
 EBADRQC
 errno.h, 335
 EBADSLT
 errno.h, 336
 EBFONT
 errno.h, 336
 EBUSY
 errno.h, 336
 ECANCELED
 errno.h, 336
 ECHILD
 errno.h, 336
 ECHRNG
 errno.h, 336
 ECOMM
 errno.h, 336
 ECONNABORTED
 errno.h, 336
 ECONNREFUSED
 errno.h, 336
 ECONNRESET
 errno.h, 337
 eCsiClockEdge

csi.h, 304
 eCsiFieldSelection
 csi.h, 305
 eCsiInputFormat
 csi.h, 305
 eCsiOutputFormat
 csi.h, 306
 eCsiPolarity
 csi.h, 307
 EDEADLK
 errno.h, 337
 EDEADLOCK
 errno.h, 337
 EDESTADDRREQ
 errno.h, 337
 EDOM
 errno.h, 337
 EDOTDOT
 errno.h, 337
 EDQUOT
 errno.h, 337
 EEXIST
 errno.h, 337
 EFAULT
 errno.h, 337
 EFBIG
 errno.h, 338
 eGpioInterruptMode
 gpio.h, 381
 EHOSTDOWN
 errno.h, 338
 EHOSTUNREACH
 errno.h, 338
 EHWPOISON
 errno.h, 338
 EIDRM
 errno.h, 338
 EILSEQ
 errno.h, 338
 EINPROGRESS
 errno.h, 338
 eintDoubleEdge
 gpio.h, 381
 eintHighLevel
 gpio.h, 381
 eintLowLevel
 gpio.h, 381
 eintNegativeEdge
 gpio.h, 381
 eintPositiveEdge
 gpio.h, 381
 EINTR
 errno.h, 338
 EINVAL
 errno.h, 338
 EIO
 errno.h, 339
 EISCONN
 errno.h, 339
 EISDIR
 errno.h, 339
 EISNAM
 errno.h, 339
 EKEYEXPIRED
 errno.h, 339
 EKEYREJECTED
 errno.h, 339
 EKEYREMOVED
 errno.h, 339
 EL2HLT
 errno.h, 339
 EL2NSYNC
 errno.h, 339
 EL3HLT
 errno.h, 340
 EL3RST
 errno.h, 340
 ElementSize
 sVector, 157
 ELIBACC
 errno.h, 340
 ELIBBAD
 errno.h, 340
 ELIBEXEC
 errno.h, 340
 ELIBMAX
 errno.h, 340
 ELIBSCN
 errno.h, 340
 ELNRNG
 errno.h, 340
 ELOOP
 errno.h, 340
 eMapstrValueType
 mapstr.h, 482
 EMEDIUMTYPE
 errno.h, 341
 EMFILE
 errno.h, 341
 EMLINK
 errno.h, 341
 emptySurface
 softgraph.h, 585
 EMSGSIZE
 errno.h, 341
 EMULTIHOP
 errno.h, 341
 enabled
 tDrvGpio, 166
 ENAMETOOLONG
 errno.h, 341
 ENAVAIL
 errno.h, 341
 end

- tMapIterators, 171
- endpoint
 - usb_descriptor, 209
- ENETDOWN
 - errno.h, 341
- ENETRESET
 - errno.h, 341
- ENETUNREACH
 - errno.h, 342
- ENFILE
 - errno.h, 342
- ENOANO
 - errno.h, 342
- ENOBUFFS
 - errno.h, 342
- ENOCSI
 - errno.h, 342
- ENODATA
 - errno.h, 342
- ENODEV
 - errno.h, 342
- ENOENT
 - errno.h, 342
- ENOEXEC
 - errno.h, 342
- ENOKEY
 - errno.h, 343
- ENOLCK
 - errno.h, 343
- ENOLINK
 - errno.h, 343
- ENOMEDIUM
 - errno.h, 343
- ENOMEM
 - errno.h, 343
- ENOMSG
 - errno.h, 343
- ENONET
 - errno.h, 343
- ENOPKG
 - errno.h, 343
- ENOPROTOPT
 - errno.h, 343
- ENOSPC
 - errno.h, 344
- ENOSR
 - errno.h, 344
- ENOSTR
 - errno.h, 344
- ENOSYS
 - errno.h, 344
- ENOTBLK
 - errno.h, 344
- ENOTCONN
 - errno.h, 344
- ENOTDIR
 - errno.h, 344
- ENOTEMPTY
 - errno.h, 344
- ENOTNAM
 - errno.h, 344
- ENOTRECOVERABLE
 - errno.h, 345
- ENOTSOCK
 - errno.h, 345
- ENOTSUP
 - errno.h, 345
- ENOTTY
 - errno.h, 345
- ENOTUNIQ
 - errno.h, 345
- env_var, 119
 - name, 119
 - next, 119
 - val, 119
- env_vars.h, 330
 - getenv_var, 330
 - printenv, 330
- ENXIO
 - errno.h, 345
- EOF
 - stdio.h, 632
- eof
 - file_fcb, 124
- EOPNOTSUPP
 - errno.h, 345
- EOVERFLOW
 - errno.h, 345
- eOverlayDataFormat
 - de2.h, 325
- EOWNERDEAD
 - errno.h, 345
- ep_desc
 - usb_interface, 222
- EPERM
 - errno.h, 346
- EPFNOSUPPORT
 - errno.h, 346
- EPIPE
 - errno.h, 346
- epmaxpacketin
 - usb_device, 212
- epmaxpacketout
 - usb_device, 212
- EPROTO
 - errno.h, 346
- EPROTONOSUPPORT
 - errno.h, 346
- EPROTOTYPE
 - errno.h, 346
- ERANGE
 - errno.h, 346
- EREMCHG
 - errno.h, 346

EREMOTE
 errno.h, 346

EREMOTEIO
 errno.h, 347

ERESTART
 errno.h, 347

ERFKILL
 errno.h, 347

EROFS
 errno.h, 347

errno
 errno.h, 349

errno-base.h, 331

errno.h, 332

 E2BIG, 334

 EACCES, 334

 EADDRINUSE, 334

 EADDRNOTAVAIL, 334

 EADV, 334

 EAFNOSUPPORT, 335

 EAGAIN, 335

 EALREADY, 335

 EBADE, 335

 EBADF, 335

 EBADFD, 335

 EBADMSG, 335

 EBADR, 335

 EBADRQC, 335

 EBADSLT, 336

 EBFONT, 336

 EBUSY, 336

 ECANCELED, 336

 ECHILD, 336

 ECHRNG, 336

 ECOMM, 336

 ECONNABORTED, 336

 ECONNREFUSED, 336

 ECONNRESET, 337

 EDEADLK, 337

 EDEADLOCK, 337

 EDESTADDRREQ, 337

 EDOM, 337

 EDOTDOT, 337

 EDQUOT, 337

 EEXIST, 337

 EFAULT, 337

 EFBIG, 338

 EHOSTDOWN, 338

 EHOSTUNREACH, 338

 EHWPOISON, 338

 EIDRM, 338

 EILSEQ, 338

 EINPROGRESS, 338

 EINTR, 338

 EINVAL, 338

 EIO, 339

 EISCONN, 339

 EISDIR, 339

 EISNAM, 339

 EKEYEXPIRED, 339

 EKEYREJECTED, 339

 EKEYREVOKED, 339

 EL2HLT, 339

 EL2NSYNC, 339

 EL3HLT, 340

 EL3RST, 340

 ELIBACC, 340

 ELIBBAD, 340

 ELIBEXEC, 340

 ELIBMAX, 340

 ELIBSCN, 340

 ELNRNG, 340

 ELOOP, 340

 EMEDIUMTYPE, 341

 EMFILE, 341

 EMLINK, 341

 EMSGSIZE, 341

 EMULTIHOP, 341

 ENAMETOOLONG, 341

 ENAVAIL, 341

 ENETDOWN, 341

 ENETRESET, 341

 ENETUNREACH, 342

 ENFILE, 342

 ENOANO, 342

 ENOBUFFS, 342

 ENOCSI, 342

 ENODATA, 342

 ENODEV, 342

 ENOENT, 342

 ENOEXEC, 342

 ENOKEY, 343

 ENOLCK, 343

 ENOLINK, 343

 ENOMEDIUM, 343

 ENOMEM, 343

 ENOMSG, 343

 ENONET, 343

 ENOPKG, 343

 ENOPROTOPT, 343

 ENOSPC, 344

 ENOSR, 344

 ENOSTR, 344

 ENOSYS, 344

 ENOTBLK, 344

 ENOTCONN, 344

 ENOTDIR, 344

 ENOTEMPTY, 344

 ENOTNAM, 344

 ENOTRECOVERABLE, 345

 ENOTSOCK, 345

 ENOTSUP, 345

 ENOTTY, 345

 ENOTUNIQ, 345

- ENXIO, 345
- EOPNOTSUPP, 345
- EOVERFLOW, 345
- EOWNERDEAD, 345
- EPERM, 346
- EPFNOSUPPORT, 346
- EPIPE, 346
- EPROTO, 346
- EPROTONOSUPPORT, 346
- EPROTOTYPE, 346
- ERANGE, 346
- EREMCHG, 346
- EREMOTE, 346
- EREMOTEIO, 347
- ERESTART, 347
- ERFKILL, 347
- EROFS, 347
- errno, 349
- ESHUTDOWN, 347
- ESOCKTNOSUPPORT, 347
- ESPIPE, 347
- ESRCH, 347
- ESRMNT, 347
- ESTALE, 348
- ESTRPIPE, 348
- ETIME, 348
- ETIMEDOUT, 348
- ETOOMANYREFS, 348
- ETXTBSY, 348
- EUCLEAN, 348
- EUNATCH, 348
- EUSERS, 348
- EWOULDBLOCK, 349
- EXDEV, 349
- EXFULL, 349
- ERROR
 - multex.h, 519
- eSciInputSequence
 - csi.h, 308
- eScreenDeviceType
 - de2.h, 326
- ESHUTDOWN
 - errno.h, 347
- ESOCKTNOSUPPORT
 - errno.h, 347
- ESPIPE
 - errno.h, 347
- ESRCH
 - errno.h, 347
- ESRMNT
 - errno.h, 347
- ESTALE
 - errno.h, 348
- ESTRPIPE
 - errno.h, 348
- ethernet_networking
 - usb_class_descriptor, 184
- ETIME
 - errno.h, 348
- ETIMEDOUT
 - errno.h, 348
- ETOOMANYREFS
 - errno.h, 348
- eTtfFontOptions
 - fonts.h, 365
- eTtfTextOutputType
 - fonts.h, 366
- ETXTBSY
 - errno.h, 348
- eUartParity
 - uart.h, 737
- EUCLEAN
 - errno.h, 348
- EUNATCH
 - errno.h, 348
- EUSERS
 - errno.h, 348
- EWOULDBLOCK
 - errno.h, 349
- EXDEV
 - errno.h, 349
- EXFULL
 - errno.h, 349
- exit
 - stdlib.h, 658
- exit_code
 - TCB, 160
- EXIT_FAILURE
 - stdlib.h, 654
- exit_fun
 - exit_st, 120
- exit_list
 - TCB, 160
- exit_proc
 - tasklib.h, 696
- exit_st, 120
 - exit_fun, 120
 - next, 120
- EXIT_SUCCESS
 - stdlib.h, 654
- exitbuf
 - stdlib.h, 659
 - TCB, 160
- exitcode
 - stdlib.h, 659
- exp
 - math.h, 490
- exp2
 - math.h, 490
- exp2f
 - math.h, 490
- expandFileName
 - fnames.h, 359
- expf

math.h, 490
 extension_unit
 usb_class_descriptor, 184
 extractDevice
 fnames.h, 359
 extractFileDevPath
 fnames.h, 359
 extractFileDrive
 fnames.h, 360
 extractFilePath
 fnames.h, 360
 extractNextDir
 fnames.h, 361

F
 msgQID, 142
 FA_ARCH
 iolib.h, 441
 FA_CAT
 iolib.h, 441
 FA_HIDE
 iolib.h, 441
 FA_LABEL
 iolib.h, 441
 FA_RO
 iolib.h, 441
 FA_SYSTEM
 iolib.h, 442
 fabs
 math.h, 490
 fabsf
 math.h, 490
 false
 stdbool.h, 615
 FCB
 iolib.h, 446
 fclose
 stdio.h, 634
 fd
 FILE, 123
 fdopen
 stdio.h, 634
 feof
 stdio.h, 634
 ferror
 stdio.h, 635
 ff_attrib
 ffblk, 121
 ff_date_time
 ffblk, 121
 ff_directory
 ffblk, 121
 ff_dname
 ffblk, 121
 ff_fsize
 ffblk, 121
 ff_idx
 ffblk, 122
 ff_lname
 ffblk, 122
 ff_mask
 ffblk, 122
 ff_name
 ffblk, 122
 ff_path
 ffblk, 122
 FFBLK
 iolib.h, 446
 ffbk, 121
 ff_attrib, 121
 ff_date_time, 121
 ff_directory, 121
 ff_dname, 121
 ff_fsize, 121
 ff_idx, 122
 ff_lname, 122
 ff_mask, 122
 ff_name, 122
 ff_path, 122
 fflush
 stdio.h, 635
 fgetc
 stdio.h, 635
 fgetpos
 stdio.h, 636
 fgets
 stdio.h, 636
 FILE, 123
 fd, 123
 signa, 123
 ugf, 123
 unget, 123
 file_fcb, 124
 blkBuf, 124
 blkNum, 124
 catIndex, 124
 devHdr, 124
 eof, 124
 fileName, 124
 fileSize, 125
 flushed, 125
 mode, 125
 paramBlk, 125
 position, 125
 startBlk, 125
 FILE_SIGNATURE
 stdio.h, 632
 fileExists
 iolib.h, 449
 fileName
 file_fcb, 124
 FILENAME_MAX
 stdio.h, 632
 fileNamePreprocess

- fnames.h, 361
- fileSize
 - file_fcb, 125
 - iolib.h, 450
- filesyst.h, 350
 - copy, 350
 - copyFile, 350
 - del, 351
 - dir, 351
 - dircopy, 352
 - dirCopy_Delay, 352
 - dirIsEmpty, 353
 - findFilesInDirAndSubdirs, 353
 - getFileSize, 354
 - loadFile, 355
 - loadFileSz, 355
 - move, 355
 - removeContentFromDir, 356
 - setWorkDevice, 356
 - silentDirCopy, 357
 - type, 357
- fillRect
 - a20graph.h, 241
- fillSurface
 - softgraph.h, 586
- findDev
 - iolib.h, 450
- findFCB
 - iolib.h, 450
- findFd
 - iolib.h, 451
- findFilesInDirAndSubdirs
 - filesyst.h, 353
- findFirst
 - iolib.h, 451
- findNext
 - iolib.h, 451
- FIOCD
 - iolib.h, 442
- FIOCHKDSK
 - iolib.h, 442
- FIODISKFORMAT
 - iolib.h, 442
- FIOEOF
 - iolib.h, 442
- FIOFILESIZE
 - iolib.h, 442
- FIOFINDFIRST
 - iolib.h, 442
- FIOFINDNEXT
 - iolib.h, 442
- FIOFLUSH
 - iolib.h, 442
- FIOFREESIZE
 - iolib.h, 443
- FIOGETDT
 - iolib.h, 443
- FIOGETLABEL
 - iolib.h, 443
- FIOGETLNAME
 - iolib.h, 443
- FIOGETTO
 - iolib.h, 443
- FIOMKD
 - iolib.h, 443
- FIORENAME
 - iolib.h, 443
- FIORESETDRV
 - iolib.h, 443
- FIORMD
 - iolib.h, 443
- FIOSEEK
 - iolib.h, 444
- FIOSETDT
 - iolib.h, 444
- FIOSETTO
 - iolib.h, 444
- FIOTELL
 - iolib.h, 444
- FIOUNMOUNT
 - iolib.h, 444
- FIOUPD
 - iolib.h, 444
- FIOWRKDIR
 - iolib.h, 444
- first
 - msgQID, 142
- flag
 - g2d_blt, 126
 - g2d_fillrect, 128
 - g2d_stretchblt, 131
- flags
 - TCB, 160
- FlipScreenAndConstr
 - a20graph.h, 242
- floor
 - math.h, 490
- floorf
 - math.h, 491
- flush
 - iolib.h, 452
- flush_dcache_all
 - cache.h, 279
- flush_dcache_range
 - cache.h, 279
- flushed
 - file_fcb, 125
 - Sem_Id, 148
- fmod
 - math.h, 491
- fnames.h, 358
 - buildFileName, 358
 - compareNames, 358
 - expandFileName, 359

- extractDevice, 359
- extractFileDevPath, 359
- extractFileDrive, 360
- extractFilePath, 360
- extractNextDir, 361
- fileNamePreprocess, 361
- getFileName, 361
- getFileNameNonConst, 362
- getWorkDevice, 362
- getWorkDirectory, 362
- setWorkDevice, 363
- FontColor
 - sTtfFont, 156
- FontOptions
 - sTtfFont, 156
- FontPixelSize
 - sTtfFont, 156
- fonts.h, 364
 - eTtfFontOptions, 365
 - eTtfTextOutputType, 366
 - pTtfFont, 365
 - pTtfPrivateFontStruct, 365
 - sTtfPrivateFontStruct, 365
 - ttf_CloseFontAfterPrerender, 366
 - ttf_ConvertFontSize, 367
 - ttf_FreeFont, 367
 - ttf_GetTextPixelLength, 368
 - ttf_GetTextPixelLengthUtf16, 368
 - ttf_GetTextRect, 369
 - ttf_GetTextRectUtf16, 369
 - ttf_KeepFontOpen, 366
 - ttf_LoadFont, 370
 - ttf_NoGlyphSaving, 366
 - ttf_NoPrerender, 366
 - ttf_NormalOrientation, 366
 - ttf_PrerenderASCII, 366
 - ttf_PrerenderDecNumbers, 366
 - ttf_Print, 370
 - ttf_PrintRect, 371
 - ttf_PrintRectNoCheckBorder, 371
 - ttf_PrintRectUft16, 372
 - ttf_PrintRectUft16NoCheckBorder, 373
 - ttf_PrintUtf16, 373
 - ttf_RotatedOrientation, 366
 - ttf_SaveGlyphs, 366
 - ttf_SetDpi, 374
 - ttf_SetTextOutputType, 374
 - TTOT_Debug, 367
 - TTOT_Errors, 367
 - TTOT_FontLoadingTime, 367
 - TTOT_None, 367
- fontsdefines.h, 376
 - alignHCenter, 377
 - alignHLeft, 377
 - alignHRight, 377
 - alignType, 377
 - alignVBottom, 378
 - alignVMiddle, 378
 - alignVTop, 378
 - ARRAY_INDEX_TO_SYMBOL, 376
 - ASCII_PRINTED_SYMBOLS_AMOUNT, 376
 - eAlignType, 377
 - noAlign, 377
 - pTextRect, 377
 - sTextRect, 377
 - SYMBOL_CODE_TO_ARRAY_INDEX, 376
 - SYMBOL_IS_PRINTED_ASCII, 376
- FontSize
 - sTtfFont, 156
- fopen
 - stdio.h, 637
- FOPEN_MAX
 - stdio.h, 632
- format
 - g2d_image, 129
- formatVolume
 - iolib.h, 452
- fp
 - REG_SET, 144
- fpos_t
 - stdio.h, 633
- fprintf
 - stdio.h, 637
- fputc
 - stdio.h, 637
- fputs
 - stdio.h, 638
- frac
 - math.h, 491
- fread
 - stdio.h, 638
- free
 - memlib.h, 496
- freeFCB
 - iolib.h, 452
- freeMpeg4FrameMem
 - mpeg4codec.h, 501
- freeSpace
 - iolib.h, 453
- freeSurface
 - softgraph.h, 586
- freopen
 - stdio.h, 639
- frexp
 - math.h, 491
- fromRGB
 - softgraph.h, 586
- fscanf
 - stdio.h, 639
- fsData
 - blk_dev, 109
- fseek
 - stdio.h, 639
- fsetpos

stdio.h, [640](#)
 ftell
 stdio.h, [640](#)
 funcCode
 blk_dev, [109](#)
 FUNCPTR
 multex.h, [518](#)
 function
 usb_class_descriptor, [184](#)
 fwrite
 stdio.h, [641](#)

 g2d_blt, [126](#)
 alpha, [126](#)
 color, [126](#)
 dst_image, [126](#)
 dst_x, [126](#)
 dst_y, [126](#)
 flag, [126](#)
 src_image, [126](#)
 src_rect, [126](#)
 G2D_BLT_DST_COLORKEY
 a20graph.h, [233](#)
 g2d_blt_flags
 a20graph.h, [232](#)
 G2D_BLT_FLIP_HORIZONTAL
 a20graph.h, [233](#)
 G2D_BLT_FLIP_VERTICAL
 a20graph.h, [233](#)
 G2D_BLT_MIRROR135
 a20graph.h, [233](#)
 G2D_BLT_MIRROR45
 a20graph.h, [233](#)
 G2D_BLT_MULTI_ALPHA
 a20graph.h, [232](#)
 G2D_BLT_NONE
 a20graph.h, [232](#)
 G2D_BLT_PIXEL_ALPHA
 a20graph.h, [232](#)
 G2D_BLT_PLANE_ALPHA
 a20graph.h, [232](#)
 G2D_BLT_ROTATE180
 a20graph.h, [233](#)
 G2D_BLT_ROTATE270
 a20graph.h, [233](#)
 G2D_BLT_ROTATE90
 a20graph.h, [233](#)
 G2D_BLT_SRC_COLORKEY
 a20graph.h, [233](#)
 g2d_data_fmt
 a20graph.h, [234](#)
 G2D_FIL_MULTI_ALPHA
 a20graph.h, [236](#)
 G2D_FIL_NONE
 a20graph.h, [236](#)
 G2D_FIL_PIXEL_ALPHA
 a20graph.h, [236](#)

G2D_FIL_PLANE_ALPHA
 a20graph.h, [236](#)
 g2d_fillrect, [128](#)
 alpha, [128](#)
 color, [128](#)
 dst_image, [128](#)
 dst_rect, [128](#)
 flag, [128](#)
 g2d_fillrect_flags
 a20graph.h, [236](#)
 G2D_FMT_1BPP_MONO
 a20graph.h, [235](#)
 G2D_FMT_1BPP_PALETTE
 a20graph.h, [235](#)
 G2D_FMT_2BPP_MONO
 a20graph.h, [235](#)
 G2D_FMT_2BPP_PALETTE
 a20graph.h, [235](#)
 G2D_FMT_4BPP_MONO
 a20graph.h, [235](#)
 G2D_FMT_4BPP_PALETTE
 a20graph.h, [235](#)
 G2D_FMT_8BPP_MONO
 a20graph.h, [235](#)
 G2D_FMT_8BPP_PALETTE
 a20graph.h, [235](#)
 G2D_FMT_ABGR1555
 a20graph.h, [234](#)
 G2D_FMT_ABGR4444
 a20graph.h, [234](#)
 G2D_FMT_ABGR8888
 a20graph.h, [234](#)
 G2D_FMT_ABGR_AVUY8888
 a20graph.h, [234](#)
 G2D_FMT_ARGB1555
 a20graph.h, [234](#)
 G2D_FMT_ARGB4444
 a20graph.h, [234](#)
 G2D_FMT_ARGB8888
 a20graph.h, [234](#)
 G2D_FMT_ARGB_AYUV8888
 a20graph.h, [234](#)
 G2D_FMT_BGR565
 a20graph.h, [235](#)
 G2D_FMT_BGRA4444
 a20graph.h, [234](#)
 G2D_FMT_BGRA5551
 a20graph.h, [234](#)
 G2D_FMT_BGRA8888
 a20graph.h, [234](#)
 G2D_FMT_BGRA_VUYA8888
 a20graph.h, [234](#)
 G2D_FMT_BGRX8888
 a20graph.h, [234](#)
 G2D_FMT_IYUV422
 a20graph.h, [235](#)
 G2D_FMT_PYUV411

[a20graph.h, 235](#)
[G2D_FMT_PYUV411UVC](#)
[a20graph.h, 235](#)
[G2D_FMT_PYUV420](#)
[a20graph.h, 235](#)
[G2D_FMT_PYUV420UVC](#)
[a20graph.h, 235](#)
[G2D_FMT_PYUV422](#)
[a20graph.h, 235](#)
[G2D_FMT_PYUV422UVC](#)
[a20graph.h, 235](#)
[G2D_FMT_RGB565](#)
[a20graph.h, 234](#)
[G2D_FMT_RGBA4444](#)
[a20graph.h, 234](#)
[G2D_FMT_RGBA5551](#)
[a20graph.h, 234](#)
[G2D_FMT_RGBA8888](#)
[a20graph.h, 234](#)
[G2D_FMT_RGBA_YUVA8888](#)
[a20graph.h, 234](#)
[G2D_FMT_RGBX8888](#)
[a20graph.h, 234](#)
[G2D_FMT_XBGR8888](#)
[a20graph.h, 234](#)
[G2D_FMT_XRGB8888](#)
[a20graph.h, 234](#)
[g2d_image, 129](#)
[addr, 129](#)
[format, 129](#)
[h, 129](#)
[pixel_seq, 129](#)
[w, 129](#)
[g2d_pixel_seq](#)
[a20graph.h, 237](#)
[g2d_rect, 130](#)
[h, 130](#)
[w, 130](#)
[x, 130](#)
[y, 130](#)
[G2D_SEQ_1BPP_BIG_BIG](#)
[a20graph.h, 237](#)
[G2D_SEQ_1BPP_BIG_LITTER](#)
[a20graph.h, 237](#)
[G2D_SEQ_1BPP_LITTER_BIG](#)
[a20graph.h, 237](#)
[G2D_SEQ_1BPP_LITTER_LITTER](#)
[a20graph.h, 237](#)
[G2D_SEQ_2BPP_BIG_BIG](#)
[a20graph.h, 237](#)
[G2D_SEQ_2BPP_BIG_LITTER](#)
[a20graph.h, 237](#)
[G2D_SEQ_2BPP_LITTER_BIG](#)
[a20graph.h, 237](#)
[G2D_SEQ_2BPP_LITTER_LITTER](#)
[a20graph.h, 237](#)
[G2D_SEQ_NORMAL](#)

[a20graph.h, 237](#)
[G2D_SEQ_P01](#)
[a20graph.h, 237](#)
[G2D_SEQ_P0123](#)
[a20graph.h, 237](#)
[G2D_SEQ_P01234567](#)
[a20graph.h, 237](#)
[G2D_SEQ_P10](#)
[a20graph.h, 237](#)
[G2D_SEQ_P10325476](#)
[a20graph.h, 237](#)
[G2D_SEQ_P3210](#)
[a20graph.h, 237](#)
[G2D_SEQ_P67452301](#)
[a20graph.h, 237](#)
[G2D_SEQ_P76543210](#)
[a20graph.h, 237](#)
[G2D_SEQ_VUVU](#)
[a20graph.h, 237](#)
[G2D_SEQ_VYUY](#)
[a20graph.h, 237](#)
[G2D_SEQ_YVYU](#)
[a20graph.h, 237](#)
[g2d_stretchblt, 131](#)
[alpha, 131](#)
[color, 131](#)
[dst_image, 131](#)
[dst_rect, 131](#)
[flag, 131](#)
[src_image, 131](#)
[src_rect, 131](#)
[gcvf](#)
[stdlib.h, 659](#)
[generic](#)
[usb_class_descriptor, 184](#)
[usb_descriptor, 209](#)
[get_c](#)
[console.h, 288](#)
[get_unaligned](#)
[multex.h, 514](#)
[getc](#)
[stdio.h, 641](#)
[getch](#)
[stdio.h, 641](#)
[getchar](#)
[stdio.h, 642](#)
[getConstr2Buffer](#)
[a20graph.h, 242](#)
[getConstrAlpSurface](#)
[a20graph.h, 242](#)
[getConstrBuffer](#)
[a20graph.h, 243](#)
[getConstrSurface](#)
[a20graph.h, 243](#)
[getCtlKey](#)
[inputstr.h, 411](#)
[getenv](#)

stdlib.h, 660
getenv_var
 env_vars.h, 330
getFileName
 fnames.h, 361
getFileNameNonConst
 fnames.h, 362
getFilesInDir
 iolib.h, 453
getFileSize
 filesyst.h, 354
getFullFileName
 iolib.h, 453
geth
 console.h, 288
getLFB
 a20graph.h, 243
getLongName
 iolib.h, 454
getMpeg4InptFrameBuff
 mpeg4codec.h, 501
getMpeg4OutFrame
 mpeg4codec.h, 502
getPrimaryIpAddress
 udp.h, 748
getRecSndBuffer
 sound.h, 602
gets
 stdio.h, 642
getScreen2Buffer
 a20graph.h, 243
getScreenAlpSurface
 a20graph.h, 243
getScreenBitsPerPixel
 a20graph.h, 244
getScreenBuffer
 a20graph.h, 244
getScreenHeight
 a20graph.h, 244
getScreenPitch
 a20graph.h, 244
getScreenSurface
 a20graph.h, 244
getScreenWidth
 a20graph.h, 245
getsocktimeout
 socket.h, 578
getVolumeLabel
 iolib.h, 454
getWord
 stdlib.h, 660
getWorkDevice
 fnames.h, 362
getWorkDir
 iolib.h, 455
getWorkDir_s
 iolib.h, 455
getWorkDirectory
 fnames.h, 362
gmtime
 time.h, 711
gmtime_r
 time.h, 711
gpio.h, 379
 eGpioInterruptMode, 381
 eintDoubleEdge, 381
 eintHighLevel, 381
 eintLowLevel, 381
 eintNegativeEdge, 381
 eintPositiveEdge, 381
 gpio_direction_input, 381
 gpio_direction_output, 382
 gpio_from_string, 382
 gpio_get_value, 383
 gpio_pull_disable, 383
 gpio_pull_down, 383
 gpio_pull_up, 384
 gpio_set_mux, 384
 gpio_set_value, 385
 gpioInterruptConnect, 385
 gpioInterruptDisconnect, 386
 P_A, 380
 P_B, 380
 P_C, 380
 P_D, 380
 P_E, 380
 P_F, 380
 P_G, 380
 P_H, 381
 P_I, 381
 gpio_direction_input
 gpio.h, 381
 gpio_direction_output
 gpio.h, 382
 gpio_from_string
 gpio.h, 382
 gpio_get_value
 gpio.h, 383
 gpio_pull_disable
 gpio.h, 383
 gpio_pull_down
 gpio.h, 383
 gpio_pull_up
 gpio.h, 384
 gpio_set_mux
 gpio.h, 384
 gpio_set_value
 gpio.h, 385
 gpioInterruptConnect
 gpio.h, 385
 gpioInterruptDisconnect
 gpio.h, 386
h

- g2d_image, 129
- g2d_rect, 130
- iniRect, 137
- textRect, 167
- h264_decode
 - cedrus.h, 282
- h264_decode_part
 - cedrus.h, 282
- h264_decoder_free
 - cedrus.h, 283
- h264_decoder_init
 - cedrus.h, 283
- h264_decoder_set_mk
 - cedrus.h, 284
- h264_decoder_set_qp
 - cedrus.h, 284
- h264_decoder_set_wm
 - cedrus.h, 284
- h264_encode
 - cedrus.h, 285
- h264_encoder_free
 - cedrus.h, 285
- h264_encoder_init
 - cedrus.h, 285
- h264_encoder_set_qp
 - cedrus.h, 286
- H264_FMT_NV12
 - cedrus.h, 282
- H264_FMT_NV16
 - cedrus.h, 282
- h264_get_out
 - cedrus.h, 286
- h264_is_keyframe
 - cedrus.h, 286
- h264_set_src_format
 - cedrus.h, 287
- halted
 - usb_device, 212
- hard_reset
 - timer.h, 719
- have_langid
 - usb_device, 212
- hcd
 - usb_device, 212
- HDC
 - a20graph.h, 232
- HDCp
 - softgraph.h, 584
- hDrv
 - blk_cache, 107
- header_function
 - usb_class_descriptor, 184
- Height
 - Display, 114
- height
 - sDisplayInfo, 146
- hexToInt
- string.h, 672
- hid
 - usb_class_descriptor, 184
- Hour
 - date_time, 111
 - dtcompact, 117
- hsync_len
 - tScreenDeviceMode, 173
- i2c.h, 387
 - I2C_0, 387
 - I2C_1, 388
 - I2C_2, 388
 - I2C_3, 388
 - I2C_4, 388
 - I2C_CLK_100_kHz, 388
 - I2C_CLK_400_kHz, 388
 - I2C_CLK_FAST, 388
 - I2C_CLK_NORMAL, 388
 - I2C_GPIO_ADDITIONAL, 388
 - I2C_GPIO_DEFAULT, 389
 - i2cInit, 389
 - i2cRead, 389
 - i2cRegisterRead, 390
 - i2cRegisterWrite, 391
 - i2cWrite, 391
- I2C_0
 - i2c.h, 387
- I2C_1
 - i2c.h, 388
- I2C_2
 - i2c.h, 388
- I2C_3
 - i2c.h, 388
- I2C_4
 - i2c.h, 388
- I2C_CLK_100_kHz
 - i2c.h, 388
- I2C_CLK_400_kHz
 - i2c.h, 388
- I2C_CLK_FAST
 - i2c.h, 388
- I2C_CLK_NORMAL
 - i2c.h, 388
- I2C_GPIO_ADDITIONAL
 - i2c.h, 388
- I2C_GPIO_DEFAULT
 - i2c.h, 389
- i2cInit
 - i2c.h, 389
- i2cRead
 - i2c.h, 389
- i2cRegisterRead
 - i2c.h, 390
- i2cRegisterWrite
 - i2c.h, 391
- i2cWrite

- i2c.h, 391
- icache_status
 - cache.h, 279
- iConfiguration
 - usb_configuration_descriptor, 208
- iCountryCodeRelDate
 - usb_class_country_selection_descriptor, 182
- idProduct
 - usb_device_descriptor, 216
- idVendor
 - usb_device_descriptor, 216
- iEndSystemIdentifier
 - usb_class_atm_networking_descriptor, 178
- if_desc
 - usb_config, 206
- iInterface
 - usb_interface_descriptor, 224
- iMACAddress
 - usb_class_ethernet_networking_descriptor, 187
- iManufacturer
 - usb_device_descriptor, 216
- imaxabs
 - inttypes.h, 436
- imaxdiv
 - inttypes.h, 436
- imaxdiv_t, 132
 - quot, 132
 - rem, 132
- IN
 - usbdescriptors.h, 780
- in_addr, 133
 - s_addr, 133
- in_port_t
 - socket.h, 576
- INADDR_ANY
 - socket.h, 574
- iName
 - usb_class_extension_unit_descriptor, 189
 - usb_class_network_channel_descriptor, 197
- inCnt
 - tRingBuffer, 172
- increaseDateTimeBySecond
 - datetime.h, 322
- inet_addr
 - socket.h, 574
- INI_FILE
 - inifiles.h, 395
- iniBinaryArray, 134
 - Array, 134
 - ArrayLength, 134
- iniCoords, 135
 - x, 135
 - y, 135
- iniFileChangeName
 - inifiles.h, 395
- iniFileCheckIfItemExists
 - inifiles.h, 395
- iniFileCheckIfSectionExists
 - inifiles.h, 396
- iniFileClose
 - inifiles.h, 396
- iniFileCreate
 - inifiles.h, 396
- iniFileDeleteItem
 - inifiles.h, 397
- iniFileDeleteSection
 - inifiles.h, 397
- iniFileFlush
 - inifiles.h, 398
- iniFileFree
 - inifiles.h, 398
- iniFileName
 - inifiles.h, 398
- iniFileOpen
 - inifiles.h, 399
- iniFilePrint
 - inifiles.h, 399
- iniFileReadBinaryArray
 - inifiles.h, 400
- iniFileReadBoolean
 - inifiles.h, 400
- iniFileReadConstString
 - inifiles.h, 401
- iniFileReadCoords
 - inifiles.h, 401
- iniFileReadHex
 - inifiles.h, 401
- iniFileReadIntArray
 - inifiles.h, 402
- iniFileReadInteger
 - inifiles.h, 402
- iniFileReadString
 - inifiles.h, 403
- iniFileReadTextRect
 - inifiles.h, 404
- iniFileRenameItem
 - inifiles.h, 404
- iniFileRenameSection
 - inifiles.h, 405
- inifiles.h, 393
 - INI_FILE, 395
 - iniFileChangeName, 395
 - iniFileCheckIfItemExists, 395
 - iniFileCheckIfSectionExists, 396
 - iniFileClose, 396
 - iniFileCreate, 396
 - iniFileDeleteItem, 397
 - iniFileDeleteSection, 397
 - iniFileFlush, 398
 - iniFileFree, 398
 - iniFileName, 398
 - iniFileOpen, 399
 - iniFilePrint, 399
 - iniFileReadBinaryArray, 400

iniFileReadBoolean, 400
 iniFileReadConstString, 401
 iniFileReadCoords, 401
 iniFileReadHex, 401
 iniFileReadIntArray, 402
 iniFileReadInteger, 402
 iniFileReadString, 403
 iniFileReadTextRect, 404
 iniFileRenameItem, 404
 iniFileRenameSection, 405
 iniFileSectionName, 405
 iniFileWriteBinaryArray, 405
 iniFileWriteBoolean, 406
 iniFileWriteCoords, 406
 iniFileWriteHex, 407
 iniFileWriteIntArray, 407
 iniFileWriteInteger, 407
 iniFileWriteString, 408
 iniFileWriteTextRect, 408
 iniFileSectionName
 inifiles.h, 405
 iniFileWriteBinaryArray
 inifiles.h, 405
 iniFileWriteBoolean
 inifiles.h, 406
 iniFileWriteCoords
 inifiles.h, 406
 iniFileWriteHex
 inifiles.h, 407
 iniFileWriteIntArray
 inifiles.h, 407
 iniFileWriteInteger
 inifiles.h, 407
 iniFileWriteString
 inifiles.h, 408
 iniFileWriteTextRect
 inifiles.h, 408
 iniIntArray, 136
 Array, 136
 ArrayLength, 136
 iniRect, 137
 h, 137
 w, 137
 x, 137
 y, 137
 init_2D_engine
 a20graph.h, 245
 INIT_STATIC_LIST
 list.h, 474
 INIT_STATIC_MUTEX
 semLib.h, 542
 INIT_STATIC_MUTEX_DEFAULT
 semLib.h, 542
 INIT_STATIC_SEM
 semLib.h, 542
 INIT_STATIC_SEM_DEFAULT
 semLib.h, 542
 INIT_TASK_NAME
 multex.h, 515
 initIntLib
 intlLib.h, 414
 initLvdsDisplay
 a20graph.h, 245
 initMemLib
 memLib.h, 497
 inputstr.h, 410
 ALT_B, 410
 ALT_D, 410
 ALT_F, 410
 CTL_DEL, 410
 CTL_DWN, 410
 CTL_END, 410
 CTL_HOME, 410
 CTL_INS, 410
 CTL_KEY, 410
 CTL_LEFT, 410
 CTL_NONE, 410
 CTL_PGDOWN, 410
 CTL_PGUP, 410
 CTL_RIGHT, 410
 CTL_UP, 410
 getCtlKey, 411
 INT16_C
 stdint.h, 620
 INT16_MAX
 stdint.h, 620
 INT16_MIN
 stdint.h, 620
 int16_t
 stdint.h, 626
 INT32_C
 stdint.h, 620
 INT32_MAX
 stdint.h, 620
 INT32_MIN
 stdint.h, 620
 int32_t
 stdint.h, 626
 INT64_C
 stdint.h, 620
 INT64_MAX
 stdint.h, 620
 INT64_MIN
 stdint.h, 620
 int64_t
 stdint.h, 627
 INT8_C
 stdint.h, 621
 INT8_MAX
 stdint.h, 621
 INT8_MIN
 stdint.h, 621
 int8_t
 stdint.h, 627

INT_FAST16_MAX
 stdint.h, 621
 INT_FAST16_MIN
 stdint.h, 621
 int_fast16_t
 stdint.h, 627
 INT_FAST32_MAX
 stdint.h, 621
 INT_FAST32_MIN
 stdint.h, 621
 int_fast32_t
 stdint.h, 627
 INT_FAST64_MAX
 stdint.h, 621
 INT_FAST64_MIN
 stdint.h, 621
 int_fast64_t
 stdint.h, 627
 INT_FAST8_MAX
 stdint.h, 622
 INT_FAST8_MIN
 stdint.h, 622
 int_fast8_t
 stdint.h, 627
 INT_LEAST16_MAX
 stdint.h, 622
 INT_LEAST16_MIN
 stdint.h, 622
 int_least16_t
 stdint.h, 627
 INT_LEAST32_MAX
 stdint.h, 622
 INT_LEAST32_MIN
 stdint.h, 622
 int_least32_t
 stdint.h, 627
 INT_LEAST64_MAX
 stdint.h, 622
 INT_LEAST64_MIN
 stdint.h, 622
 int_least64_t
 stdint.h, 627
 INT_LEAST8_MAX
 stdint.h, 622
 INT_LEAST8_MIN
 stdint.h, 623
 int_least8_t
 stdint.h, 628
 INT_MAX
 limits.h, 470
 INT_MIN
 limits.h, 471
 intConnect
 intlib.h, 414
 intCounter
 TCB, 161
 interface
 Display, 114
 usb_descriptor, 209
 INTERRUPT
 usbdescriptors.h, 781
 INTERRUPT_GROUP_MAJOR
 intlib.h, 412
 INTERRUPT_GROUP_MINOR
 intlib.h, 412
 INTERRUPT_GROUP_SYSTEM
 intlib.h, 413
 INTERRUPT_GROUP_TIME_CRITICAL
 intlib.h, 413
 INTERRUPT_PRIORITY_BASE
 intlib.h, 413
 INTERRUPT_PRIORITY_HIGHEST
 intlib.h, 413
 INTERRUPT_PRIORITY_LOWEST
 intlib.h, 413
 interruptConnect
 intlib.h, 414
 intlib.h, 412
 initIntLib, 414
 intConnect, 414
 INTERRUPT_GROUP_MAJOR, 412
 INTERRUPT_GROUP_MINOR, 412
 INTERRUPT_GROUP_SYSTEM, 413
 INTERRUPT_GROUP_TIME_CRITICAL, 413
 INTERRUPT_PRIORITY_BASE, 413
 INTERRUPT_PRIORITY_HIGHEST, 413
 INTERRUPT_PRIORITY_LOWEST, 413
 interruptConnect, 414
 irq, 415
 IRQ_HANDLED, 413
 IRQ_NONE, 413
 usr_int_proc, 413
 INTMAX_C
 stdint.h, 623
 INTMAX_MAX
 stdint.h, 623
 INTMAX_MIN
 stdint.h, 623
 intmax_t
 stdint.h, 628
 INTPTR_MAX
 stdint.h, 623
 INTPTR_MIN
 stdint.h, 623
 intptr_t
 stdint.h, 628
 intToHex
 string.h, 672
 intToHexUniversal
 string.h, 672
 inttypes.h, 416
 imaxabs, 436
 imaxdiv, 436
 PRId16, 419

PRId32, 419
PRId64, 419
PRId8, 419
PRIdFAST16, 419
PRIdFAST32, 419
PRIdFAST64, 419
PRIdFAST8, 419
PRIdLEAST16, 419
PRIdLEAST32, 419
PRIdLEAST64, 420
PRIdLEAST8, 420
PRIdMAX, 420
PRIdPTR, 420
PRIi16, 420
PRIi32, 420
PRIi64, 420
PRIi8, 420
PRIiFAST16, 420
PRIiFAST32, 421
PRIiFAST64, 421
PRIiFAST8, 421
PRIiLEAST16, 421
PRIiLEAST32, 421
PRIiLEAST64, 421
PRIiLEAST8, 421
PRIiMAX, 421
PRIiPTR, 421
PRIo16, 422
PRIo32, 422
PRIo64, 422
PRIo8, 422
PRIoFAST16, 422
PRIoFAST32, 422
PRIoFAST64, 422
PRIoFAST8, 422
PRIoLEAST16, 422
PRIoLEAST32, 423
PRIoLEAST64, 423
PRIoLEAST8, 423
PRIoMAX, 423
PRIoPTR, 423
PRIu16, 423
PRIu32, 423
PRIu64, 423
PRIu8, 423
PRIuFAST16, 424
PRIuFAST32, 424
PRIuFAST64, 424
PRIuFAST8, 424
PRIULEAST16, 424
PRIULEAST32, 424
PRIULEAST64, 424
PRIULEAST8, 424
PRIuMAX, 424
PRIuPTR, 425
PRIX16, 425
PRIX16, 425
PRIX32, 425
PRIX32, 425
PRIX64, 425
PRIX64, 425
PRIX8, 425
PRIX8, 425
PRIXFAST16, 426
PRIXFAST16, 426
PRIXFAST32, 426
PRIXFAST32, 426
PRIXFAST64, 426
PRIXFAST64, 426
PRIXFAST8, 426
PRIXFAST8, 426
PRIXLEAST16, 427
PRIXLEAST16, 426
PRIXLEAST32, 427
PRIXLEAST32, 427
PRIXLEAST64, 427
PRIXLEAST64, 427
PRIXLEAST8, 427
PRIXLEAST8, 427
PRIXMAX, 427
PRIXMAX, 427
PRIXPTR, 428
PRIXPTR, 428
SCNd16, 428
SCNd32, 428
SCNd64, 428
SCNd8, 428
SCNdFAST16, 428
SCNdFAST32, 428
SCNdFAST64, 428
SCNdFAST8, 429
SCNdLEAST16, 429
SCNdLEAST32, 429
SCNdLEAST64, 429
SCNdLEAST8, 429
SCNdMAX, 429
SCNdPTR, 429
SCNi16, 429
SCNi32, 429
SCNi64, 430
SCNi8, 430
SCNiFAST16, 430
SCNiFAST32, 430
SCNiFAST64, 430
SCNiFAST8, 430
SCNiLEAST16, 430
SCNiLEAST32, 430
SCNiLEAST64, 430
SCNiLEAST8, 431
SCNiMAX, 431
SCNiPTR, 431
SCNo16, 431
SCNo32, 431
SCNo64, 431

SCNo8, 431
 SCNoFAST16, 431
 SCNoFAST32, 431
 SCNoFAST64, 432
 SCNoFAST8, 432
 SCNoLEAST16, 432
 SCNoLEAST32, 432
 SCNoLEAST64, 432
 SCNoLEAST8, 432
 SCNoMAX, 432
 SCNoPTR, 432
 SCNu16, 432
 SCNu32, 433
 SCNu64, 433
 SCNu8, 433
 SCNuFAST16, 433
 SCNuFAST32, 433
 SCNuFAST64, 433
 SCNuFAST8, 433
 SCNuLEAST16, 433
 SCNuLEAST32, 433
 SCNuLEAST64, 434
 SCNuLEAST8, 434
 SCNuMAX, 434
 SCNuPTR, 434
 SCNx16, 434
 SCNx32, 434
 SCNx64, 434
 SCNx8, 434
 SCNxFAST16, 434
 SCNxFAST32, 435
 SCNxFAST64, 435
 SCNxFAST8, 435
 SCNxLEAST16, 435
 SCNxLEAST32, 435
 SCNxLEAST64, 435
 SCNxLEAST8, 435
 SCNxMAX, 435
 SCNxPTR, 435
 strtoumax, 436
 strtoumax, 437
 INVALID_SOCKET
 socket.h, 575
 invalidate_dcache_all
 cache.h, 279
 invalidate_dcache_range
 cache.h, 279
 invalidate_icache_all
 cache.h, 280
 ioctl
 iolib.h, 455
 ioGlobalStdSet
 iolib.h, 456
 iolib.h, 438
 BD_STD_SIGNATURE, 441
 BLK_DEV, 445
 cd, 447
 chkDsk, 447
 close, 448
 creat, 448
 creat_empty, 448
 DCB, 445
 DEV_HDR, 445
 DEVICE, 446
 dirExists, 449
 FA_ARCH, 441
 FA_CAT, 441
 FA_HIDE, 441
 FA_LABEL, 441
 FA_RO, 441
 FA_SYSTEM, 442
 FCB, 446
 FFBLK, 446
 fileExists, 449
 fileSize, 450
 findDev, 450
 findFCB, 450
 findFd, 451
 findFirst, 451
 findNext, 451
 FIOCD, 442
 FIOCHKDSK, 442
 FIODISKFORMAT, 442
 FIOEOF, 442
 FIOFILESIZE, 442
 FIOFINDFIRST, 442
 FIOFINDNEXT, 442
 FIOFLUSH, 442
 FIOFREESIZE, 443
 FIOGETDT, 443
 FIOGETLABEL, 443
 FIOGETLNAME, 443
 FIOGETTO, 443
 FIOMKD, 443
 FIORENAME, 443
 FIORESETDRV, 443
 FIORMD, 443
 FIOSEEK, 444
 FIOSETDT, 444
 FIOSETTO, 444
 FIOTELL, 444
 FIOUNMOUNT, 444
 FIOUPD, 444
 FIOWRKDIR, 444
 flush, 452
 formatVolume, 452
 freeFCB, 452
 freeSpace, 453
 getFilesInDir, 453
 getFullFileName, 453
 getLongName, 454
 getVolumeLabel, 454
 getWorkDir, 455
 getWorkDir_s, 455

ioctl, 455
 ioGlobalStdSet, 456
 iosDevAdd, 456
 iosDevCloseProc, 446
 iosDevCreateProc, 446
 iosDevIoctlProc, 446
 iosDevOpenProc, 446
 iosDevRdProc, 446
 iosDevRemove, 457
 iosDevRemoveProc, 446
 iosDevShow, 457
 iosDevWrProc, 447
 iosDrvInstall, 457
 iosFdShow, 458
 ioTaskStdSet, 458
 lseek, 459
 mkdir, 459
 mountVolume, 460
 O_BINARY, 444
 O_CAT, 444
 O_CREAT, 445
 O_FIXED, 445
 O_RDONLY, 445
 O_RDWR, 445
 O_TRUNC, 445
 O_WRONLY, 445
 open, 460
 P_FCB, 447
 read, 461
 remove, 461
 removeDevice, 462
 reset, 462
 rmdir, 462
 seek, 463
 SEEKBLK, 447
 sysDeviceList, 465
 tell, 463
 unloadVolume, 464
 upd, 464
 write, 464
 iosDevAdd
 iolib.h, 456
 iosDevCloseProc
 iolib.h, 446
 iosDevCreateProc
 iolib.h, 446
 iosDevIoctlProc
 iolib.h, 446
 iosDevOpenProc
 iolib.h, 446
 iosDevRdProc
 iolib.h, 446
 iosDevRemove
 iolib.h, 457
 iosDevRemoveProc
 iolib.h, 446
 iosDevShow
 iolib.h, 457
 iosDevWrProc
 iolib.h, 447
 iosDrvInstall
 iolib.h, 457
 iosFdShow
 iolib.h, 458
 ioTaskStdSet
 iolib.h, 458
 IP_PACKET
 udp.h, 747
 ip_packet, 138
 d_addr, 138
 data, 138
 length, 138
 protocol, 138
 s_addr, 138
 iProduct
 usb_device_descriptor, 216
 iptostr
 udp.h, 748
 irq
 intlib.h, 415
 irq_act_len
 usb_device, 213
 irq_handle
 usb_device, 213
 IRQ_HANDLED
 intlib.h, 413
 IRQ_NONE
 intlib.h, 413
 irq_q
 usb_device, 213
 irq_status
 usb_device, 213
 irqreturn_t
 multex.h, 518
 isalnum
 ctype.h, 314
 isalpha
 ctype.h, 315
 isascii
 ctype.h, 315
 isblank
 ctype.h, 315
 iscntrl
 ctype.h, 316
 isdigit
 ctype.h, 316
 isdigitex
 stdlib.h, 660
 iSerialNumber
 usb_device_descriptor, 216
 isgraph
 ctype.h, 316
 isinf
 math.h, 491

- islower
 - ctype.h, 317
- isnan
 - math.h, 491
- iso646.h, 466
 - and, 466
 - and_eq, 466
 - bitand, 466
 - bitor, 466
 - compl, 466
 - not, 466
 - not_eq, 467
 - or, 467
 - or_eq, 467
 - xor, 467
 - xor_eq, 467
- ISOCHRONOUS
 - usbdescriptors.h, 781
- isprint
 - ctype.h, 317
- ispunct
 - ctype.h, 317
- isspace
 - ctype.h, 318
- isupper
 - ctype.h, 318
- isxdigit
 - ctype.h, 318
- itoa
 - stdlib.h, 661
- jmp_buf, 139
 - REGS, 139
- KERN_DEBUG
 - multex.h, 515
- KERN_INFO
 - multex.h, 515
- kernel.dox, 468
- kernelInit
 - tasklib.h, 697
- kernelTimeSlice
 - tasklib.h, 698
- keyPressed
 - crt.h, 299
- kfree
 - memlib.h, 495
- kill
 - signal.h, 564
- kmalloc
 - memlib.h, 495
- L_tmpnam
 - stdio.h, 633
- labs
 - stdlib.h, 661
- last
 - msgQID, 142
- ldexp
 - math.h, 491
- ldexpf
 - math.h, 491
- ldiv
 - stdlib.h, 661
- ldiv_t, 140
 - quot, 140
 - rem, 140
- left_margin
 - tScreenDeviceMode, 173
- length
 - ip_packet, 138
- LFB
 - Display, 115
- likely
 - multex.h, 515
- limits.h, 469
 - CHAR_BIT, 470
 - CHAR_MAX, 470
 - CHAR_MIN, 470
 - INT_MAX, 470
 - INT_MIN, 471
 - LLONG_MAX, 471
 - LLONG_MIN, 471
 - LONG_MAX, 471
 - LONG_MIN, 471
 - MB_LEN_MAX, 471
 - SCHAR_MAX, 471
 - SCHAR_MIN, 471
 - SHRT_MAX, 471
 - SHRT_MIN, 472
 - UCHAR_MAX, 472
 - UINT_MAX, 472
 - ULLONG_MAX, 472
 - ULONG_MAX, 472
 - USHRT_MAX, 472
- list.h, 473
 - INIT_STATIC_LIST, 474
 - list_AddToBack, 474
 - list_AddToHead, 474
 - list_CreateNewList, 475
 - list_CreateNewNode, 475
 - list_CreateNewNodeWithoutDataCopy, 475
 - list_CutByIndex, 476
 - list_CutByNodePointer, 476
 - list_CutFromBack, 476
 - list_CutFromHead, 477
 - list_DoForeach, 477
 - list_FreeAll, 477
 - list_FreeNodesAndList, 478
 - list_RemoveByIndex, 478
 - list_RemoveByNodePointer, 478
 - list_RemoveFromBack, 479
 - list_RemoveFromHead, 479
 - NODE_FIELD, 474
 - sListNode, 474

[list_AddToBack](#)
[list.h, 474](#)

[list_AddToHead](#)
[list.h, 474](#)

[list_CreateNewList](#)
[list.h, 475](#)

[list_CreateNewNode](#)
[list.h, 475](#)

[list_CreateNewNodeWithoutDataCopy](#)
[list.h, 475](#)

[list_CutByIndex](#)
[list.h, 476](#)

[list_CutByNodePointer](#)
[list.h, 476](#)

[list_CutFromBack](#)
[list.h, 476](#)

[list_CutFromHead](#)
[list.h, 477](#)

[list_DoForeach](#)
[list.h, 477](#)

[list_FreeAll](#)
[list.h, 477](#)

[list_FreeNodesAndList](#)
[list.h, 478](#)

[list_RemoveByIndex](#)
[list.h, 478](#)

[list_RemoveByNodePointer](#)
[list.h, 478](#)

[list_RemoveFromBack](#)
[list.h, 479](#)

[list_RemoveFromHead](#)
[list.h, 479](#)

[listen](#)
[socket.h, 578](#)

[listNode, 141](#)
[pData, 141](#)
[pNext, 141](#)
[pPrev, 141](#)

[llabs](#)
[stdlib.h, 662](#)

[LLONG_MAX](#)
[limits.h, 471](#)

[LLONG_MIN](#)
[limits.h, 471](#)

[llrint](#)
[math.h, 492](#)

[llrintf](#)
[math.h, 492](#)

[lltoa](#)
[stdlib.h, 662](#)

[loadBMPSurface](#)
[a20graph.h, 245](#)

[loadFile](#)
[filesyst.h, 355](#)

[loadFileSz](#)
[filesyst.h, 355](#)

[loadFromBitmap](#)
[softgraph.h, 586](#)

[loadFromJPG](#)
[softgraph.h, 587](#)

[loadFromPNG](#)
[softgraph.h, 587](#)

[loadJPEGSurface](#)
[a20graph.h, 246](#)

[loadPNGSurface](#)
[a20graph.h, 246](#)

[loadRawSurface](#)
[a20graph.h, 247](#)

[localtime](#)
[time.h, 711](#)

[localtime_r](#)
[time.h, 712](#)

[log](#)
[math.h, 492](#)

[log10](#)
[math.h, 492](#)

[log10f](#)
[math.h, 492](#)

[log2f](#)
[math.h, 492](#)

[LONG_MAX](#)
[limits.h, 471](#)

[LONG_MIN](#)
[limits.h, 471](#)

[longjmp](#)
[setjmp.h, 551](#)

[longlongToHex](#)
[string.h, 673](#)

[lowercase](#)
[string.h, 673](#)

[lower_margin](#)
[tScreenDeviceMode, 173](#)

[lr](#)
[REG_SET, 144](#)

[lrint](#)
[math.h, 492](#)

[lrintf](#)
[math.h, 492](#)

[lseek](#)
[iolib.h, 459](#)

[LVDS_1024x768](#)
[a20graph.h, 238](#)

[LVDS_1280x800](#)
[a20graph.h, 238](#)

[LVDS_1400x1050](#)
[a20graph.h, 239](#)

[LVDS_158x1920](#)
[a20graph.h, 239](#)

[LVDS_1920x1080](#)
[a20graph.h, 238](#)

[LVDS_1920x165](#)
[a20graph.h, 239](#)

[LVDS_1920x360](#)
[a20graph.h, 238](#)

LVDS_800x480
 a20graph.h, 238
 LVDS_800x600
 a20graph.h, 238
 lvds_param_t
 a20graph.h, 238

 M_LN2
 math.h, 488
 M_PI
 math.h, 488
 M_SQRT2
 math.h, 488
 malloc
 memlib.h, 497
 mallocForOwner
 memlib.h, 497
 manual.dox, 480
 mapAppendInt
 mapstr.h, 482
 mapAppendIntArray
 mapstr.h, 483
 mapAppendString
 mapstr.h, 483
 mapCheckString
 mapstr.h, 484
 mapFind
 mapstr.h, 484
 mapFree
 mapstr.h, 485
 mapPrint
 mapstr.h, 485
 mapRestore
 mapstr.h, 485
 mapRestoreFromEverywhere
 mapstr.h, 485
 mapStore
 mapstr.h, 486
 mapstr.h, 481
 eMapstrValueType, 482
 mapAppendInt, 482
 mapAppendIntArray, 483
 mapAppendString, 483
 mapCheckString, 484
 mapFind, 484
 mapFree, 485
 mapPrint, 485
 mapRestore, 485
 mapRestoreFromEverywhere, 485
 mapStore, 486
 MAPSTR_SIGNATURE, 482
 mapTypeInt, 482
 mapTypeString, 482
 mapTypeUnknown, 482
 newMap, 486
 MAPSTR_SIGNATURE
 mapstr.h, 482

 mapTypeInt
 mapstr.h, 482
 mapTypeString
 mapstr.h, 482
 mapTypeUnknown
 mapstr.h, 482
 marker
 Sem_Id, 148
 TCB, 161
 mask
 tDrvBitGroup, 165
 math.h, 487
 acos, 488
 asin, 488
 atan, 488
 atan2, 488
 atan2f, 488
 atanf, 489
 cabs, 489
 cbrt, 489
 cbrtf, 489
 ceil, 489
 ceilf, 489
 cos, 489
 cosf, 489
 cosh, 490
 exp, 490
 exp2, 490
 exp2f, 490
 expf, 490
 fabs, 490
 fabsf, 490
 floor, 490
 floorf, 491
 fmod, 491
 frac, 491
 frexp, 491
 isinf, 491
 isnan, 491
 ldexp, 491
 ldexpf, 491
 llrint, 492
 llrintf, 492
 log, 492
 log10, 492
 log10f, 492
 log2f, 492
 lrint, 492
 lrintf, 492
 M_LN2, 488
 M_PI, 488
 M_SQRT2, 488
 pi, 493
 pow, 493
 powf, 493
 round, 493
 roundf, 493

sin, [493](#)
 sinf, [493](#)
 sinh, [493](#)
 sqrt, [494](#)
 sqrtf, [494](#)
 tan, [494](#)
 tanh, [494](#)
 trunc, [494](#)
 truncf, [494](#)
 MAX
 multex.h, [515](#)
 MAX_SAFE
 multex.h, [515](#)
 maxAvail
 memlib.h, [498](#)
 maxchild
 usb_device, [213](#)
 maxCnt
 tRingBuffer, [172](#)
 MaxElementsAmount
 sVector, [157](#)
 maxpacketize
 usb_device, [213](#)
 MB_LEN_MAX
 limits.h, [471](#)
 memAvail
 memlib.h, [498](#)
 MEMBER_SIZE
 multex.h, [515](#)
 memchr
 string.h, [674](#)
 memcmp
 string.h, [674](#)
 memcpy
 string.h, [675](#)
 memlib.h, [495](#)
 _free, [495](#)
 calloc, [496](#)
 free, [496](#)
 initMemLib, [497](#)
 kfree, [495](#)
 kmalloc, [495](#)
 malloc, [497](#)
 mallocForOwner, [497](#)
 maxAvail, [498](#)
 memAvail, [498](#)
 memReport, [498](#)
 mfree, [499](#)
 minfo, [499](#)
 realloc, [499](#)
 memmove
 string.h, [675](#)
 memReport
 memlib.h, [498](#)
 memscan
 string.h, [676](#)
 memset
 string.h, [676](#)
 merge
 string.h, [677](#)
 mf
 usb_device, [213](#)
 mfree
 memlib.h, [499](#)
 MIN
 multex.h, [516](#)
 MIN_SAFE
 multex.h, [516](#)
 minfo
 memlib.h, [499](#)
 Minute
 date_time, [111](#)
 dtcompact, [117](#)
 mkdir
 iolib.h, [459](#)
 MKSINSEC
 sleep.h, [569](#)
 mktime
 time.h, [712](#)
 mobile_direct
 usb_class_descriptor, [184](#)
 mobile_direct_detail
 usb_class_descriptor, [185](#)
 mode
 Display, [115](#)
 file_fcb, [125](#)
 MODE_1024x768
 a20graph.h, [239](#)
 MODE_1280x1024
 a20graph.h, [239](#)
 MODE_1280x720
 a20graph.h, [239](#)
 MODE_1280x768
 a20graph.h, [239](#)
 MODE_1280x800
 a20graph.h, [239](#)
 MODE_1368x768
 a20graph.h, [239](#)
 MODE_1920x1080
 a20graph.h, [239](#)
 MODE_640x480
 a20graph.h, [239](#)
 MODE_800x480
 a20graph.h, [239](#)
 MODE_800x600
 a20graph.h, [239](#)
 MODE_TV
 a20graph.h, [239](#)
 modeline
 Display, [115](#)
 Month
 date_time, [111](#)
 dtcompact, [117](#)
 mountVDisk

- vdisk.h, 786
- mountVolume
 - iolib.h, 460
- move
 - filesyst.h, 355
- mpeg4codec.h, 501
 - freeMpeg4FrameMem, 501
 - getMpeg4InptFrameBuff, 501
 - getMpeg4OutFrame, 502
 - mpeg4DecodeBlock, 502
 - mpeg4InitDecoder, 502
- mpeg4DecodeBlock
 - mpeg4codec.h, 502
- mpeg4InitDecoder
 - mpeg4codec.h, 502
- MSG_DONTROUTE
 - socket.h, 575
- MSG_EOF
 - socket.h, 575
- MSG_EOR
 - socket.h, 575
- MSG_OOB
 - socket.h, 575
- MSG_PEEK
 - socket.h, 575
- MSG_PRI_NORMAL
 - msgqlib.h, 505
- MSG_PRI_URGENT
 - msgqlib.h, 505
- MSG_Q_FIFO
 - msgqlib.h, 505
- MSG_Q_ID
 - msgqlib.h, 506
- MSG_Q_PRIORITY
 - msgqlib.h, 505
- msgQCreate
 - msgqlib.h, 506
- msgQDelete
 - msgqlib.h, 506
- msgQID, 142
 - count, 142
 - F, 142
 - first, 142
 - last, 142
 - p_rd, 142
 - p_wr, 142
 - PW_Id, 142
 - Sem_R, 143
 - Sem_W, 143
 - size, 143
- msgqlib.h, 504
 - MSG_PRI_NORMAL, 505
 - MSG_PRI_URGENT, 505
 - MSG_Q_FIFO, 505
 - MSG_Q_ID, 506
 - MSG_Q_PRIORITY, 505
 - msgQCreate, 506
 - msgQDelete, 506
 - msgQNumMsgs, 507
 - msgQReceive, 507
 - msgQSend, 508
- msgQNumMsgs
 - msgqlib.h, 507
- msgQReceive
 - msgqlib.h, 507
- msgQSend
 - msgqlib.h, 508
- MSINSEC
 - sleep.h, 569
- multex.h, 510
 - _MULTEX_, 511
 - _ALIGN_MASK, 511
 - _LITTLE_ENDIAN, 511
 - _be32_to_cpu, 511
 - ALIGN, 511
 - ALLOC_ALIGN_BUFFER, 512
 - ALLOC_CACHE_ALIGN_BUFFER, 512
 - ARCH_DMA_MINALIGN, 512
 - ARRAY_SIZE, 512
 - clamp, 512
 - debug_cond, 513
 - DEFINE_ALIGN_BUFFER, 513
 - DEFINE_CACHE_ALIGN_BUFFER, 513
 - DIV_ROUND, 514
 - DIV_ROUND_CLOSEST, 514
 - DIV_ROUND_UP, 514
 - dma_addr_t, 518
 - ERROR, 519
 - FUNCPTR, 518
 - get_unaligned, 514
 - INIT_TASK_NAME, 515
 - irqreturn_t, 518
 - KERN_DEBUG, 515
 - KERN_INFO, 515
 - likely, 515
 - MAX, 515
 - MAX_SAFE, 515
 - MEMBER_SIZE, 515
 - MIN, 516
 - MIN_SAFE, 516
 - OK, 518
 - printk, 516
 - put_unaligned, 516
 - resource_size_t, 518
 - ROUND, 517
 - roundup, 517
 - SHELL_NAME, 517
 - SHELL_PRIORITY, 517
 - START_ONCE, 517
 - START_ONCE_R, 517
 - STATUS, 518
 - unlikely, 518
 - VX_SUPERVISOR_MODE, 518
- multi_channel

usb_class_descriptor, 185
 multimedia.dox, 520
 mux
 tDrvGpio, 166
 MX_FP_TASK
 tasklib.h, 695

 n
 tDrvBit, 164
 tDrvBitGroup, 165
 name
 env_var, 119
 TCB, 161
 names.h, 521
 _findSTName, 521
 appendSymbol, 521
 createDynSymTbl, 522
 registerSymTbl, 522
 net.dox, 523
 network_channel
 usb_class_descriptor, 185
 newAVIFile
 avilib.h, 269
 newAVIFileSnd
 avilib.h, 270
 newMap
 mapstr.h, 486
 newRingBuffer
 ringbuffer.h, 536
 newSurface
 a20graph.h, 247
 Next
 TCB, 161
 next
 device_header, 113
 env_var, 119
 exit_st, 120
 udp_service, 176
 no_of_ep
 usb_interface, 222
 no_of_if
 usb_config, 206
 NO_WAIT
 semplib.h, 543
 noAlign
 fontdefines.h, 377
 NODE_FIELD
 list.h, 474
 nodesleep
 sleep.h, 569
 nodetick
 sleep.h, 570
 noreturn
 stdnoreturn.h, 669
 nosound
 crt.h, 299
 not
 iso646.h, 466
 not_eq
 iso646.h, 467
 nSize
 tRingBuffer, 172
 NULL
 stddef.h, 616
 num_altsetting
 usb_interface, 222
 numBlks
 blk_dev, 109
 NumOfNodes
 sList, 153

 O_BINARY
 iolib.h, 444
 O_CAT
 iolib.h, 444
 O_CREAT
 iolib.h, 445
 O_FIXED
 iolib.h, 445
 O_RDONLY
 iolib.h, 445
 O_RDWR
 iolib.h, 445
 O_TRUNC
 iolib.h, 445
 O_WRONLY
 iolib.h, 445
 offsetof
 stddef.h, 616
 OK
 multex.h, 518
 open
 iolib.h, 460
 openAVIFile
 avilib.h, 270
 options
 Sem_Id, 148
 or
 iso646.h, 467
 or_eq
 iso646.h, 467
 OT_INRGB888_PLANAR
 a20graph.h, 230
 OT_MB_INTERLEAVED
 a20graph.h, 230
 OT_MB_PLANAR
 a20graph.h, 230
 OT_MB_UV_COMBINED
 a20graph.h, 230
 OT_PLANAR
 a20graph.h, 230
 OT_UV_COMBINED
 a20graph.h, 230
 OT_YUV420_COMBINED

a20graph.h, [230](#)
 OT_YUV420_INTERLEAVED
 a20graph.h, [231](#)
 OT_YUV420_MB_COMBINED
 a20graph.h, [231](#)
 OT_YUV420_MB_PLANAR
 a20graph.h, [231](#)
 OT_YUV420_PLANAR
 a20graph.h, [231](#)
 OT_YUV422_COMBINED
 a20graph.h, [231](#)
 OT_YUV422_INTERLEAVED
 a20graph.h, [231](#)
 OT_YUV422_MB_COMBINED
 a20graph.h, [231](#)
 OT_YUV422_MB_PLANAR
 a20graph.h, [231](#)
 OT_YUV422_PLANAR
 a20graph.h, [231](#)
 OUT
 usbdescriptors.h, [781](#)
 outCnt
 tRingBuffer, [172](#)
 ovDataFormat_ABGR_1555
 de2.h, [325](#)
 ovDataFormat_ABGR_4444
 de2.h, [325](#)
 ovDataFormat_ABGR_8888
 de2.h, [325](#)
 ovDataFormat_ARGB_1555
 de2.h, [325](#)
 ovDataFormat_ARGB_4444
 de2.h, [325](#)
 ovDataFormat_ARGB_8888
 de2.h, [325](#)
 ovDataFormat_BGR_565
 de2.h, [325](#)
 ovDataFormat_BGR_888
 de2.h, [325](#)
 ovDataFormat_BGRA_4444
 de2.h, [325](#)
 ovDataFormat_BGRA_5551
 de2.h, [325](#)
 ovDataFormat_BGRA_8888
 de2.h, [325](#)
 ovDataFormat_BGRX_8888
 de2.h, [325](#)
 ovDataFormat_RGB_565
 de2.h, [325](#)
 ovDataFormat_RGB_888
 de2.h, [325](#)
 ovDataFormat_RGBA_4444
 de2.h, [325](#)
 ovDataFormat_RGBA_5551
 de2.h, [325](#)
 ovDataFormat_RGBA_8888
 de2.h, [325](#)
 ovDataFormat_RGBX_8888
 de2.h, [325](#)
 ovDataFormat_RGBX_8888
 de2.h, [325](#)
 OverlayClose
 a20graph.h, [248](#)
 OverlayInit
 a20graph.h, [248](#)
 OverlayOpen
 a20graph.h, [248](#)
 OverlayOpenDI
 a20graph.h, [248](#)
 OverlaySetAddr
 a20graph.h, [249](#)
 owner
 Sem_Id, [148](#)
 ownpri
 Sem_Id, [148](#)
 P_A
 gpio.h, [380](#)
 P_B
 gpio.h, [380](#)
 P_C
 gpio.h, [380](#)
 P_D
 gpio.h, [380](#)
 P_E
 gpio.h, [380](#)
 P_F
 gpio.h, [380](#)
 P_FCB
 iolib.h, [447](#)
 P_G
 gpio.h, [380](#)
 P_H
 gpio.h, [381](#)
 P_I
 gpio.h, [381](#)
 p_rd
 msgQID, [142](#)
 p_wr
 msgQID, [142](#)
 PACKET_SIZE_16
 usb.h, [770](#)
 PACKET_SIZE_32
 usb.h, [770](#)
 PACKET_SIZE_64
 usb.h, [770](#)
 PACKET_SIZE_8
 usb.h, [770](#)
 paramBlk
 file_fcb, [125](#)
 parent
 TCB, [161](#)

usb_device, 213
 pBuf
 blk_dev, 109
 pc
 REG_SET, 144
 pData
 listNode, 141
 pDev
 blk_dev, 109
 PF_INET
 socket.h, 575
 pFirst
 sList, 153
 pi
 math.h, 493
 pin
 tDrvGpio, 166
 pipeDevCreate
 pipelib.h, 524
 pipelib.h, 524
 pipeDevCreate, 524
 pixclock_khz
 tScreenDeviceMode, 173
 pixel_seq
 g2d_image, 129
 pLast
 sList, 153
 playMP3File
 sound.h, 602
 playSndBuffer
 sound.h, 602
 playWaveBuffer
 sound.h, 603
 playWaveFile
 sound.h, 603
 pll
 pll.h, 527
 pll.h, 525
 AHB_1, 526
 AHB_2, 526
 APB_1, 526
 APB_2, 526
 pll, 527
 PLL_1, 526
 PLL_2, 526
 PLL_3, 526
 PLL_4, 526
 PLL_5, 526
 PLL_6, 526
 PLL_7, 527
 PLL_8, 527
 PLL_9, 527
 PLL_AUDIO, 527
 PLL_CPU, 527
 PLL_PERIPH0, 527
 PLL_VE, 527
 pllAhbGet, 527
 pllAhbGet, 528
 pllAxiGet, 528
 pllGet, 529
 pllSet, 529
 pllUpdate, 530
 pNext
 listNode, 141
 port
 udp_service, 176
 portnr
 usb_device, 213
 position
 file_fcb, 125
 pow
 math.h, 493
 powf
 math.h, 493
 pPrev
 listNode, 141
 pllAhbGet, 528
 pllAxiGet, 528
 pllGet, 529
 pllSet, 529
 pllUpdate, 530
 PLL_1
 pll.h, 526
 PLL_2
 pll.h, 526
 PLL_3
 pll.h, 526
 PLL_4
 pll.h, 526
 PLL_5
 pll.h, 526
 PLL_6
 pll.h, 526
 PLL_7
 pll.h, 527
 PLL_8
 pll.h, 527
 PLL_9
 pll.h, 527
 PLL_AUDIO
 pll.h, 527
 PLL_CPU
 pll.h, 527
 PLL_PERIPH0
 pll.h, 527
 PLL_VE
 pll.h, 527
 pllAhbGet
 pll.h, 527
 pllAhbGet
 pll.h, 528
 pllAxiGet
 pll.h, 528
 pllGet
 pll.h, 529
 pllSet
 pll.h, 529
 pllUpdate
 pll.h, 530
 pNext
 listNode, 141
 port
 udp_service, 176
 portnr
 usb_device, 213
 position
 file_fcb, 125
 pow
 math.h, 493
 powf
 math.h, 493
 pPrev
 listNode, 141

Prev
 TCB, 161
 PRId16
 inttypes.h, 419
 PRId32
 inttypes.h, 419
 PRId64
 inttypes.h, 419
 PRId8
 inttypes.h, 419
 PRIdFAST16
 inttypes.h, 419
 PRIdFAST32
 inttypes.h, 419
 PRIdFAST64
 inttypes.h, 419
 PRIdFAST8
 inttypes.h, 419
 PRIdLEAST16
 inttypes.h, 419
 PRIdLEAST32
 inttypes.h, 419
 PRIdLEAST64
 inttypes.h, 420
 PRIdLEAST8
 inttypes.h, 420
 PRIdMAX
 inttypes.h, 420
 PRIdPTR
 inttypes.h, 420
 PRIi16
 inttypes.h, 420
 PRIi32
 inttypes.h, 420
 PRIi64
 inttypes.h, 420
 PRIi8
 inttypes.h, 420
 PRIiFAST16
 inttypes.h, 420
 PRIiFAST32
 inttypes.h, 421
 PRIiFAST64
 inttypes.h, 421
 PRIiFAST8
 inttypes.h, 421
 PRIiLEAST16
 inttypes.h, 421
 PRIiLEAST32
 inttypes.h, 421
 PRIiLEAST64
 inttypes.h, 421
 PRIiLEAST8
 inttypes.h, 421
 PRIiMAX
 inttypes.h, 421
 PRIiPTR
 inttypes.h, 421
 print_device_descriptor
 usbdescriptors.h, 781
 printenv
 env_vars.h, 330
 printerr
 stdio.h, 642
 printf
 stdio.h, 643
 printHeader
 shell.h, 554
 printk
 multex.h, 516
 printTasksInfo
 tasklib.h, 698
 PRIo16
 inttypes.h, 422
 PRIo32
 inttypes.h, 422
 PRIo64
 inttypes.h, 422
 PRIo8
 inttypes.h, 422
 PRIoFAST16
 inttypes.h, 422
 PRIoFAST32
 inttypes.h, 422
 PRIoFAST64
 inttypes.h, 422
 PRIoFAST8
 inttypes.h, 422
 PRIoLEAST16
 inttypes.h, 422
 PRIoLEAST32
 inttypes.h, 423
 PRIoLEAST64
 inttypes.h, 423
 PRIoLEAST8
 inttypes.h, 423
 PRIoMAX
 inttypes.h, 423
 PRIoPTR
 inttypes.h, 423
 priority
 TCB, 161
 PRIu16
 inttypes.h, 423
 PRIu32
 inttypes.h, 423
 PRIu64
 inttypes.h, 423
 PRIu8
 inttypes.h, 423
 PRIuFAST16
 inttypes.h, 424
 PRIuFAST32
 inttypes.h, 424

PRIuFAST64
 inttypes.h, 424
 PRIuFAST8
 inttypes.h, 424
 PRIuLEAST16
 inttypes.h, 424
 PRIuLEAST32
 inttypes.h, 424
 PRIuLEAST64
 inttypes.h, 424
 PRIuLEAST8
 inttypes.h, 424
 PRIuMAX
 inttypes.h, 424
 PRIuPTR
 inttypes.h, 425
 PrivateStructPointer
 sTtfFont, 156
 privptr
 usb_device, 214
 PRIx16
 inttypes.h, 425
 PRIx16
 inttypes.h, 425
 PRIx32
 inttypes.h, 425
 PRIx32
 inttypes.h, 425
 PRIx64
 inttypes.h, 425
 PRIx64
 inttypes.h, 425
 PRIx8
 inttypes.h, 425
 PRIx8
 inttypes.h, 425
 PRIxFAST16
 inttypes.h, 426
 PRIxFAST16
 inttypes.h, 426
 PRIxFAST32
 inttypes.h, 426
 PRIxFAST32
 inttypes.h, 426
 PRIxFAST64
 inttypes.h, 426
 PRIxFAST64
 inttypes.h, 426
 PRIxFAST8
 inttypes.h, 426
 PRIxFAST8
 inttypes.h, 426
 PRIxLEAST16
 inttypes.h, 427
 PRIxLEAST16
 inttypes.h, 426
 PRIxLEAST32
 inttypes.h, 427
 PRIxLEAST32
 inttypes.h, 427
 PRIXLEAST64
 inttypes.h, 427
 PRIxLEAST64
 inttypes.h, 427
 PRIXLEAST8
 inttypes.h, 427
 PRIxLEAST8
 inttypes.h, 427
 PRIXMAX
 inttypes.h, 427
 PRIxMAX
 inttypes.h, 427
 PRIXPTR
 inttypes.h, 428
 PRIxPTR
 inttypes.h, 428
 prod
 usb_device, 214
 project.dox, 531
 protocol
 ip_packet, 138
 ps_sp
 TCB, 161
 ps_stack
 TCB, 161
 pTextRect
 fontsdefines.h, 377
 PTRDIFF_MAX
 stdint.h, 623
 PTRDIFF_MIN
 stdint.h, 623
 ptrdiff_t
 stddef.h, 616
 pTtfFont
 fonts.h, 365
 pTtfPrivateFontStruct
 fonts.h, 365
 put_unaligned
 multex.h, 516
 putb
 console.h, 288
 putbytes
 stdio.h, 643
 putc
 stdio.h, 643
 putchar
 stdio.h, 644
 puti
 console.h, 289
 putl
 console.h, 289
 puts
 stdio.h, 644
 putw

console.h, 289
 PW_Id
 msgQID, 142
 pwm.h, 532
 PWM_0, 532
 PWM_1, 533
 PWM_ACTIVE_HIGH, 533
 PWM_ACTIVE_LOW, 533
 pwmInit, 533
 pwmInitPulse, 533
 pwmPulseDurationCalc, 534
 pwmPulseStart, 534
 pwmSetFillFactor, 535
 PWM_0
 pwm.h, 532
 PWM_1
 pwm.h, 533
 PWM_ACTIVE_HIGH
 pwm.h, 533
 PWM_ACTIVE_LOW
 pwm.h, 533
 pwmInit
 pwm.h, 533
 pwmInitPulse
 pwm.h, 533
 pwmPulseDurationCalc
 pwm.h, 534
 pwmPulseStart
 pwm.h, 534
 pwmSetFillFactor
 pwm.h, 535

 qsort
 stdlib.h, 662
 quot
 div_t, 116
 imaxdiv_t, 132
 ldiv_t, 140

 r10
 REG_SET, 144
 r2
 REG_SET, 144
 r3
 REG_SET, 144
 r4
 REG_SET, 145
 r5
 REG_SET, 145
 r6
 REG_SET, 145
 r7
 REG_SET, 145
 r8
 REG_SET, 145
 r9
 REG_SET, 145
 raise

 signal.h, 564
 rand
 stdlib.h, 663
 RAND_MAX
 stdlib.h, 654
 read
 blk_cache, 107
 iolib.h, 461
 readAVIAudio
 avilib.h, 271
 readAVIFrame
 avilib.h, 271
 readKey
 crt.h, 300
 readKey_Timeout
 crt.h, 300
 realloc
 memlib.h, 499
 recStart
 sound.h, 604
 recStop
 sound.h, 604
 REG_SET, 144
 fp, 144
 lr, 144
 pc, 144
 r10, 144
 r2, 144
 r3, 144
 r4, 145
 r5, 145
 r6, 145
 r7, 145
 r8, 145
 r9, 145
 sp, 145
 registerSymTbl
 names.h, 522
 registerTerminator
 terminator.h, 707
 REGS
 jmp_buf, 139
 rem
 div_t, 116
 imaxdiv_t, 132
 ldiv_t, 140
 remove
 iolib.h, 461
 stdio.h, 644
 removeContentFromDir
 filesyst.h, 356
 removeDevice
 iolib.h, 462
 rename
 stdio.h, 645
 reset
 iolib.h, 462

resource_size_t
 multex.h, 518
 rewind
 stdio.h, 645
 RGB
 softgraph.h, 588
 right_margin
 tScreenDeviceMode, 174
 ringbuffer.h, 536
 deleteRingBuffer, 536
 newRingBuffer, 536
 ringBufferCounter, 537
 ringBufferFlush, 537
 ringBufferRead, 538
 ringBufferWrite, 538
 ringBufferCounter
 ringbuffer.h, 537
 ringBufferFlush
 ringbuffer.h, 537
 ringBufferRead
 ringbuffer.h, 538
 ringBufferWrite
 ringbuffer.h, 538
 rmdir
 iolib.h, 462
 ROUND
 multex.h, 517
 round
 math.h, 493
 roundf
 math.h, 493
 roundup
 multex.h, 517

 s_addr
 in_addr, 133
 ip_packet, 138
 s_err
 TCB, 162
 s_in
 TCB, 162
 s_out
 TCB, 162
 sa_data
 sockaddr, 154
 sa_family
 sockaddr, 154
 sa_family_t
 socket.h, 576
 sa_flags
 sigaction, 150
 sa_handler
 sigaction, 150
 SA_INTERRUPT
 signal.h, 558
 sa_mask
 sigaction, 150

 TCB, 162
 SA_NOCLDSTOP
 signal.h, 558
 SA_ONSTACK
 signal.h, 558
 SA_RESETHAND
 signal.h, 558
 sa_sigaction
 sigaction, 150
 SA_SIGINFO
 signal.h, 558
 safe
 TCB, 162
 scanf
 stdio.h, 645
 SCHAR_MAX
 limits.h, 471
 SCHAR_MIN
 limits.h, 471
 SCI (Camera Sensor Interface), 99
 sciSequence_BGBG
 csi.h, 308
 sciSequence_GBGB
 csi.h, 308
 sciSequence_GRGR
 csi.h, 308
 sciSequence_RGRG
 csi.h, 308
 SCNd16
 inttypes.h, 428
 SCNd32
 inttypes.h, 428
 SCNd64
 inttypes.h, 428
 SCNd8
 inttypes.h, 428
 SCNdFAST16
 inttypes.h, 428
 SCNdFAST32
 inttypes.h, 428
 SCNdFAST64
 inttypes.h, 428
 SCNdFAST8
 inttypes.h, 429
 SCNdLEAST16
 inttypes.h, 429
 SCNdLEAST32
 inttypes.h, 429
 SCNdLEAST64
 inttypes.h, 429
 SCNdLEAST8
 inttypes.h, 429
 SCNdMAX
 inttypes.h, 429
 SCNdPTR
 inttypes.h, 429
 SCNi16

inttypes.h, 429
SCNi32
inttypes.h, 429
SCNi64
inttypes.h, 430
SCNi8
inttypes.h, 430
SCNiFAST16
inttypes.h, 430
SCNiFAST32
inttypes.h, 430
SCNiFAST64
inttypes.h, 430
SCNiFAST8
inttypes.h, 430
SCNiLEAST16
inttypes.h, 430
SCNiLEAST32
inttypes.h, 430
SCNiLEAST64
inttypes.h, 430
SCNiLEAST8
inttypes.h, 431
SCNiMAX
inttypes.h, 431
SCNiPTR
inttypes.h, 431
SCNo16
inttypes.h, 431
SCNo32
inttypes.h, 431
SCNo64
inttypes.h, 431
SCNo8
inttypes.h, 431
SCNoFAST16
inttypes.h, 431
SCNoFAST32
inttypes.h, 431
SCNoFAST64
inttypes.h, 432
SCNoFAST8
inttypes.h, 432
SCNoLEAST16
inttypes.h, 432
SCNoLEAST32
inttypes.h, 432
SCNoLEAST64
inttypes.h, 432
SCNoLEAST8
inttypes.h, 432
SCNoMAX
inttypes.h, 432
SCNoPTR
inttypes.h, 432
SCNu16
inttypes.h, 432
SCNu32
inttypes.h, 433
SCNu64
inttypes.h, 433
SCNu8
inttypes.h, 433
SCNuFAST16
inttypes.h, 433
SCNuFAST32
inttypes.h, 433
SCNuFAST64
inttypes.h, 433
SCNuFAST8
inttypes.h, 433
SCNuLEAST16
inttypes.h, 433
SCNuLEAST32
inttypes.h, 433
SCNuLEAST64
inttypes.h, 434
SCNuLEAST8
inttypes.h, 434
SCNuMAX
inttypes.h, 434
SCNuPTR
inttypes.h, 434
SCNx16
inttypes.h, 434
SCNx32
inttypes.h, 434
SCNx64
inttypes.h, 434
SCNx8
inttypes.h, 434
SCNxFAST16
inttypes.h, 434
SCNxFAST32
inttypes.h, 435
SCNxFAST64
inttypes.h, 435
SCNxFAST8
inttypes.h, 435
SCNxLEAST16
inttypes.h, 435
SCNxLEAST32
inttypes.h, 435
SCNxLEAST64
inttypes.h, 435
SCNxLEAST8
inttypes.h, 435
SCNxMAX
inttypes.h, 435
SCNxPTR
inttypes.h, 435
SCREEN
a20graph.h, 232
Screen

- Display, 115
- screenType_Lcd_480x272
 - de2.h, 326
- screenType_Lcd_800x480
 - de2.h, 326
- screenType_TM043NBH02
 - de2.h, 326
- screenType_TM070RxH10
 - de2.h, 326
- scSemB
 - semLib.h, 544
- scSemC
 - semLib.h, 544
- scSemM
 - semLib.h, 544
- SD_BOTH
 - socket.h, 575
- SD_RECEIVE
 - socket.h, 575
- SD_SEND
 - socket.h, 576
- sDisplayInfo, 146
 - bpp, 146
 - height, 146
 - width, 146
- Sec2
 - dtcompact, 117
- Second
 - date_time, 111
- seek
 - ioLib.h, 463
- SEEK_CUR
 - stdio.h, 633
- SEEK_END
 - stdio.h, 633
- SEEK_SET
 - stdio.h, 633
- SEEKBLK
 - ioLib.h, 447
- seekblk, 147
 - seektype, 147
 - shift, 147
- seekToFirstVideoFrame
 - avilib.h, 271
- seektype
 - seekblk, 147
- sem
 - TCB, 162
- SEM_B_STATE
 - semLib.h, 544
- SEM_CLASS
 - semLib.h, 544
- SEM_DELETE_SAFE
 - semLib.h, 543
- SEM_EMPTY
 - semLib.h, 544
- SEM_FLUSH_STATE
 - semLib.h, 545
- semLib.h, 545
- SEM_FULL
 - semLib.h, 544
- SEM_ID
 - semLib.h, 544
- Sem_Id, 148
 - count, 148
 - flushed, 148
 - marker, 148
 - options, 148
 - owner, 148
 - ownpri, 148
 - semclass, 148
 - state, 149
- SEM_INVERSION_SAFE
 - semLib.h, 543
- SEM_MARKER
 - semLib.h, 543
- SEM_Q_FIFO
 - semLib.h, 543
- SEM_Q_PRIORITY
 - semLib.h, 543
- Sem_R
 - msgQID, 143
- Sem_W
 - msgQID, 143
- semBCreate
 - semLib.h, 545
- semBCreate_Default
 - semLib.h, 546
- semCCount
 - semLib.h, 546
- semCCreate
 - semLib.h, 546
- semclass
 - Sem_Id, 148
- semDelete
 - semLib.h, 547
- semFlush
 - semLib.h, 547
- semGive
 - semLib.h, 548
- semLib.h, 540
 - INIT_STATIC_MUTEX, 542
 - INIT_STATIC_MUTEX_DEFAULT, 542
 - INIT_STATIC_SEM, 542
 - INIT_STATIC_SEM_DEFAULT, 542
 - NO_WAIT, 543
 - scSemB, 544
 - scSemC, 544
 - scSemM, 544
 - SEM_B_STATE, 544
 - SEM_CLASS, 544
 - SEM_DELETE_SAFE, 543
 - SEM_EMPTY, 544
 - SEM_FLUSH_STATE, 545
 - SEM_FULL, 544

SEM_ID, 544
SEM_INVERSION_SAFE, 543
SEM_MARKER, 543
SEM_Q_FIFO, 543
SEM_Q_PRIORITY, 543
semBCreate, 545
semBCreate_Default, 546
semCCount, 546
semCCreate, 546
semDelete, 547
semFlush, 547
semGive, 548
semMCreate, 548
semMCreate_Default, 549
semMCUnblock, 549
semTake, 550
sfsFlush, 545
sfsNoFlush, 545
sfsUnblocked, 545
WAIT_FOREVER, 543
semMCreate
semMCreate_Default
semMCreate_Default
semMCUnblock
semMCUnblock
semTake
semTake
serial
serial
usb_device, 214
service
service
udp_service, 176
set_lvds_mode
set_lvds_mode
a20graph.h, 249
set_lvds_param
set_lvds_param
a20graph.h, 249
setAVIType
setAVIType
avilib.h, 272
setbuf
setbuf
stdio.h, 646
setenv
setenv
stdlib.h, 663
setexit
setexit
stdlib.h, 663
setjmp
setjmp
setjmp.h, 551
setjmp.h, 551
longjmp, 551
setjmp, 551
setMasterVol
setMasterVol
sound.h, 604
setOverlayPriority
setOverlayPriority
a20graph.h, 250
setPrimaryIpAddress
setPrimaryIpAddress
udp.h, 749
setSndFmt
setSndFmt
sound.h, 604
setsockopt
setsockopt
socket.h, 579
setTime
setTime
datetime.h, 323
setvbuf
setvbuf
stdio.h, 646
setVideoMode
setVideoMode
a20graph.h, 250
setWorkDevice
setWorkDevice
filesyst.h, 356
fnames.h, 363
sfBar
sfBar
softgraph.h, 588
sfBitBlt
sfBitBlt
softgraph.h, 589
sfBitBltAlphaColor
sfBitBltAlphaColor
softgraph.h, 589
sfBPP
sfBPP
tagSURFACE, 158
sfCircle
sfCircle
softgraph.h, 590
sfClone
sfClone
softgraph.h, 590
sfCloneSample
sfCloneSample
softgraph.h, 591
sfCloneZone
sfCloneZone
softgraph.h, 591
sfCopy
sfCopy
softgraph.h, 592
sfCurvedRectangle
sfCurvedRectangle
softgraph.h, 592
sfCurvedRectangleTest
sfCurvedRectangleTest
softgraph.h, 593
sfData
sfData
tagSURFACE, 158
sfDraw
sfDraw
softgraph.h, 593
sfDrawPolygon
sfDrawPolygon
softgraph.h, 594
sfDrawTransparent
sfDrawTransparent
softgraph.h, 594
sfFilledCircle
sfFilledCircle
softgraph.h, 595
sfGetPixel
sfGetPixel
softgraph.h, 595
sfHeight
sfHeight
tagSURFACE, 158
sfLine
sfLine
softgraph.h, 596
sfLineTo
sfLineTo
softgraph.h, 596
sfMoveTo
sfMoveTo
softgraph.h, 596
sfPutPixel
sfPutPixel
softgraph.h, 597
sfRectangle
sfRectangle
softgraph.h, 597

sfsFlush
 semlib.h, 545
 sfSmoothCircle
 softgraph.h, 598
 sfsNoFlush
 semlib.h, 545
 sfStretchBlt
 softgraph.h, 598
 sfsUnblocked
 semlib.h, 545
 sfWidth
 tagSURFACE, 158
 sfX
 tagSURFACE, 158
 sfY
 tagSURFACE, 158
 sh
 TCB, 162
 shell
 shell.h, 554
 shell.dox, 553
 shell.h, 554
 printHeader, 554
 shell, 554
 shellTask, 554
 SHELL_NAME
 multex.h, 517
 SHELL_PRIORITY
 multex.h, 517
 shellTask
 shell.h, 554
 shift
 seekblk, 147
 shortToHex
 string.h, 677
 SHRT_MAX
 limits.h, 471
 SHRT_MIN
 limits.h, 472
 shutdown
 socket.h, 579
 SI_ASYNCIO
 signal.h, 558
 si_code
 siginfo, 151
 SI_KILL
 signal.h, 559
 SI_MESGQ
 signal.h, 559
 SI_QUEUE
 signal.h, 559
 si_signo
 siginfo, 151
 SI_SYNC
 signal.h, 559
 SI_TIMER
 signal.h, 559
 si_value
 siginfo, 151
 SIG_ATOMIC_MAX
 stdint.h, 623
 SIG_ATOMIC_MIN
 stdint.h, 624
 sig_atomic_t
 signal.h, 563
 SIG_BLOCK
 signal.h, 559
 SIG_DFL
 signal.h, 559
 SIG_ERR
 signal.h, 559
 sig_handle
 signal.h, 564
 SIG_IGN
 signal.h, 559
 SIG_SETMASK
 signal.h, 560
 SIG_UNBLOCK
 signal.h, 560
 SIGABRT
 signal.h, 560
 sigaction, 150
 sa_flags, 150
 sa_handler, 150
 sa_mask, 150
 sa_sigaction, 150
 signal.h, 565
 sigaddset
 signal.h, 565
 SIGALRM
 signal.h, 560
 SIGBUS
 signal.h, 560
 SIGCHLD
 signal.h, 560
 SIGCONT
 signal.h, 560
 sigdelset
 signal.h, 566
 sigemptyset
 signal.h, 566
 SIGEMT
 signal.h, 560
 sigfillset
 signal.h, 566
 SIGFMT
 signal.h, 560
 SIGFPE
 signal.h, 561
 SIGHUP
 signal.h, 561
 SIGILL
 signal.h, 561
 siginfo, 151

- si_code, 151
- si_signo, 151
- si_value, 151
- siginfo_t
 - signal.h, 564
- SIGINT
 - signal.h, 561
- sigismember
 - signal.h, 567
- SIGKILL
 - signal.h, 561
- signa
 - FILE, 123
- signal
 - signal.h, 567
 - TCB, 162
- signal.h, 556
 - kill, 564
 - raise, 564
 - SA_INTERRUPT, 558
 - SA_NOCLDSTOP, 558
 - SA_ONSTACK, 558
 - SA_RESETHAND, 558
 - SA_SIGINFO, 558
 - SI_ASYNCIO, 558
 - SI_KILL, 559
 - SI_MESGQ, 559
 - SI_QUEUE, 559
 - SI_SYNC, 559
 - SI_TIMER, 559
 - sig_atomic_t, 563
 - SIG_BLOCK, 559
 - SIG_DFL, 559
 - SIG_ERR, 559
 - sig_handle, 564
 - SIG_IGN, 559
 - SIG_SETMASK, 560
 - SIG_UNBLOCK, 560
 - SIGABRT, 560
 - sigaction, 565
 - sigaddset, 565
 - SIGALRM, 560
 - SIGBUS, 560
 - SIGCHLD, 560
 - SIGCONT, 560
 - sigdelset, 566
 - sigemptyset, 566
 - SIGEMT, 560
 - sigfillset, 566
 - SIGFMT, 560
 - SIGFPE, 561
 - SIGHUP, 561
 - SIGILL, 561
 - siginfo_t, 564
 - SIGINT, 561
 - sigismember, 567
 - SIGKILL, 561
 - signal, 567
 - SIGNONE, 561
 - SIGPIPE, 561
 - SIGPOLL, 561
 - SIGPROF, 561
 - SIGQUIT, 562
 - SIGRTMAX, 562
 - SIGRTMIN, 562
 - SIGSEGV, 562
 - sigset_t, 564
 - SIGSTOP, 562
 - SIGSYS, 562
 - SIGTERM, 562
 - SIGTRAP, 562
 - SIGTSTP, 562
 - SIGTTIN, 563
 - SIGTTOU, 563
 - SIGURG, 563
 - SIGUSR1, 563
 - SIGUSR2, 563
 - SIGVTALRM, 563
 - SIGXCPU, 563
 - SIGXFSZ, 563
 - wait, 567
- SIGNONE
 - signal.h, 561
- SIGPIPE
 - signal.h, 561
- SIGPOLL
 - signal.h, 561
- SIGPROF
 - signal.h, 561
- SIGQUIT
 - signal.h, 562
- SIGRTMAX
 - signal.h, 562
- SIGRTMIN
 - signal.h, 562
- SIGSEGV
 - signal.h, 562
- sigset_t
 - signal.h, 564
- SIGSTOP
 - signal.h, 562
- SIGSYS
 - signal.h, 562
- SIGTERM
 - signal.h, 562
- SIGTRAP
 - signal.h, 562
- SIGTSTP
 - signal.h, 562
- SIGTTIN
 - signal.h, 563
- SIGTTOU
 - signal.h, 563
- SIGURG

- signal.h, 563
- SIGUSR1
 - signal.h, 563
- SIGUSR2
 - signal.h, 563
- sigval, 152
 - sival_int, 152
 - sival_ptr, 152
- SIGVTALRM
 - signal.h, 563
- SIGXCPU
 - signal.h, 563
- SIGXFSZ
 - signal.h, 563
- silentDirCopy
 - filesyst.h, 357
- sin
 - math.h, 493
- sin_addr
 - sockaddr_in, 155
- sin_family
 - sockaddr_in, 155
- sin_port
 - sockaddr_in, 155
- sin_zero
 - sockaddr_in, 155
- sinf
 - math.h, 493
- sinh
 - math.h, 493
- sival_int
 - sigval, 152
- sival_ptr
 - sigval, 152
- size
 - msgQID, 143
- SIZE_MAX
 - stdint.h, 624
- SIZEOF_UDPHDR
 - udp.h, 747
- sleep.h, 569
 - MKSINSEC, 569
 - MSINSEC, 569
 - nodesleep, 569
 - nodetick, 570
 - sleep60, 570
 - sleepmks, 570
 - sleepms, 571
 - tick60, 571
 - tickmks, 571
 - tickms, 572
- sleep60
 - sleep.h, 570
- sleepmks
 - sleep.h, 570
- sleepms
 - sleep.h, 571
- sList, 153
 - NumOfNodes, 153
 - pFirst, 153
 - pLast, 153
- sListNode
 - list.h, 474
- snprintf
 - stdio.h, 647
- SOCK_DGRAM
 - socket.h, 576
- SOCK_RAW
 - socket.h, 576
- SOCK_STREAM
 - socket.h, 576
- SOCK_TYPE_MASK
 - socket.h, 576
- sockaddr, 154
 - sa_data, 154
 - sa_family, 154
- sockaddr_in, 155
 - sin_addr, 155
 - sin_family, 155
 - sin_port, 155
 - sin_zero, 155
- socket
 - socket.h, 579
- socket.h, 573
 - accept, 576
 - AF_INET, 574
 - bind, 577
 - connect, 577
 - getsockopt, 578
 - in_port_t, 576
 - INADDR_ANY, 574
 - inet_addr, 574
 - INVALID_SOCKET, 575
 - listen, 578
 - MSG_DONTROUTE, 575
 - MSG_EOF, 575
 - MSG_EOR, 575
 - MSG_OOB, 575
 - MSG_PEEK, 575
 - PF_INET, 575
 - sa_family_t, 576
 - SD_BOTH, 575
 - SD_RECEIVE, 575
 - SD_SEND, 576
 - setsockopt, 579
 - shutdown, 579
 - SOCK_DGRAM, 576
 - SOCK_RAW, 576
 - SOCK_STREAM, 576
 - SOCK_TYPE_MASK, 576
 - socket, 579
- softgraph.dox, 581
- softgraph.h, 582
 - clearSurface, 584

createScreenSurface, 584
 createSHdr, 585
 createSurface, 585
 emptySurface, 585
 fillSurface, 586
 freeSurface, 586
 fromRGB, 586
 HDCp, 584
 loadFromBitmap, 586
 loadFromJPG, 587
 loadFromPNG, 587
 RGB, 588
 sfBar, 588
 sfBitBlt, 589
 sfBitBltAlphaColor, 589
 sfCircle, 590
 sfClone, 590
 sfCloneSample, 591
 sfCloneZone, 591
 sfCopy, 592
 sfCurvedRectangle, 592
 sfCurvedRectangleTest, 593
 sfDraw, 593
 sfDrawPolygon, 594
 sfDrawTransparent, 594
 sfFilledCircle, 595
 sfGetPixel, 595
 sfLine, 596
 sfLineTo, 596
 sfMoveTo, 596
 sfPutPixel, 597
 sfRectangle, 597
 sfSmoothCircle, 598
 sfStretchBlit, 598
 softGraphConstr, 599
 softGraphConstrUpdate, 599
 softGraphInit, 599
 SURFACE, 584
 softGraphConstr
 softgraph.h, 599
 softGraphConstrUpdate
 softgraph.h, 599
 softGraphInit
 softgraph.h, 599
 sound
 crt.h, 301
 sound.h, 601
 getRecSndBuffer, 602
 playMP3File, 602
 playSndBuffer, 602
 playWaveBuffer, 603
 playWaveFile, 603
 recStart, 604
 recStop, 604
 setMasterVol, 604
 setSndFmt, 604
 soundInit, 605
 stopMP3, 605
 soundInit
 sound.h, 605
 soundt
 crt.h, 301
 sp
 REG_SET, 145
 speed
 usb_device, 214
 spi.h, 606
 SPI_0, 607
 SPI_1, 607
 SPI_2, 607
 SPI_3, 607
 SPI_CS_0, 607
 SPI_CS_1, 607
 SPI_GPIO_ADDITIONAL, 607
 SPI_GPIO_DEFAULT, 607
 spiInit, 608
 spiInitChipSelectLine, 608
 spiSetBitOrderLsbFirst, 609
 spiSetClkFrequency, 609
 spiSetClkInitialHigh, 609
 spiSetClkTrailingEdge, 610
 spiSetCsInitialHigh, 610
 spiTransfer, 611
 SPI_0
 spi.h, 607
 SPI_1
 spi.h, 607
 SPI_2
 spi.h, 607
 SPI_3
 spi.h, 607
 SPI_CS_0
 spi.h, 607
 SPI_CS_1
 spi.h, 607
 SPI_GPIO_ADDITIONAL
 spi.h, 607
 SPI_GPIO_DEFAULT
 spi.h, 607
 spiInit
 spi.h, 608
 spiInitChipSelectLine
 spi.h, 608
 spiSetBitOrderLsbFirst
 spi.h, 609
 spiSetClkFrequency
 spi.h, 609
 spiSetClkInitialHigh
 spi.h, 609
 spiSetClkTrailingEdge
 spi.h, 610
 spiSetCsInitialHigh
 spi.h, 610
 spiTransfer

- spi.h, 611
- sprintf
 - stdio.h, 647
- sqrt
 - math.h, 494
- sqrtf
 - math.h, 494
- srand
 - stdlib.h, 664
- src_image
 - g2d_blt, 126
 - g2d_stretchblt, 131
- src_rect
 - g2d_blt, 126
 - g2d_stretchblt, 131
- sscanf
 - stdio.h, 648
- stack
 - TCB, 162
- stackSize
 - TCB, 163
- start
 - tMapIterators, 171
- START_ONCE
 - multex.h, 517
- START_ONCE_R
 - multex.h, 517
- startBlk
 - blk_dev, 109
 - file_fcb, 125
- startSecond
 - TCB, 163
- state
 - Sem_Id, 149
- static_assert
 - assert.h, 265
- STATUS
 - multex.h, 518
- status
 - usb_device, 214
- stdarg.h, 613
 - va_arg, 613
 - va_copy, 613
 - va_end, 614
 - va_list, 614
 - va_start, 614
- stdbool.h, 615
 - __bool_true_false_are_defined, 615
 - bool, 615
 - false, 615
 - true, 615
- stddef.h, 616
 - __typeof, 617
 - NULL, 616
 - offsetof, 616
 - ptrdiff_t, 616
 - wchar_t, 616
- stderr
 - stdio.h, 652
- stdin
 - stdio.h, 652
- stdint.h, 618
 - INT16_C, 620
 - INT16_MAX, 620
 - INT16_MIN, 620
 - int16_t, 626
 - INT32_C, 620
 - INT32_MAX, 620
 - INT32_MIN, 620
 - int32_t, 626
 - INT64_C, 620
 - INT64_MAX, 620
 - INT64_MIN, 620
 - int64_t, 627
 - INT8_C, 621
 - INT8_MAX, 621
 - INT8_MIN, 621
 - int8_t, 627
 - INT_FAST16_MAX, 621
 - INT_FAST16_MIN, 621
 - int_fast16_t, 627
 - INT_FAST32_MAX, 621
 - INT_FAST32_MIN, 621
 - int_fast32_t, 627
 - INT_FAST64_MAX, 621
 - INT_FAST64_MIN, 621
 - int_fast64_t, 627
 - INT_FAST8_MAX, 622
 - INT_FAST8_MIN, 622
 - int_fast8_t, 627
 - INT_LEAST16_MAX, 622
 - INT_LEAST16_MIN, 622
 - int_least16_t, 627
 - INT_LEAST32_MAX, 622
 - INT_LEAST32_MIN, 622
 - int_least32_t, 627
 - INT_LEAST64_MAX, 622
 - INT_LEAST64_MIN, 622
 - int_least64_t, 627
 - INT_LEAST8_MAX, 622
 - INT_LEAST8_MIN, 623
 - int_least8_t, 628
 - INTMAX_C, 623
 - INTMAX_MAX, 623
 - INTMAX_MIN, 623
 - intmax_t, 628
 - INTPTR_MAX, 623
 - INTPTR_MIN, 623
 - intptr_t, 628
 - PTRDIFF_MAX, 623
 - PTRDIFF_MIN, 623
 - SIG_ATOMIC_MAX, 623
 - SIG_ATOMIC_MIN, 624
 - SIZE_MAX, 624

UINT16_C, 624
 UINT16_MAX, 624
 uint16_t, 628
 UINT32_C, 624
 UINT32_MAX, 624
 uint32_t, 628
 UINT64_C, 624
 UINT64_MAX, 624
 uint64_t, 628
 UINT8_C, 624
 UINT8_MAX, 625
 uint8_t, 628
 UINT_FAST16_MAX, 625
 uint_fast16_t, 628
 UINT_FAST32_MAX, 625
 uint_fast32_t, 628
 UINT_FAST64_MAX, 625
 uint_fast64_t, 629
 UINT_FAST8_MAX, 625
 uint_fast8_t, 629
 UINT_LEAST16_MAX, 625
 uint_least16_t, 629
 UINT_LEAST32_MAX, 625
 uint_least32_t, 629
 UINT_LEAST64_MAX, 625
 uint_least64_t, 629
 UINT_LEAST8_MAX, 625
 uint_least8_t, 629
 UINTMAX_C, 626
 UINTMAX_MAX, 626
 uintmax_t, 629
 UINTPTR_MAX, 626
 uintptr_t, 629
 WCHAR_MAX, 626
 WCHAR_MIN, 626
 WINT_MAX, 626
 WINT_MIN, 626
 stdio.h, 630
 _IOFBF, 632
 _IOLBF, 632
 _IONBF, 632
 BUFSIZ, 632
 clearerr, 633
 EOF, 632
 fclose, 634
 fdopen, 634
 feof, 634
 ferror, 635
 fflush, 635
 fgetc, 635
 fgetpos, 636
 fgets, 636
 FILE_SIGNATURE, 632
 FILENAME_MAX, 632
 fopen, 637
 FOPEN_MAX, 632
 fpos_t, 633
 fprintf, 637
 fputc, 637
 fputs, 638
 fread, 638
 freopen, 639
 fscanf, 639
 fseek, 639
 fsetpos, 640
 ftell, 640
 fwrite, 641
 getc, 641
 getch, 641
 getchar, 642
 gets, 642
 L_tmpnam, 633
 printerr, 642
 printf, 643
 putbytes, 643
 putc, 643
 putchar, 644
 puts, 644
 remove, 644
 rename, 645
 rewind, 645
 scanf, 645
 SEEK_CUR, 633
 SEEK_END, 633
 SEEK_SET, 633
 setbuf, 646
 setvbuf, 646
 snprintf, 647
 sprintf, 647
 sscanf, 648
 stderr, 652
 stdin, 652
 stdout, 652
 TMP_MAX, 633
 ungetc, 648
 vfprintf, 648
 vfscanf, 649
 vprintf, 649
 vscanf, 650
 vsnprintf, 650
 vsprintf, 651
 vsscanf, 651
 stdlib.h, 653
 _Exit, 654
 abort, 655
 abs, 655
 atexit, 655
 atof, 655
 atoi, 656
 atol, 656
 bcd2d, 656
 bsearch, 657
 compareStr, 657
 d2bcd, 658

stricmp, 684
 stristr, 684
 strlcat, 685
 strlcpy, 685
 strlen, 685
 strlwr, 686
 strncat, 686
 strncmp, 687
 strncpy, 687
 strnlen, 688
 strpbrk, 688
 strrchr, 688
 strsep, 689
 strspn, 689
 strstr, 689
 strtok, 690
 strtok_r, 690
 strup, 691
 strxfrm, 691
 upcase, 692
 string_langid
 usb_device, 214
 stristr
 string.h, 684
 strlcat
 string.h, 685
 strlcpy
 string.h, 685
 strlen
 string.h, 685
 strlwr
 string.h, 686
 strncat
 string.h, 686
 strncmp
 string.h, 687
 strncpy
 string.h, 687
 strnlen
 string.h, 688
 strpbrk
 string.h, 688
 strrchr
 string.h, 688
 strsep
 string.h, 689
 strspn
 string.h, 689
 strstr
 string.h, 689
 strtod
 stdlib.h, 664
 strtodf
 stdlib.h, 664
 strtodh
 console.h, 289
 strtoumax
 inttypes.h, 436
 strtouip
 udp.h, 749
 strtok
 string.h, 690
 strtok_r
 string.h, 690
 strtol
 stdlib.h, 665
 strtolf
 stdlib.h, 665
 strtoll
 stdlib.h, 666
 strtoul
 stdlib.h, 666
 strtoull
 stdlib.h, 667
 strtoumax
 inttypes.h, 437
 strup
 string.h, 691
 strxfrm
 string.h, 691
 sTtfFont, 156
 FontColor, 156
 FontOptions, 156
 FontPixelSize, 156
 FontSize, 156
 PrivateStructPointer, 156
 sTtfPrivateFontStruct
 fonts.h, 365
 submit_int_msg
 usb.h, 770
 SURFACE
 softgraph.h, 584
 surface_t
 a20graph.h, 232
 suspendWorkTask
 tasklib.h, 698
 sVector, 157
 CurrentElementsAmount, 157
 Data, 157
 ElementSize, 157
 MaxElementsAmount, 157
 swap_16
 usb.h, 762
 swap_32
 usb.h, 762
 SYMBOL_CODE_TO_ARRAY_INDEX
 fontsdefines.h, 376
 SYMBOL_IS_PRINTED_ASCII
 fontsdefines.h, 376
 sync
 tScreenDeviceMode, 174
 sysClkRateGet
 timer.h, 719
 sysClkRateSet

timer.h, 719
 sysDeviceList
 iolib.h, 465
 system
 stdlib.h, 667

 taBgLight
 crt.h, 297
 taCount
 timer-arm.h, 716
 tagSURFACE, 158
 sfBPP, 158
 sfData, 158
 sfHeight, 158
 sfWidth, 158
 sfX, 158
 sfY, 158
 taInverse
 crt.h, 298
 taLight
 crt.h, 298
 tan
 math.h, 494
 tanh
 math.h, 494
 taNormal
 crt.h, 298
 taSetInterrupt
 timer-arm.h, 716
 TASK_DELAY
 tasklib.h, 695
 TASK_INT_HANDLING
 tasklib.h, 695
 TASK_MARKER
 tasklib.h, 695
 TASK_PEND
 tasklib.h, 695
 TASK_PS_STACK_SIZE
 tasklib.h, 695
 TASK_SEM_EXIT
 tasklib.h, 696
 TASK_SUSPEND
 tasklib.h, 695
 TASK_WDT
 tasklib.h, 695
 taskCreate
 tasklib.h, 699
 taskDelay
 tasklib.h, 699
 taskDelete
 tasklib.h, 700
 taskDeleteForce
 tasklib.h, 700
 taskId
 tasklib.h, 696
 taskIdSelf
 tasklib.h, 700

 taskIdVerify
 tasklib.h, 701
 tasklib.h, 693
 cpuUsage, 697
 deleteWorkTask, 697
 exit_proc, 696
 kernelInit, 697
 kernelTimeSlice, 698
 MX_FP_TASK, 695
 printTasksInfo, 698
 suspendWorkTask, 698
 TASK_DELAY, 695
 TASK_INT_HANDLING, 695
 TASK_MARKER, 695
 TASK_PEND, 695
 TASK_PS_STACK_SIZE, 695
 TASK_SEM_EXIT, 696
 TASK_SUSPEND, 695
 TASK_WDT, 695
 taskCreate, 699
 taskDelay, 699
 taskDelete, 700
 taskDeleteForce, 700
 taskId, 696
 taskIdSelf, 700
 taskIdVerify, 701
 taskLock, 701
 taskName, 701
 taskNameToId, 702
 taskPriority, 702
 taskPriorityGet, 702
 taskPrioritySet, 703
 taskResume, 703
 taskSafe, 703
 taskSpawn, 704
 taskSuspend, 705
 taskSwitch, 705
 taskTcb, 705
 taskUnlock, 706
 taskUnsafe, 706
 tickAnnounce, 706
 tseFlushed, 696
 tseMutexOrCounterError, 696
 tseTakenFromAnotherTask, 696
 tseTimeoutOrNone, 696
 VX_FP_TASK, 696
 taskLock
 tasklib.h, 701
 taskName
 tasklib.h, 701
 taskNameToId
 tasklib.h, 702
 taskPriority
 tasklib.h, 702
 taskPriorityGet
 tasklib.h, 702
 taskPrioritySet

tasklib.h, 703
 taskResume
 tasklib.h, 703
 taskSafe
 tasklib.h, 703
 taskSemExit
 TCB, 163
 taskSpawn
 tasklib.h, 704
 taskSuspend
 tasklib.h, 705
 taskSwitch
 tasklib.h, 705
 taskTcb
 tasklib.h, 705
 taskUnlock
 tasklib.h, 706
 taskUnsafe
 tasklib.h, 706
 taStart
 timer-arm.h, 717
 taStop
 timer-arm.h, 718
 taSystemNumber
 timer-arm.h, 718
 taUnderlined
 crt.h, 298
 TCB, 159
 child, 160
 childstatus, 160
 cursp, 160
 delay, 160
 exit_code, 160
 exit_list, 160
 exitbuf, 160
 flags, 160
 intCounter, 161
 marker, 161
 name, 161
 Next, 161
 parent, 161
 Prev, 161
 priority, 161
 ps_sp, 161
 ps_stack, 161
 s_err, 162
 s_in, 162
 s_out, 162
 sa_mask, 162
 safe, 162
 sem, 162
 sh, 162
 signal, 162
 stack, 162
 stackSize, 163
 startSecond, 163
 taskSemExit, 163
 vfparea, 163
 workTime, 163
 workTimeOverflowCount, 163
 tDrvBit, 164
 addr, 164
 n, 164
 tDrvBitGroup, 165
 addr, 165
 mask, 165
 n, 165
 tDrvGpio, 166
 available, 166
 enabled, 166
 mux, 166
 pin, 166
 telephone_call
 usb_class_descriptor, 185
 telephone_operational
 usb_class_descriptor, 185
 telephone_ringer
 usb_class_descriptor, 185
 tell
 iolib.h, 463
 terminator.h, 707
 doTerminate, 707
 registerTerminator, 707
 textAttr
 crt.h, 302
 textBackground
 crt.h, 302
 textColor
 crt.h, 302
 textRect, 167
 h, 167
 w, 167
 x, 167
 y, 167
 tick60
 sleep.h, 571
 tickAnnounce
 tasklib.h, 706
 tickGet
 timer.h, 720
 tickmks
 sleep.h, 571
 tickms
 sleep.h, 572
 tickSet
 timer.h, 720
 time
 time.h, 713
 time.h, 708
 asctime, 709
 asctime_r, 709
 clock, 710
 clock_t, 709
 CLOCKS_PER_SEC, 708

ctime, 710
 difftime, 710
 gmtime, 711
 gmtime_r, 711
 localtime, 711
 localtime_r, 712
 mktime, 712
 strftime, 712
 time, 713
 time_t, 709
 TIME_UTC, 708
 timespec_get, 713
 time_t
 time.h, 709
 TIME_UTC
 time.h, 708
 timer-arm.h, 715
 taCount, 716
 taSetInterrupt, 716
 taStart, 717
 taStop, 718
 taSystemNumber, 718
 TIMER_0, 715
 TIMER_1, 715
 TIMER_2, 716
 TIMER_3, 716
 TIMER_4, 716
 TIMER_5, 716
 TIMER_SYS, 716
 timer.h, 719
 hard_reset, 719
 sysClkRateGet, 719
 sysClkRateSet, 719
 tickGet, 720
 tickSet, 720
 TIMER_0
 timer-arm.h, 715
 TIMER_1
 timer-arm.h, 715
 TIMER_2
 timer-arm.h, 716
 TIMER_3
 timer-arm.h, 716
 TIMER_4
 timer-arm.h, 716
 TIMER_5
 timer-arm.h, 716
 TIMER_SYS
 timer-arm.h, 716
 timespec, 168
 tv_nsec, 168
 tv_sec, 168
 timespec_get
 time.h, 713
 tm, 169
 tm_hour, 169
 tm_isdst, 169
 tm_mday, 169
 tm_min, 169
 tm_mon, 169
 tm_sec, 169
 tm_wday, 169
 tm_yday, 170
 tm_year, 170
 tm_hour
 tm, 169
 tm_isdst
 tm, 169
 tm_mday
 tm, 169
 tm_min
 tm, 169
 tm_mon
 tm, 169
 tm_sec
 tm, 169
 tm_wday
 tm, 169
 tm_yday
 tm, 170
 tm_year
 tm, 170
 tMapIterators, 171
 end, 171
 start, 171
 TMP_MAX
 stdio.h, 633
 toascii
 ctype.h, 319
 toggle
 usb_device, 214
 tolower
 ctype.h, 319
 toupper
 ctype.h, 319
 tRingBuffer, 172
 data, 172
 delta, 172
 inCnt, 172
 maxCnt, 172
 nSize, 172
 outCnt, 172
 true
 stdbool.h, 615
 trunc
 math.h, 494
 truncf
 math.h, 494
 tScreenDeviceMode, 173
 bpp, 173
 hsync_len, 173
 left_margin, 173
 lower_margin, 173
 pixclock_khz, 173

- right_margin, 174
- sync, 174
- upper_margin, 174
- vmode, 174
- vsync_len, 174
- xres, 174
- yres, 174
- tseFlushed
 - tasklib.h, 696
- tseMutexOrCounterError
 - tasklib.h, 696
- tseTakenFromAnotherTask
 - tasklib.h, 696
- tseTimeoutOrNone
 - tasklib.h, 696
- ttf_CloseFontAfterPrerender
 - fonts.h, 366
- ttf_ConvertFontSize
 - fonts.h, 367
- ttf_FreeFont
 - fonts.h, 367
- ttf_GetTextPixelLength
 - fonts.h, 368
- ttf_GetTextPixelLengthUtf16
 - fonts.h, 368
- ttf_GetTextRect
 - fonts.h, 369
- ttf_GetTextRectUtf16
 - fonts.h, 369
- ttf_KeepFontOpen
 - fonts.h, 366
- ttf_LoadFont
 - fonts.h, 370
- ttf_NoGlyphSaving
 - fonts.h, 366
- ttf_NoPrerender
 - fonts.h, 366
- ttf_NormalOrientation
 - fonts.h, 366
- ttf_PrerenderASCII
 - fonts.h, 366
- ttf_PrerenderDecNumbers
 - fonts.h, 366
- ttf_Print
 - fonts.h, 370
- ttf_PrintRect
 - fonts.h, 371
- ttf_PrintRectNoCheckBorder
 - fonts.h, 371
- ttf_PrintRectUft16
 - fonts.h, 372
- ttf_PrintRectUft16NoCheckBorder
 - fonts.h, 373
- ttf_PrintUtf16
 - fonts.h, 373
- ttf_RotatedOrientation
 - fonts.h, 366
- ttf_SaveGlyphs
 - fonts.h, 366
- ttf_SetDpi
 - fonts.h, 374
- ttf_SetTextOutputType
 - fonts.h, 374
- TTOT_Debug
 - fonts.h, 367
- TTOT_Errors
 - fonts.h, 367
- TTOT_FontLoadingTime
 - fonts.h, 367
- TTOT_None
 - fonts.h, 367
- tv-decoder.h, 721
 - TVD_0, 722
 - TVD_1, 722
 - TVD_2, 722
 - TVD_3, 722
 - tvd_capture_off, 724
 - tvd_capture_on, 724
 - tvd_chnl_t, 722
 - TVD_CVBS, 723
 - tvd_fmt_t, 722
 - tvd_get_status, 724
 - tvd_init, 725
 - tvd_interface_t, 723
 - tvd_irq_all_status_clear, 725
 - tvd_irq_all_status_get, 725
 - tvd_irq_disable, 725
 - tvd_irq_enable, 726
 - tvd_irq_status_clear, 726
 - tvd_irq_status_get, 726
 - TVD_MB_YUV420, 722
 - TVD_NTSC, 723
 - TVD_PAL, 723
 - TVD_PL_YUV420, 722
 - TVD_PL_YUV422, 722
 - TVD_SECAM, 723
 - tvd_set_chroma, 726
 - tvd_set_color, 727
 - tvd_set_fmt, 727
 - tvd_set_height, 727
 - tvd_set_luma, 728
 - tvd_set_width, 728
 - tvd_system_t, 723
 - TVD_YPBPR_I, 723
 - TVD_YPBPR_P, 723
- tv-receiver.h, 729
 - tvrDevCreate, 730
 - tvrDevDelete, 730
 - tvrGetData, 730
 - tvrInit, 731
 - tvrWaitFrame, 731
- tv_nsec
 - timespec, 168
- tv_sec

- timespec, 168
- TVD_0
 - tv-decoder.h, 722
- TVD_1
 - tv-decoder.h, 722
- TVD_2
 - tv-decoder.h, 722
- TVD_3
 - tv-decoder.h, 722
- tvd_capture_off
 - tv-decoder.h, 724
- tvd_capture_on
 - tv-decoder.h, 724
- tvd_chnl_t
 - tv-decoder.h, 722
- TVD_CVBS
 - tv-decoder.h, 723
- tvd_fmt_t
 - tv-decoder.h, 722
- tvd_get_status
 - tv-decoder.h, 724
- tvd_init
 - tv-decoder.h, 725
- tvd_interface_t
 - tv-decoder.h, 723
- tvd_irq_all_status_clear
 - tv-decoder.h, 725
- tvd_irq_all_status_get
 - tv-decoder.h, 725
- tvd_irq_disable
 - tv-decoder.h, 725
- tvd_irq_enable
 - tv-decoder.h, 726
- tvd_irq_status_clear
 - tv-decoder.h, 726
- tvd_irq_status_get
 - tv-decoder.h, 726
- TVD_MB_YUV420
 - tv-decoder.h, 722
- TVD_NTSC
 - tv-decoder.h, 723
- TVD_PAL
 - tv-decoder.h, 723
- TVD_PL_YUV420
 - tv-decoder.h, 722
- TVD_PL_YUV422
 - tv-decoder.h, 722
- TVD_SECAM
 - tv-decoder.h, 723
- tvd_set_chroma
 - tv-decoder.h, 726
- tvd_set_color
 - tv-decoder.h, 727
- tvd_set_fmt
 - tv-decoder.h, 727
- tvd_set_height
 - tv-decoder.h, 727
- tvd_set_luma
 - tv-decoder.h, 728
- tvd_set_width
 - tv-decoder.h, 728
- tvd_system_t
 - tv-decoder.h, 723
- TVD_YPBPR_I
 - tv-decoder.h, 723
- TVD_YPBPR_P
 - tv-decoder.h, 723
- tvrDevCreate
 - tv-receiver.h, 730
- tvrDevDelete
 - tv-receiver.h, 730
- tvrGetData
 - tv-receiver.h, 730
- tvrInit
 - tv-receiver.h, 731
- tvrWaitFrame
 - tv-receiver.h, 731
- type
 - filesystem.h, 357
- uart.h, 733
 - com_port, 734
 - eUartParity, 737
 - UART_0, 735
 - UART_1, 735
 - UART_2, 735
 - UART_3, 735
 - UART_4, 735
 - UART_5, 735
 - UART_6, 735
 - UART_7, 735
 - UART_BAUD_115200, 735
 - UART_BAUD_19200, 736
 - UART_BAUD_230400, 736
 - UART_BAUD_2400, 736
 - UART_BAUD_38400, 736
 - UART_BAUD_460800, 736
 - UART_BAUD_4800, 736
 - UART_BAUD_57600, 736
 - UART_BAUD_7200, 736
 - UART_BAUD_921600, 736
 - UART_BAUD_9600, 737
 - UART_GPIO_ADDITIONAL, 737
 - UART_GPIO_DEFAULT, 737
 - uartEvenParity, 737
 - uartFlush, 738
 - uartInit, 738
 - uartMarkParity, 737
 - uartName, 739
 - uartNoParity, 737
 - uartOddParity, 737
 - uartReadBufferCounter, 739
 - uartReadBufferOverflowCounter, 739
 - uartSetBaudRate, 740

uartSetBitsNumber, 740
uartSetParity, 741
uartSetReadBufferSize, 741
uartSetReadTimeout, 741
uartSetStopBitsNumber, 742
uartSetTextMode, 742
uartSetWaitTxComplete, 743
uartSetWriteBufferSize, 743
uartSetWriteTimeout, 744
uartSpaceParity, 737
uartWriteBufferCounter, 744

UART_0
uart.h, 735

UART_1
uart.h, 735

UART_2
uart.h, 735

UART_3
uart.h, 735

UART_4
uart.h, 735

UART_5
uart.h, 735

UART_6
uart.h, 735

UART_7
uart.h, 735

UART_BAUD_115200
uart.h, 735

UART_BAUD_19200
uart.h, 736

UART_BAUD_230400
uart.h, 736

UART_BAUD_2400
uart.h, 736

UART_BAUD_38400
uart.h, 736

UART_BAUD_460800
uart.h, 736

UART_BAUD_4800
uart.h, 736

UART_BAUD_57600
uart.h, 736

UART_BAUD_7200
uart.h, 736

UART_BAUD_921600
uart.h, 736

UART_BAUD_9600
uart.h, 737

UART_GPIO_ADDITIONAL
uart.h, 737

UART_GPIO_DEFAULT
uart.h, 737

uartEvenParity
uart.h, 737

uartFlush
uart.h, 738

uartInit
uart.h, 738

uartMarkParity
uart.h, 737

uartName
uart.h, 739

uartNoParity
uart.h, 737

uartOddParity
uart.h, 737

uartReadBufferCounter
uart.h, 739

uartReadBufferOverflowCounter
uart.h, 739

uartSetBaudRate
uart.h, 740

uartSetBitsNumber
uart.h, 740

uartSetParity
uart.h, 741

uartSetReadBufferSize
uart.h, 741

uartSetReadTimeout
uart.h, 741

uartSetStopBitsNumber
uart.h, 742

uartSetTextMode
uart.h, 742

uartSetWaitTxComplete
uart.h, 743

uartSetWriteBufferSize
uart.h, 743

uartSetWriteTimeout
uart.h, 744

uartSpaceParity
uart.h, 737

uartWriteBufferCounter
uart.h, 744

uchar.h, 745

char16_t, 745

char32_t, 745

UCHAR_MAX
limits.h, 472

UCS_FLASHDRIVE_IS_CONNECTED
usb.h, 762

UCS_KEYBOARD_IS_CONNECTED
usb.h, 762

UCS_SOMETHING_IS_CONNECTED
usb.h, 762

udp.h, 746

getPrimaryIpAddress, 748

IP_PACKET, 747

iptostr, 748

setPrimaryIpParity, 749

SIZEOF_UDPHDR, 747

strtoip, 749

UDP_HDR, 748

UDP_SERVICE, 748
 udpGetDataPointer, 749
 udpGetUdpHeader, 749
 udpKillServices, 750
 udpRegisterService, 750
 udpSendPacket, 750
 udpServRout, 748
 udp_d_port
 udp_hdr, 175
 UDP_HDR
 udp.h, 748
 udp_hdr, 175
 udp_d_port, 175
 udp_len, 175
 udp_s_port, 175
 udp_sum, 175
 udp_len
 udp_hdr, 175
 udp_s_port
 udp_hdr, 175
 UDP_SERVICE
 udp.h, 748
 udp_service, 176
 next, 176
 port, 176
 service, 176
 udp_sum
 udp_hdr, 175
 udpGetDataPointer
 udp.h, 749
 udpGetUdpHeader
 udp.h, 749
 udpKillServices
 udp.h, 750
 udpRegisterService
 udp.h, 750
 udpSendPacket
 udp.h, 750
 udpServRout
 udp.h, 748
 ugf
 FILE, 123
 UINT16_C
 stdint.h, 624
 UINT16_MAX
 stdint.h, 624
 uint16_t
 stdint.h, 628
 UINT32_C
 stdint.h, 624
 UINT32_MAX
 stdint.h, 624
 uint32_t
 stdint.h, 628
 UINT64_C
 stdint.h, 624
 UINT64_MAX
 stdint.h, 624
 uint64_t
 stdint.h, 628
 UINT8_C
 stdint.h, 624
 UINT8_MAX
 stdint.h, 625
 uint8_t
 stdint.h, 628
 UINT_FAST16_MAX
 stdint.h, 625
 uint_fast16_t
 stdint.h, 628
 UINT_FAST32_MAX
 stdint.h, 625
 uint_fast32_t
 stdint.h, 628
 UINT_FAST64_MAX
 stdint.h, 625
 uint_fast64_t
 stdint.h, 629
 UINT_FAST8_MAX
 stdint.h, 625
 uint_fast8_t
 stdint.h, 629
 UINT_LEAST16_MAX
 stdint.h, 625
 uint_least16_t
 stdint.h, 629
 UINT_LEAST32_MAX
 stdint.h, 625
 uint_least32_t
 stdint.h, 629
 UINT_LEAST64_MAX
 stdint.h, 625
 uint_least64_t
 stdint.h, 629
 UINT_LEAST8_MAX
 stdint.h, 625
 uint_least8_t
 stdint.h, 629
 UINT_MAX
 limits.h, 472
 UINTMAX_C
 stdint.h, 626
 UINTMAX_MAX
 stdint.h, 626
 uintmax_t
 stdint.h, 629
 UINTPTR_MAX
 stdint.h, 626
 uintptr_t
 stdint.h, 629
 uitoa
 stdlib.h, 667
 ULLONG_MAX
 limits.h, 472

- ulltoa
 - stdlib.h, 668
- ULONG_MAX
 - limits.h, 472
- unget
 - FILE, 123
- ungetc
 - stdio.h, 648
- uni2char
 - unicode.h, 756
- unicode.h, 752
 - convert_AsciiToUtf16, 752
 - convert_Cp1251ToUtf16, 752
 - convert_Cp1251ToUtf8, 753
 - convert_Utf16ToAscii, 753
 - convert_Utf16ToCp1251, 754
 - convert_Utf16ToUtf8, 754
 - convert_Utf8ToCp1251, 755
 - convert_Utf8ToUtf16, 755
 - dos2win, 756
 - uni2char, 756
 - win2dos, 757
- union_function
 - usb_class_descriptor, 185
- unlikely
 - multex.h, 518
- unloadVolume
 - iolib.h, 464
- upcase
 - string.h, 692
- upd
 - iolib.h, 464
- upper_margin
 - tScreenDeviceMode, 174
- USB, 100
- usb.h, 758
 - __LITTLE_ENDIAN, 760
 - __swap_16, 760
 - __swap_32, 761
 - ARCH_DMA_MINALIGN, 761
 - CONFIG_USB_EHCI, 761
 - create_pipe, 761
 - default_pipe, 761
 - PACKET_SIZE_16, 770
 - PACKET_SIZE_32, 770
 - PACKET_SIZE_64, 770
 - PACKET_SIZE_8, 770
 - submit_int_msg, 770
 - swap_16, 762
 - swap_32, 762
 - UCS_FLASHDRIVE_IS_CONNECTED, 762
 - UCS_KEYBOARD_IS_CONNECTED, 762
 - UCS_SOMETHING_IS_CONNECTED, 762
 - USB_ALTSETTINGALLOC, 762
 - usb_bulk_msg, 771
 - USB_CNTRL_TIMEOUT, 762
 - usb_connection_status, 772
 - USB_DMA_MINALIGN, 762
 - usb_dotoggle, 762
 - usb_endpoint_halt, 763
 - usb_endpoint_halted, 763
 - usb_endpoint_out, 763
 - usb_endpoint_running, 763
 - usb_gettoggle, 763
 - USB_MAX_DEVICE, 763
 - USB_MAX_HUB, 763
 - USB_MAXALTSETTING, 764
 - USB_MAXCHILDREN, 764
 - USB_MAXCONFIG, 764
 - USB_MAXENDPOINTS, 764
 - USB_MAXINTERFACES, 764
 - usb_packetid, 764
 - usb_pipe_endpdev, 764
 - usb_pipebulk, 764
 - usb_pipecontrol, 765
 - usb_pipedata, 765
 - usb_pipedevice, 765
 - usb_pipeendpoint, 765
 - usb_pipein, 765
 - usb_pipeint, 765
 - usb_pipeisoc, 765
 - usb_pipeout, 765
 - usb_pipeslow, 766
 - usb_pipespeed, 766
 - usb_pipetype, 766
 - usb_rcvbulkpipe, 766
 - usb_rcvctrlpipe, 766
 - usb_rcvdefctrl, 767
 - usb_rcvintpipe, 767
 - usb_rcvisocpipe, 767
 - usb_set_configuration, 772
 - usb_set_interface, 772
 - usb_settoggle, 767
 - usb_sndbulkpipe, 768
 - usb_sndctrlpipe, 768
 - usb_snddefctrl, 768
 - usb_sndintpipe, 768
 - usb_sndisocpipe, 769
 - USB_TIMEOUT_MS, 769
 - USB_UHCI_DEV_ID, 769
 - USB_UHCI_VEND_ID, 769
 - USB_ALTSETTINGALLOC
 - usb.h, 762
 - usb_bulk_msg
 - usb.h, 771
 - usb_class_abstract_control_descriptor, 177
 - bDescriptorSubtype, 177
 - bDescriptorType, 177
 - bFunctionLength, 177
 - bmCapabilities, 177
 - usb_class_atm_networking_descriptor, 178
 - bDescriptorSubtype, 178
 - bDescriptorType, 178
 - bFunctionLength, 178

- bmATMDeviceStatistics, 178
- bmDataCapabilities, 178
- iEndSystemIdentifier, 178
- wMaxVC, 178
- wType2MaxSegmentSize, 178
- wType3MaxSegmentSize, 179
- usb_class_call_management_descriptor, 180
 - bDataInterface, 180
 - bDescriptorSubtype, 180
 - bDescriptorType, 180
 - bFunctionLength, 180
 - bmCapabilities, 180
- usb_class_capi_control_descriptor, 181
 - bDescriptorSubtype, 181
 - bDescriptorType, 181
 - bFunctionLength, 181
 - bmCapabilities, 181
- usb_class_country_selection_descriptor, 182
 - bDescriptorSubtype, 182
 - bDescriptorType, 182
 - bFunctionLength, 182
 - iCountryCodeRelDate, 182
 - wCountryCode0, 182
- usb_class_descriptor, 183
 - abstract_control, 183
 - atm_networking, 183
 - call_management, 183
 - capi_control, 183
 - country_selection, 183
 - descriptor, 184
 - direct_line, 184
 - ethernet_networking, 184
 - extension_unit, 184
 - function, 184
 - generic, 184
 - header_function, 184
 - hid, 184
 - mobile_direct, 184
 - mobile_direct_detail, 185
 - multi_channel, 185
 - network_channel, 185
 - telephone_call, 185
 - telephone_operational, 185
 - telephone_ringer, 185
 - union_function, 185
 - usb_terminal, 185
- usb_class_direct_line_descriptor, 186
 - bDescriptorSubtype, 186
 - bDescriptorType, 186
 - bFunctionLength, 186
- usb_class_ethernet_networking_descriptor, 187
 - bDescriptorSubtype, 187
 - bDescriptorType, 187
 - bFunctionLength, 187
 - bmEthernetStatistics, 187
 - bNumberPowerFilters, 187
 - iMACAddress, 187
 - wMaxSegmentSize, 187
 - wNumberMCFilters, 187
- usb_class_extension_unit_descriptor, 189
 - bChild0, 189
 - bDescriptorSubtype, 189
 - bDescriptorType, 189
 - bEntityId, 189
 - bExtensionCode, 189
 - bFunctionLength, 189
 - iName, 189
- usb_class_function_descriptor, 190
 - bDescriptorSubtype, 190
 - bDescriptorType, 190
 - bFunctionLength, 190
- usb_class_function_descriptor_generic, 191
 - bDescriptorSubtype, 191
 - bDescriptorType, 191
 - bFunctionLength, 191
 - bmCapabilities, 191
- usb_class_header_function_descriptor, 192
 - bcdCDC, 192
 - bDescriptorSubtype, 192
 - bDescriptorType, 192
 - bFunctionLength, 192
- usb_class_hid_descriptor, 193
 - bcdCDC, 193
 - bCountryCode, 193
 - bDescriptorType, 193
 - bDescriptorType0, 193
 - bLength, 193
 - bNumDescriptors, 193
 - wDescriptorLength0, 193
- usb_class_mdIIm_descriptor, 194
 - bcdVersion, 194
 - bDescriptorSubtype, 194
 - bDescriptorType, 194
 - bFunctionLength, 194
 - bGUID, 194
- usb_class_mdIImd_descriptor, 195
 - bDescriptorSubtype, 195
 - bDescriptorType, 195
 - bDetailData, 195
 - bFunctionLength, 195
 - bGuidDescriptorType, 195
- usb_class_multi_channel_descriptor, 196
 - bDescriptorSubtype, 196
 - bDescriptorType, 196
 - bFunctionLength, 196
 - bmCapabilities, 196
- usb_class_network_channel_descriptor, 197
 - bChannelIndex, 197
 - bDescriptorSubtype, 197
 - bDescriptorType, 197
 - bEntityId, 197
 - bFunctionLength, 197
 - bPhysicalInterface, 197
 - iName, 197

- usb_class_protocol_unit_function_descriptor, 198
 - bChild0, 198
 - bDescriptorSubtype, 198
 - bDescriptorType, 198
 - bEntityId, 198
 - bFunctionLength, 198
 - bProtocol, 198
- usb_class_report_descriptor, 199
 - bData, 199
 - bDescriptorType, 199
 - bLength, 199
 - wLength, 199
- usb_class_telephone_call_descriptor, 200
 - bDescriptorSubtype, 200
 - bDescriptorType, 200
 - bFunctionLength, 200
 - bmCapabilities, 200
- usb_class_telephone_operational_descriptor, 201
 - bDescriptorSubtype, 201
 - bDescriptorType, 201
 - bFunctionLength, 201
 - bmCapabilities, 201
- usb_class_telephone_ringer_descriptor, 202
 - bDescriptorSubtype, 202
 - bDescriptorType, 202
 - bFunctionLength, 202
 - bNumRingerPatterns, 202
 - bRingerVolSeps, 202
- usb_class_union_function_descriptor, 203
 - bDescriptorSubtype, 203
 - bDescriptorType, 203
 - bFunctionLength, 203
 - bMasterInterface, 203
 - bSlaveInterface0, 203
- usb_class_usb_terminal_descriptor, 204
 - bChild0, 204
 - bDescriptorSubtype, 204
 - bDescriptorType, 204
 - bEntityId, 204
 - bFunctionLength, 204
 - bInterfaceNo, 204
 - bmOptions, 204
 - bOutInterfaceNo, 204
- USB_CNTL_TIMEOUT
 - usb.h, 762
- usb_config, 206
 - desc, 206
 - if_desc, 206
 - no_of_if, 206
- usb_configuration_descriptor, 207
 - bConfigurationValue, 207
 - bDescriptorType, 207
 - bLength, 207
 - bmAttributes, 207
 - bMaxPower, 208
 - bNumInterfaces, 208
 - iConfiguration, 208
 - wTotalLength, 208
- usb_connection_status
 - usb.h, 772
- usb_descriptor, 209
 - configuration, 209
 - descriptor, 209
 - device, 209
 - endpoint, 209
 - generic, 209
 - interface, 209
 - string, 209
- usb_device, 211
 - act_len, 211
 - children, 211
 - config, 211
 - configno, 211
 - descriptor, 212
 - devname, 212
 - devnum, 212
 - driver, 212
 - epmaxpacketin, 212
 - epmaxpacketout, 212
 - halted, 212
 - have_langid, 212
 - hcd, 212
 - irq_act_len, 213
 - irq_handle, 213
 - irq_q, 213
 - irq_status, 213
 - maxchild, 213
 - maxpacketsize, 213
 - mf, 213
 - parent, 213
 - portnr, 213
 - privptr, 214
 - prod, 214
 - serial, 214
 - speed, 214
 - status, 214
 - string_langid, 214
 - toggle, 214
- usb_device_descriptor, 215
 - bcdDevice, 215
 - bcdUSB, 215
 - bDescriptorType, 215
 - bDeviceClass, 215
 - bDeviceProtocol, 215
 - bDeviceSubClass, 216
 - bLength, 216
 - bMaxPacketSize0, 216
 - bNumConfigurations, 216
 - idProduct, 216
 - idVendor, 216
 - iManufacturer, 216
 - iProduct, 216
 - iSerialNumber, 216
- USB_DMA_MINALIGN

- usb.h, 762
- usb_dotoggle
 - usb.h, 762
- usb_driver.h, 774
 - usb_register_driver, 774
- usb_endpoint_descriptor, 218
 - bDescriptorType, 218
 - bEndpointAddress, 218
 - bInterval, 218
 - bLength, 218
 - bmAttributes, 218
 - wMaxPacketSize, 220
- usb_endpoint_halt
 - usb.h, 763
- usb_endpoint_halted
 - usb.h, 763
- usb_endpoint_out
 - usb.h, 763
- usb_endpoint_running
 - usb.h, 763
- usb_generic_descriptor, 221
 - bDescriptorSubtype, 221
 - bDescriptorType, 221
 - bLength, 221
- usb_gettoggle
 - usb.h, 763
- usb_interface, 222
 - act_altsetting, 222
 - desc, 222
 - ep_desc, 222
 - no_of_ep, 222
 - num_altsetting, 222
- usb_interface_descriptor, 223
 - bAlternateSetting, 223
 - bDescriptorType, 223
 - bInterfaceClass, 223
 - bInterfaceNumber, 223
 - bInterfaceProtocol, 223
 - bInterfaceSubClass, 224
 - bLength, 224
 - bNumEndpoints, 224
 - iInterface, 224
- USB_MAX_DEVICE
 - usb.h, 763
- USB_MAX_HUB
 - usb.h, 763
- USB_MAXALTSETTING
 - usb.h, 764
- USB_MAXCHILDREN
 - usb.h, 764
- USB_MAXCONFIG
 - usb.h, 764
- USB_MAXENDPOINTS
 - usb.h, 764
- USB_MAXINTERFACES
 - usb.h, 764
- usb_packetid
 - usb.h, 764
- usb_pipe_endpdev
 - usb.h, 764
- usb_pipebulk
 - usb.h, 764
- usb_pipecontrol
 - usb.h, 765
- usb_pipedata
 - usb.h, 765
- usb_pipedevice
 - usb.h, 765
- usb_pipeendpoint
 - usb.h, 765
- usb_pipein
 - usb.h, 765
- usb_pipeint
 - usb.h, 765
- usb_pipeisoc
 - usb.h, 765
- usb_pipeout
 - usb.h, 765
- usb_pipeslow
 - usb.h, 766
- usb_pipespeed
 - usb.h, 766
- usb_pipetype
 - usb.h, 766
- usb_rcvbulkpipe
 - usb.h, 766
- usb_rcvctrlpipe
 - usb.h, 766
- usb_rcvdefctrl
 - usb.h, 767
- usb_rcvintpipe
 - usb.h, 767
- usb_rcvisocpipe
 - usb.h, 767
- usb_register_driver
 - usb_driver.h, 774
- usb_set_configuration
 - usb.h, 772
- usb_set_interface
 - usb.h, 772
- usb_settoggle
 - usb.h, 767
- usb_sndbulkpipe
 - usb.h, 768
- usb_sndctrlpipe
 - usb.h, 768
- usb_snddefctrl
 - usb.h, 768
- usb_sndintpipe
 - usb.h, 768
- usb_sndisocpipe
 - usb.h, 769
- USB_ST_ACMF
 - usbdescriptors.h, 781

USB_ST_ATMNF
 usbdescriptors.h, 781
 USB_ST_CCMF
 usbdescriptors.h, 781
 USB_ST_CMF
 usbdescriptors.h, 781
 USB_ST_CS
 usbdescriptors.h, 781
 USB_ST_CSD
 usbdescriptors.h, 782
 USB_ST_CSF
 usbdescriptors.h, 782
 USB_ST_DLMF
 usbdescriptors.h, 782
 USB_ST_DMM
 usbdescriptors.h, 782
 USB_ST_ENF
 usbdescriptors.h, 782
 USB_ST_EUF
 usbdescriptors.h, 782
 USB_ST_HEADER
 usbdescriptors.h, 782
 USB_ST_MCMF
 usbdescriptors.h, 782
 USB_ST_MDLM
 usbdescriptors.h, 782
 USB_ST_MDLMD
 usbdescriptors.h, 783
 USB_ST_NCT
 usbdescriptors.h, 783
 USB_ST_OBEX
 usbdescriptors.h, 783
 USB_ST_PUF
 usbdescriptors.h, 783
 USB_ST_TCLF
 usbdescriptors.h, 783
 USB_ST_TCM
 usbdescriptors.h, 783
 USB_ST_TOMF
 usbdescriptors.h, 783
 USB_ST_TRF
 usbdescriptors.h, 783
 USB_ST_UF
 usbdescriptors.h, 783
 USB_ST_USBTF
 usbdescriptors.h, 784
 USB_ST_WHCM
 usbdescriptors.h, 784
 usb_string_descriptor, 225
 bDescriptorType, 225
 bLength, 225
 wData, 225
 usb_terminal
 usb_class_descriptor, 185
 USB_TIMEOUT_MS
 usb.h, 769
 USB_UHCI_DEV_ID
 usb.h, 769
 USB_UHCI_VEND_ID
 usb.h, 769
 usbdescriptors.h, 776
 BMAATTRIBUTE_RESERVED, 777
 BMAATTRIBUTE_SELF_POWERED, 777
 BULK, 778
 CLASS_BCD_VERSION, 778
 COMMUNICATIONS_ACM_SUBCLASS, 778
 COMMUNICATIONS_ANCM_SUBCLASS, 778
 COMMUNICATIONS_CCM_SUBCLASS, 778
 COMMUNICATIONS_DEVICE_CLASS, 778
 COMMUNICATIONS_DLCM_SUBCLASS, 778
 COMMUNICATIONS_DMM_SUBCLASS, 778
 COMMUNICATIONS_ENCM_SUBCLASS, 778
 COMMUNICATIONS_INTERFACE_CLASS_CONTROL, 779
 COMMUNICATIONS_INTERFACE_CLASS_DATA, 779
 COMMUNICATIONS_INTERFACE_CLASS_VENDOR, 779
 COMMUNICATIONS_MCCM_SUBCLASS, 779
 COMMUNICATIONS_MDLM_SUBCLASS, 779
 COMMUNICATIONS_NO_PROTOCOL, 779
 COMMUNICATIONS_NO_SUBCLASS, 779
 COMMUNICATIONS_OBEX_SUBCLASS, 779
 COMMUNICATIONS_TCM_SUBCLASS, 779
 COMMUNICATIONS_V25TER_PROTOCOL, 780
 COMMUNICATIONS_WHCM_SUBCLASS, 780
 CONTROL, 780
 CS_ENDPOINT, 780
 CS_INTERFACE, 780
 DATA_INTERFACE_CLASS, 780
 DATA_INTERFACE_PROTOCOL_NONE, 780
 DATA_INTERFACE_SUBCLASS_NONE, 780
 IN, 780
 INTERRUPT, 781
 ISOCHRONOUS, 781
 OUT, 781
 print_device_descriptor, 781
 USB_ST_ACMF, 781
 USB_ST_ATMNF, 781
 USB_ST_CCMF, 781
 USB_ST_CMF, 781
 USB_ST_CS, 781
 USB_ST_CSD, 782
 USB_ST_CSF, 782
 USB_ST_DLMF, 782
 USB_ST_DMM, 782
 USB_ST_ENF, 782
 USB_ST_EUF, 782
 USB_ST_HEADER, 782
 USB_ST_MCMF, 782
 USB_ST_MDLM, 782
 USB_ST_MDLMD, 783
 USB_ST_NCT, 783
 USB_ST_OBEX, 783
 USB_ST_PUF, 783
 USB_ST_TCLF, 783
 USB_ST_TCM, 783

- USB_ST_TOMF, 783
- USB_ST_TRF, 783
- USB_ST_UF, 783
- USB_ST_USBTf, 784
- USB_ST_WHCM, 784
- usbman.dox, 785
- USHRT_MAX
 - limits.h, 472
- usr_int_proc
 - intlib.h, 413
- va_arg
 - stdarg.h, 613
- va_copy
 - stdarg.h, 613
- va_end
 - stdarg.h, 614
- va_list
 - stdarg.h, 614
- va_start
 - stdarg.h, 614
- val
 - env_var, 119
- vdisk.h, 786
 - mountVDisk, 786
- ve_close
 - cedrus.h, 287
- ve_open
 - cedrus.h, 287
- vector.h, 787
 - vector_Add, 787
 - vector_AddEmpty, 787
 - vector_Copy, 788
 - vector_Create, 788
 - vector_Duplicate, 789
 - vector_Free, 789
 - vector_Get, 789
 - vector_GetFirst, 790
 - vector_GetLast, 790
 - vector_Remove, 790
 - vector_RemoveLast, 791
- vector_Add
 - vector.h, 787
- vector_AddEmpty
 - vector.h, 787
- vector_Copy
 - vector.h, 788
- vector_Create
 - vector.h, 788
- vector_Duplicate
 - vector.h, 789
- vector_Free
 - vector.h, 789
- vector_Get
 - vector.h, 789
- vector_GetFirst
 - vector.h, 790
- vector_GetLast
 - vector.h, 790
- vector_Remove
 - vector.h, 790
- vector_RemoveLast
 - vector.h, 791
- vector_GetLast
 - vector.h, 790
- vector_Remove
 - vector.h, 790
- vector_RemoveLast
 - vector.h, 791
- vfparea
 - TCB, 163
- vfprintf
 - stdio.h, 648
- vfscanf
 - stdio.h, 649
- VideoMode
 - Display, 115
- VideoModes
 - a20graph.h, 239
- vmode
 - tScreenDeviceMode, 174
- volConfig
 - blk_dev, 109
- vprintf
 - stdio.h, 649
- vscanf
 - stdio.h, 650
- vsnprintf
 - stdio.h, 650
- vsprintf
 - stdio.h, 651
- vsscanf
 - stdio.h, 651
- vsync_len
 - tScreenDeviceMode, 174
- VX_FP_TASK
 - tasklib.h, 696
- VX_SUPERVISOR_MODE
 - multex.h, 518
- w
 - g2d_image, 129
 - g2d_rect, 130
 - iniRect, 137
 - textRect, 167
- wait
 - signal.h, 567
- WAIT_FOREVER
 - semLib.h, 543
- waitVerticalRetrace
 - a20graph.h, 251
- WCHAR_MAX
 - stdint.h, 626
- WCHAR_MIN
 - stdint.h, 626
- wchar_t
 - stddef.h, 616
- wCountryCode0
 - usb_class_country_selection_descriptor, 182
- wData

usb_string_descriptor, 225
wDescriptorLength0
usb_class_hid_descriptor, 193
Width
Display, 115
width
sDisplayInfo, 146
win2dos
unicode.h, 757
WINT_MAX
stdint.h, 626
WINT_MIN
stdint.h, 626
wLength
usb_class_report_descriptor, 199
wMaxPacketSize
usb_endpoint_descriptor, 220
wMaxSegmentSize
usb_class_ethernet_networking_descriptor, 187
wMaxVC
usb_class_atm_networking_descriptor, 178
wNumberMCFilters
usb_class_ethernet_networking_descriptor, 187
workTime
TCB, 163
workTimeOverflowCount
TCB, 163
write
blk_cache, 107
iolib.h, 464
writeAVIFrame
avilib.h, 272
writeSNDFrame
avilib.h, 273
wTotalLength
usb_configuration_descriptor, 208
wType2MaxSegmentSize
usb_class_atm_networking_descriptor, 178
wType3MaxSegmentSize
usb_class_atm_networking_descriptor, 179

x
complex, 110
g2d_rect, 130
iniCoords, 135
iniRect, 137
textRect, 167

xor
iso646.h, 467

xor_eq
iso646.h, 467

xres
tScreenDeviceMode, 174

y
complex, 110
g2d_rect, 130
iniCoords, 135

iniRect, 137
textRect, 167
Year
date_time, 111
dtcompact, 117
yres
tScreenDeviceMode, 174

Драйвера интерфейсов, 101
Мультимедиа, 102
Стандартные типы, 103
Ядро MULTEX-ARM, 104