

BuguRTOS

4.1.0

Создано системой Doxygen 1.8.17

1	Титульная страница	1
2	Алфавитный указатель структур данных	1
2.1	Структуры данных	1
3	Список файлов	2
3.1	Файлы	2
4	Структуры данных	3
4.1	Структура <code>_bgrt_va_wr_t</code>	3
4.1.1	Поля	3
4.2	Структура <code>bgrt_priv_item_t</code>	3
4.2.1	Подробное описание	3
4.2.2	Поля	4
4.3	Структура <code>bgrt_priv_kblock_t</code>	4
4.3.1	Подробное описание	4
4.3.2	Поля	4
4.4	Структура <code>bgrt_priv_kernel_t</code>	5
4.4.1	Подробное описание	5
4.4.2	Поля	5
4.5	Структура <code>bgrt_priv_kstat_t</code>	5
4.5.1	Поля	6
4.6	Структура <code>bgrt_priv_ktimer_t</code>	6
4.6.1	Поля	6
4.7	Структура <code>bgrt_priv_pcounter_t</code>	7
4.7.1	Подробное описание	7
4.7.2	Поля	7
4.8	Структура <code>bgrt_priv_pitem_t</code>	7
4.8.1	Подробное описание	8
4.8.2	Поля	8
4.9	Структура <code>bgrt_priv_proc_t</code>	8
4.9.1	Подробное описание	9
4.9.2	Поля	9
4.10	Структура <code>bgrt_priv_sched_t</code>	11
4.10.1	Подробное описание	11
4.10.2	Поля	11
4.11	Структура <code>bgrt_priv_sync_t</code>	12
4.11.1	Подробное описание	12
4.11.2	Поля	12
4.12	Структура <code>bgrt_priv_uspd_t</code>	13
4.12.1	Поля	13
4.13	Структура <code>bgrt_priv_vic_t</code>	14
4.13.1	Подробное описание	14
4.13.2	Поля	14

4.14 Структура <code>bgrt_priv_vint_t</code>	15
4.14.1 Подробное описание	15
4.14.2 Поля	15
4.15 Структура <code>bgrt_priv_xlist_t</code>	15
4.15.1 Подробное описание	16
4.15.2 Поля	16
5 Файлы	16
5.1 Файл <code>bugurtos/arch/common/atm_cortex_m34_1.h</code>	16
5.1.1 Макросы	16
5.2 Файл <code>bugurtos/arch/common/atm_gen_1.h</code>	17
5.2.1 Макросы	17
5.3 Файл <code>bugurtos/doc/doxygen/bugurt_port.h</code>	18
5.3.1 Макросы	18
5.3.2 Функции	20
5.4 Файл <code>bugurtos/kernel/bugurt.h</code>	22
5.4.1 Подробное описание	24
5.4.2 Макросы	24
5.4.3 Типы	27
5.4.4 Функции	27
5.5 Файл <code>bugurtos/kernel/crit_sec.h</code>	33
5.5.1 Подробное описание	33
5.5.2 Макросы	33
5.5.3 Функции	34
5.6 Файл <code>bugurtos/kernel/default/syscall_api.h</code>	34
5.6.1 Подробное описание	36
5.6.2 Макросы	36
5.7 Файл <code>bugurtos/kernel/default/syscall_routines.h</code>	42
5.7.1 Типы	42
5.7.2 Функции	43
5.8 Файл <code>bugurtos/kernel/index.h</code>	45
5.8.1 Подробное описание	45
5.8.2 Функции	45
5.9 Файл <code>bugurtos/kernel/item.h</code>	46
5.9.1 Подробное описание	46
5.9.2 Макросы	46
5.9.3 Типы	47
5.9.4 Функции	47
5.10 Файл <code>bugurtos/kernel/kernel.h</code>	48
5.10.1 Подробное описание	49
5.10.2 Макросы	49
5.10.3 Типы	49
5.10.4 Функции	49

5.10.5	Переменные	50
5.11	Файл <code>bugurtos/kernel/pcounter.h</code>	51
5.11.1	Подробное описание	51
5.11.2	Макросы	52
5.11.3	Типы	52
5.11.4	Функции	52
5.12	Файл <code>bugurtos/kernel/pitem.h</code>	56
5.12.1	Подробное описание	57
5.12.2	Макросы	57
5.12.3	Типы	57
5.12.4	Функции	57
5.13	Файл <code>bugurtos/kernel/proc.h</code>	59
5.13.1	Подробное описание	62
5.13.2	Макросы	62
5.13.3	Типы	69
5.13.4	Функции	69
5.14	Файл <code>bugurtos/kernel/sched.h</code>	75
5.14.1	Подробное описание	76
5.14.2	Макросы	76
5.14.3	Типы	76
5.14.4	Функции	76
5.15	Файл <code>bugurtos/kernel/sync.h</code>	79
5.15.1	Подробное описание	80
5.15.2	Макросы	80
5.15.3	Типы	81
5.15.4	Функции	81
5.16	Файл <code>bugurtos/kernel/syscall.h</code>	84
5.16.1	Подробное описание	85
5.16.2	Макросы	85
5.16.3	Типы	86
5.16.4	Перечисления	86
5.16.5	Функции	87
5.17	Файл <code>bugurtos/kernel/timer.h</code>	88
5.17.1	Подробное описание	88
5.17.2	Макросы	88
5.17.3	Типы	89
5.17.4	Функции	90
5.18	Файл <code>bugurtos/kernel/vint.h</code>	91
5.18.1	Подробное описание	92
5.18.2	Макросы	92
5.18.3	Типы	92
5.18.4	Функции	92
5.19	Файл <code>bugurtos/kernel/xlist.h</code>	94

5.19.1 Подробное описание	95
5.19.2 Типы	95
5.19.3 Функции	95
Предметный указатель	97

1 Титульная страница

BuguRTOS - ядро операционной системы реального времени. Написано анонимусом ДЛЯ УДО-ВОЛЬСТВИЯ.

Предупреждения

Распространяется под изменённой лицензией GPLv3, смотрите exception.txt.

2 Алфавитный указатель структур данных

2.1 Структуры данных

Структуры данных с их кратким описанием.

_bgrt_va_wr_t	3
bgrt_priv_item_t Элемент 2-связного списка	3
bgrt_priv_kblock_t Блок ядра BuguRTOS	4
bgrt_priv_kernel_t Ядро BuguRTOS	5
bgrt_priv_kstat_t	5
bgrt_priv_ktimer_t	6
bgrt_priv_pcounter_t Счётчик захваченных ресурсов	7
bgrt_priv_pitem_t Элемент списка с приоритетами	7
bgrt_priv_proc_t Процесс	8
bgrt_priv_sched_t Планировщик	11
bgrt_priv_sync_t Базовый примитив синхронизации	12
bgrt_priv_uspd_t	13

bgrt_priv_vic_t	Виртуальный контроллер прерываний	14
bgrt_priv_vint_t	Виртуальное прерывание	15
bgrt_priv_xlist_t	Список с приоритетами	15

3 Список файлов

3.1 Файлы

Полный список файлов.

bugurtos/arch/common/atm_cortex_m34_1.h		16
bugurtos/arch/common/atm_gen_1.h		17
bugurtos/doc/doxygen/bugurt_port.h		18
bugurtos/kernel/bugurt.h	Главный заголовочный файл	22
bugurtos/kernel/crit_sec.h	Заголовок критических секций	33
bugurtos/kernel/index.h	Заголовок функции поиска в бинарном индексе	45
bugurtos/kernel/item.h	Заголовок элементов 2-связного списка	46
bugurtos/kernel/kernel.h	Заголовок Ядра	48
bugurtos/kernel/pcounter.h	Заголовок счётчиков захваченных ресурсов	51
bugurtos/kernel/pitem.h	Заголовок элементов списка с приоритетами	56
bugurtos/kernel/proc.h	Заголовок процессов	59
bugurtos/kernel/sched.h	Заголовок планировщика	75
bugurtos/kernel/sync.h	Заголовок базового примитива синхронизации	79
bugurtos/kernel/syscall.h	Заголовок системных вызовов	84
bugurtos/kernel/timer.h	Заголовок программных таймеров	88

bugurtos/kernel/vint.h Заголовок виртуальных прерываний	91
bugurtos/kernel/xlist.h Заголовок списков с приоритетами	94
bugurtos/kernel/default/syscall_api.h Заголовок системных вызовов	34
bugurtos/kernel/default/syscall_routines.h	42

4 Структуры данных

4.1 Структура `_bgrt_va_wr_t`

```
#include "bugurtos/kernel/syscall.h"
```

Поля данных

- `va_list list`

4.1.1 Поля

4.1.1.1 `list va_list _bgrt_va_wr_t::list`

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/syscall.h](#)

4.2 Структура `bgrt_priv_item_t`

Элемент 2-связного списка.

```
#include "bugurtos/kernel/item.h"
```

Поля данных

- `bgrt_item_t * next`
- `bgrt_item_t * prev`

4.2.1 Подробное описание

Элемент 2-связного списка.

Все структуры, где будут применяться 2-связные списки, унаследуют свойства и методы `bgrt_↔item_t`.

4.2.2 Поля

4.2.2.1 next [bgrt_item_t*](#) bgrt_priv_item_t::next

Следующий элемент.

4.2.2.2 prev [bgrt_item_t*](#) bgrt_priv_item_t::prev

Предыдущий элемент.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/item.h](#)

4.3 Структура bgrt_priv_kblock_t

Блок ядра VuguRTOS.

```
#include "bugurtos/kernel/kernel.h"
```

Поля данных

- [bgrt_map_t](#) [hpmap](#)
- [bgrt_map_t](#) [lpmap](#)

4.3.1 Подробное описание

Блок ядра VuguRTOS.

Отвечает за обработку виртуальных прерываний, обработку системных вызовов, работу планировщика на отдельном процессорном ядре.

4.3.2 Поля

4.3.2.1 [hpmap](#) [bgrt_map_t](#) bgrt_priv_kblock_t::hpmap

Виртуальный контроллер "быстрых" прерываний высокого приоритета.

4.3.2.2 [lpmap](#) [bgrt_map_t](#) bgrt_priv_kblock_t::lpmap

Виртуальный контроллер "быстрых" прерываний низкого приоритета.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/kernel.h](#)

4.4 Структура bgrt_priv_kernel_t

Ядро BuguRTOS.

```
#include "bugurtos/kernel/kernel.h"
```

Поля данных

- [bgrt_kblock_t kblock](#) [BGRT_MAX_CPU]
- [bgrt_sched_t sched](#) [BGRT_MAX_CPU]
- [bgrt_kstat_t stat](#)
- [bgrt_ktimer_t timer](#)

4.4.1 Подробное описание

Ядро BuguRTOS.

В ядре хранится информация о запущенных процессах, процессе(ах) холостого хода.

4.4.2 Поля

4.4.2.1 [kblock](#) [bgrt_kblock_t](#) bgrt_priv_kernel_t::kblock[BGRT_MAX_CPU]

Контроллеры программных прерываний.

4.4.2.2 [sched](#) [bgrt_sched_t](#) bgrt_priv_kernel_t::sched[BGRT_MAX_CPU]

Планировщики для каждого процессорного ядра.

4.4.2.3 [stat](#) [bgrt_kstat_t](#) bgrt_priv_kernel_t::stat

Статистика для балансировки нагрузки, на Hotplug работать не собираемся, все будет статично.

4.4.2.4 [timer](#) [bgrt_ktimer_t](#) bgrt_priv_kernel_t::timer

Системный таймер.

Объявления и описания членов структуры находятся в файле:

- bugurtos/kernel/[kernel.h](#)

4.5 Структура bgrt_priv_kstat_t

```
#include "bugurtos/kernel/sched.h"
```

Поля данных

- `bgrt_ls_t` [val](#) [BGRT_MAX_CPU]
- `bgrt_lock_t` [lock](#)

4.5.1 Поля

4.5.1.1 `lock` `bgrt_lock_t` `bgrt_priv_kstat_t::lock`

Спин-блокировка.

4.5.1.2 `val` `bgrt_ls_t` `bgrt_priv_kstat_t::val`[BGRT_MAX_CPU]

Значения.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/sched.h](#)

4.6 Структура `bgrt_priv_ktimer_t`

```
#include "bugurtos/kernel/timer.h"
```

Поля данных

- `void(* tick)(void)`
- `bgrt_tmr_t` [val](#)
- `bgrt_lock_t` [lock](#)

4.6.1 Поля

4.6.1.1 `lock` `bgrt_lock_t` `bgrt_priv_ktimer_t::lock`

Спин-блокировка.

4.6.1.2 `tick` `void(* bgrt_priv_ktimer_t::tick)(void)`

Хук.

4.6.1.3 `val bgrt_tmr_t bgrt_priv_ktimer_t::val`

Значение.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/timer.h](#)

4.7 Структура `bgrt_priv_pcounter_t`

Счётчик захваченных ресурсов.

```
#include "bugurtos/kernel/pcounter.h"
```

Поля данных

- `bgrt_cnt_t counter` [BGRT_BITS_IN_INDEX_T]
- `bgrt_map_t map`

4.7.1 Подробное описание

Счётчик захваченных ресурсов.

Используется для хранения информации о наследуемых приоритетах.

4.7.2 Поля

4.7.2.1 `counter bgrt_cnt_t bgrt_priv_pcounter_t::counter[BGRT_BITS_IN_INDEX_T]`

Массив счётчиков.

4.7.2.2 `map bgrt_map_t bgrt_priv_pcounter_t::map`

Индекс для ускорения поиска.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/pcounter.h](#)

4.8 Структура `bgrt_priv_pitem_t`

Элемент списка с приоритетами

```
#include "bugurtos/kernel/pitem.h"
```

Поля данных

- [bgrt_item_t](#) parent
- [bgrt_xlist_t](#) * list
- [bgrt_prio_t](#) prio

4.8.1 Подробное описание

Элемент списка с приоритетами

4.8.2 Поля

4.8.2.1 list [bgrt_xlist_t](#)* [bgrt_priv_pitem_t::list](#)

Указатель на список в который будем вставлять.

4.8.2.2 parent [bgrt_item_t](#) [bgrt_priv_pitem_t::parent](#)

Родитель - [bgrt_item_t](#).

4.8.2.3 prio [bgrt_prio_t](#) [bgrt_priv_pitem_t::prio](#)

Приоритет.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/pitem.h](#)

4.9 Структура [bgrt_priv_proc_t](#)

Процесс.

```
#include "bugurtos/kernel/proc.h"
```

Поля данных

- [bgrt_pitem_t](#) parent
- [bgrt_flag_t](#) flags
- [bgrt_prio_t](#) base_prio
- [bgrt_pcounter_t](#) lres
- [bgrt_tmr_t](#) time_quant
- [bgrt_tmr_t](#) timer
- struct [bgrt_priv_sync_t](#) * sync
- [bgrt_cnt_t](#) cnt_lock
- [bgrt_cpuid_t](#) core_id
- [bgrt_aff_t](#) affinity
- [bgrt_lock_t](#) lock
- [bgrt_code_t](#) pmain
- [bgrt_code_t](#) sv_hook
- [bgrt_code_t](#) rs_hook
- void * arg
- [bgrt_stack_t](#) * sstart
- volatile [bgrt_stack_t](#) * spointer
- [BGRT_USPD_PROC_T](#) udata

4.9.1 Подробное описание

Процесс.

В разных ОС это называется по-разному: процесс, поток, задача и пр., суть такова: это независимый поток исполнения инструкций процессора.

То есть это исполняющийся кусок твоей программы, у которого есть своя собственная «main» (смотри поле `rmain`), и эта «main» может быть написана так, как будто других процессов нет!

Можно использовать 1 функцию `rmain` для нескольких процессов, каждый запущенный экземпляр `rmain` не зависит от других, но есть одно но.

Предупреждения

Осторожно со статическими переменными, они будут общими для всех запущенных экземпляров, доступ к ним необходимо организовывать только с помощью средств синхронизации процессов.

4.9.2 Поля

4.9.2.1 `affinity` `bgrt_aff_t` `bgrt_priv_proc_t::affinity`

Аффинность (индекс процессоров, на котором может исполняться процесс).

4.9.2.2 `arg` `void*` `bgrt_priv_proc_t::arg`

Аргумент для `rmain`, `sv_hook`, `rs_hook`, может хранить ссылку на локальные данные конкретного экземпляра процесса.

4.9.2.3 `base_prio` `bgrt_prio_t` `bgrt_priv_proc_t::base_prio`

Базовый приоритет.

4.9.2.4 `cnt_lock` `bgrt_cnt_t` `bgrt_priv_proc_t::cnt_lock`

Счётчик уровней вложенности `BGRT_PROC_LOCK`.

4.9.2.5 `core_id` `bgrt_cpuid_t` `bgrt_priv_proc_t::core_id`

Идентификатор процессора, на котором исполняется процесс.

4.9.2.6 `flags` `bgrt_flag_t` `bgrt_priv_proc_t::flags`

Флаги (для ускорения анализа состояния процесса).

4.9.2.7 `lock` `bgrt_lock_t` `bgrt_priv_proc_t::lock`

Спин блокировка процесса.

4.9.2.8 `lres` `bgrt_pcounter_t` `bgrt_priv_proc_t::lres`

Счётчик захваченных ресурсов.

4.9.2.9 `parent` `bgrt_pitem_t` `bgrt_priv_proc_t::parent`

Родитель - `bgrt_pitem_t`.

4.9.2.10 `pmain` `bgrt_code_t` `bgrt_priv_proc_t::pmain`

Главная функция процесса.

4.9.2.11 `rs_hook` `bgrt_code_t` `bgrt_priv_proc_t::rs_hook`

Хук, исполняется планировщиком перед восстановлением контекста процесса.

4.9.2.12 `spointer` `volatile bgrt_stack_t*` `bgrt_priv_proc_t::spointer`

Указатель на вершину стека экземпляра процесса.

4.9.2.13 `sstart` `bgrt_stack_t*` `bgrt_priv_proc_t::sstart`

Указатель на дно стека экземпляра процесса.

4.9.2.14 `sv_hook` `bgrt_code_t` `bgrt_priv_proc_t::sv_hook`

Хук, исполняется планировщиком после сохранения контекста процесса.

4.9.2.15 `sync` `struct bgrt_priv_sync_t*` `bgrt_priv_proc_t::sync`

4.9.2.16 `time_quant` `bgrt_tmr_t` `bgrt_priv_proc_t::time_quant`

Квант времени процесса.

4.9.2.17 `timer` `bgrt_tmr_t` `bgrt_priv_proc_t::timer`

Таймер процесса, для процессов жесткого реального времени используется как `watchdog`.

4.9.2.18 `udata` [BGRT_USPD_PROC_T](#) `bgrt_priv_proc_t::udata`

Данные процесса из пространства пользователя.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/proc.h](#)

4.10 Структура `bgrt_priv_sched_t`

Планировщик.

```
#include "bugurtos/kernel/sched.h"
```

Поля данных

- `bgrt_proc_t * current_proc`
- `bgrt_xlist_t * ready`
- `bgrt_xlist_t * expired`
- `bgrt_xlist_t plst [2]`
- `bgrt_cnt_t nested_crit_sec`
- `bgrt_lock_t lock`

4.10.1 Подробное описание

Планировщик.

Планировщик содержит информацию о процессах, запущенных на процессоре (процессорном ядре).

4.10.2 Поля

4.10.2.1 `current_proc` [bgrt_proc_t*](#) `bgrt_priv_sched_t::current_proc`

Текущий процесс.

4.10.2.2 `expired` [bgrt_xlist_t*](#) `bgrt_priv_sched_t::expired`

Указатель на список процессов, исчерпавших свой квант времени.

4.10.2.3 `lock` [bgrt_lock_t](#) `bgrt_priv_sched_t::lock`

Спин-блокировка планировщика.

4.10.2.4 `nested_crit_sec` `bgrt_cnt_t` `bgrt_priv_sched_t::nested_crit_sec`

Счётчик вложенности критических секций.

4.10.2.5 `plst` `bgrt_xlist_t` `bgrt_priv_sched_t::plst[2]`

Сами списки процессов.

4.10.2.6 `ready` `bgrt_xlist_t*` `bgrt_priv_sched_t::ready`

Указатель на список готовых к выполнению процессов.

Объявления и описания членов структуры находятся в файле:

- `bugurtos/kernel/sched.h`

4.11 Структура `bgrt_priv_sync_t`

Базовый примитив синхронизации.

```
#include "bugurtos/kernel/sync.h"
```

Поля данных

- `bgrt_xlist_t` `sleep`
- `bgrt_proc_t` * `owner`
- `bgrt_cnt_t` `dirty`
- `bgrt_cnt_t` `snum`
- `bgrt_cnt_t` `pwake`
- `bgrt_prio_t` `prio`
- `bgrt_lock_t` `lock`

4.11.1 Подробное описание

Базовый примитив синхронизации.

Базовый тип, отвечающий за блокирующую синхронизацию процессов. Путём "обёртывания" данного типа можно получить привычные примитивы синхронизации (мьютексы, семафоры, условные переменные, FIFO-буферы, блокирующий IPC, и т.д.).

Поддерживает протокол наследования приоритетов (Basic Priority Inheritance).

4.11.2 Поля

4.11.2.1 `dirty` `bgrt_cnt_t` `bgrt_priv_sync_t::dirty`

Счётчик незавершённых транзакций наследования приоритетов.

4.11.2.2 `lock` `bgrt_lock_t` `bgrt_priv_sync_t::lock`

Спин-блокировка.

4.11.2.3 `owner` `bgrt_proc_t*` `bgrt_priv_sync_t::owner`

Указатель на процесс-хозяин.

4.11.2.4 `prio` `bgrt_prio_t` `bgrt_priv_sync_t::prio`

Приоритет.

4.11.2.5 `pwake` `bgrt_cnt_t` `bgrt_priv_sync_t::pwake`

Счётчик отложенных пробуждений.

4.11.2.6 `sleep` `bgrt_xlist_t` `bgrt_priv_sync_t::sleep`

Список ожидающих процессов.

4.11.2.7 `snum` `bgrt_cnt_t` `bgrt_priv_sync_t::snum`

Счётчик спящих процессов.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/sync.h](#)

4.12 Структура `bgrt_priv_uspd_t`

```
#include "bugurtos/kernel/proc.h"
```

Поля данных

- `void *` `scarg`
- `bgrt_syscall_t` `snum`
- `bgrt_st_t` `secret`

4.12.1 Поля

4.12.1.1 `scarg` `void* bgrt_priv_uspd_t::scarg`

Указатель на аргумент системного вызова.

4.12.1.2 `snum` `bgrt_syscall_t bgrt_priv_uspd_t::snum`

Номер системного вызова.

4.12.1.3 `cret` `bgrt_st_t bgrt_priv_uspd_t::cret`

Результат системного вызова.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/proc.h](#)

4.13 Структура `bgrt_priv_vic_t`

Виртуальный контроллер прерываний.

```
#include "bugurtos/kernel/vint.h"
```

Поля данных

- [bgrt_xlist_t list](#)
- [bgrt_prio_t prio](#)

4.13.1 Подробное описание

Виртуальный контроллер прерываний.

4.13.2 Поля

4.13.2.1 `list` `bgrt_xlist_t bgrt_priv_vic_t::list`

Родитель - `bgrt_xlist_t`.

4.13.2.2 `prio` `bgrt_prio_t bgrt_priv_vic_t::prio`

Текущий приоритет.

Объявления и описания членов структуры находятся в файле:

- [bugurtos/kernel/vint.h](#)

4.14 Структура `bgrt_priv_vint_t`

Виртуальное прерывание.

```
#include "bugurtos/kernel/vint.h"
```

Поля данных

- `bgrt_pitem_t` `parent`
- `bgrt_code_t` `func`
- `void *` `arg`

4.14.1 Подробное описание

Виртуальное прерывание.

4.14.2 Поля

4.14.2.1 `arg` `void*` `bgrt_priv_vint_t::arg`

Аргумент обработчика.

4.14.2.2 `func` `bgrt_code_t` `bgrt_priv_vint_t::func`

Указатель на обработчик.

4.14.2.3 `parent` `bgrt_pitem_t` `bgrt_priv_vint_t::parent`

Родитель - `bgrt_item_t`.

Объявления и описания членов структуры находятся в файле:

- `bugurtos/kernel/vint.h`

4.15 Структура `bgrt_priv_xlist_t`

Список с приоритетами.

```
#include "bugurtos/kernel/xlist.h"
```

Поля данных

- `bgrt_item_t *` `item` [`BGRT_BITS_IN_INDEX_T`]
- `bgrt_map_t` `map`

4.15.1 Подробное описание

Список с приоритетами.

Такой список хранит ссылки на структуры типа `bgrt_item_t`. Фактически в нем будут храниться ссылки на элементы типа `bgrt_pitem_t`.

4.15.2 Поля

4.15.2.1 `item` `bgrt_item_t*` `bgrt_priv_xlist_t::item[BGRT_BITS_IN_INDEX_T]`

Массив указателей на элементы.

4.15.2.2 `map` `bgrt_map_t` `bgrt_priv_xlist_t::map`

Индекс, показывает, где в массиве ненулевые указатели.

Объявления и описания членов структуры находятся в файле:

- `bugurtos/kernel/xlist.h`

5 Файлы

5.1 Файл `bugurtos/arch/common/atm_cortex_m34_1.h`

Макросы

- `#define BGRT_ATM_INIT_ISR(map_ptr) do{*(map_ptr) = (bgrt_map_t)0;}while(0)`
- `#define BGRT_ATM_BSET_ISR(map_ptr, msk) __atomic_fetch_or((map_ptr), (msk), _↔
_ATOMIC_SEQ_CST)`
- `#define BGRT_ATM_BGET_ISR(map_ptr, msk) (*(map_ptr) & (msk))`
- `#define BGRT_ATM_BCLR_ISR(map_ptr, msk) ((msk) & __atomic_fetch_and((map_ptr),
~(msk), _ATOMIC_SEQ_CST))`
- `#define BGRT_VINT_PUSH_ISR bgrt_vint_push`

5.1.1 Макросы

5.1.1.1 `BGRT_ATM_BCLR_ISR` `#define BGRT_ATM_BCLR_ISR(map_ptr, msk) ((msk) & __atomic_fetch_and((map_ptr), ~(msk), _ATOMIC_SEQ_CST))`

```
5.1.1.2 BGRT_ATM_BGET_ISR #define BGRT_ATM_BGET_ISR(  
    map_ptr,  
    msk ) (*(map_ptr) & (msk))
```

```
5.1.1.3 BGRT_ATM_BSET_ISR #define BGRT_ATM_BSET_ISR(  
    map_ptr,  
    msk ) __atomic_fetch_or((map_ptr), (msk), __ATOMIC_SEQ_CST)
```

```
5.1.1.4 BGRT_ATM_INIT_ISR #define BGRT_ATM_INIT_ISR(  
    map_ptr ) do{*(map_ptr) = (bgrt_map_t)0;}while(0)
```

```
5.1.1.5 BGRT_VINT_PUSH_ISR #define BGRT_VINT_PUSH_ISR bgrt_vint_push
```

5.2 Файл bugurtos/arch/common/atm_gen_1.h

Макросы

- #define [BGRT_ATM_INIT_ISR](#)(map_ptr) do{*(map_ptr) = (bgrt_map_t)0;}while(0)
- #define [BGRT_ATM_BSET_ISR](#)(map_ptr, msk) do{ *(map_ptr) |= (msk); }while(0)
- #define [BGRT_ATM_BGET_ISR](#)(map_ptr, msk) (*(map_ptr) & (msk))
- #define [BGRT_ATM_BCLR_ISR](#)(map_ptr, msk) (__bgrt_atm_bclr_isr((map_ptr), (msk)))
- #define [BGRT_VINT_PUSH_ISR](#) bgrt_vint_push_isr

5.2.1 Макросы

```
5.2.1.1 BGRT_ATM_BCLR_ISR #define BGRT_ATM_BCLR_ISR(  
    map_ptr,  
    msk ) (__bgrt_atm_bclr_isr((map_ptr), (msk)))
```

```
5.2.1.2 BGRT_ATM_BGET_ISR #define BGRT_ATM_BGET_ISR(  
    map_ptr,  
    msk ) (*(map_ptr) & (msk))
```

```
5.2.1.3 BGRT_ATM_BSET_ISR #define BGRT_ATM_BSET_ISR(  
    map_ptr,  
    msk ) do{ *(map_ptr) |= (msk); }while(0)
```

5.2.1.4 `BGRT_ATM_INIT_ISR` `#define BGRT_ATM_INIT_ISR(map_ptr) do{*(map_ptr) = (bgrt_map_t)0;}while(0)`

5.2.1.5 `BGRT_VINT_PUSH_ISR` `#define BGRT_VINT_PUSH_ISR bgrt_vint_push_isr`

5.3 Файл `bugurtos/doc/doxygen/bugurt_port.h`

Макросы

- `#define BGRT_INT_LOCK()`
Запретить прерывания.
- `#define BGRT_INT_FREE()`
Разрешить прерывания.
- `#define BGRT_KBLOCK`
Текущий блок ядра.
- `#define BGRT_CURR_PROC`
Текущий процесс.
- `#define BGRT_ISR(v)`
Шаблон обработчика прерывания.
- `#define BGRT_ATM_INIT_ISR(map_ptr)`
Инициализация атомарной карты.
- `#define BGRT_ATM_BSET_ISR(map_ptr, msk)`
Поставить биты в 1 по маске.
- `#define BGRT_ATM_BGET_ISR(map_ptr, msk)`
Считать биты по маске.
- `#define BGRT_ATM_BCLR_ISR(map_ptr, msk)`
Сбросить значения битов по маске.

Функции

- `void bgrt_atm_init (bgrt_map_t *map_ptr)`
Инициализация атомарной карты.
- `void bgrt_atm_bset (bgrt_map_t *map_ptr, bgrt_map_t msk)`
Поставить биты в 1 по маске.
- `bgrt_map_t bgrt_atm_bget (bgrt_map_t *map_ptr, bgrt_map_t msk)`
Считать биты по маске.
- `bgrt_map_t bgrt_atm_bclr (bgrt_map_t *map_ptr, bgrt_map_t msk)`
Сбросить биты по маске.

5.3.1 Макросы

5.3.1.1 `BGRT_ATM_BCLR_ISR` `#define BGRT_ATM_BCLR_ISR(map_ptr, msk)`

Сбросить значения битов по маске.

Предупреждения

Для вызова из обработчиков прерываний/критических секций!

Аргументы

map_ptr	Указатель на атомарную карту.
msk	Маска.

Возвращает

Последнее состояние интересующих битов.

```
5.3.1.2 BGRT_ATM_BGET_ISR #define BGRT_ATM_BGET_ISR(  
    map_ptr,  
    msk )
```

Считать биты по маске.

Предупреждения

Для вызова из обработчиков прерываний/критических секций!

Аргументы

map_ptr	Указатель на атомарную карту.
msk	Маска.

Возвращает

Состояние векторов прерываний.

```
5.3.1.3 BGRT_ATM_BSET_ISR #define BGRT_ATM_BSET_ISR(  
    map_ptr,  
    msk )
```

Поставить биты в 1 по маске.

Аргументы

map_ptr	Указатель на атомарную карту.
msk	Маска.

```
5.3.1.4 BGRT_ATM_INIT_ISR #define BGRT_ATM_INIT_ISR(  
    map_ptr )
```

Инициализация атомарной карты.

Предупреждения

Для вызова из обработчиков прерываний/критических секций!

Аргументы

map_ptr	Указатель на атомарную карту.
---------	-------------------------------

5.3.1.5 BGRT_CURR_PROC #define BGRT_CURR_PROC

Текущий процесс.

5.3.1.6 BGRT_INT_FREE #define BGRT_INT_FREE()

Разрешить прерывания.

5.3.1.7 BGRT_INT_LOCK #define BGRT_INT_LOCK()

Запретить прерывания.

5.3.1.8 BGRT_ISR #define BGRT_ISR(
v)

Шаблон обработчика прерывания.

Аргументы

v	Идентификатор обработчика прерывания.
---	---------------------------------------

5.3.1.9 BGRT_KBLOCK #define BGRT_KBLOCK

Текущий блок ядра.

5.3.2 Функции

5.3.2.1 `bgrt_atm_bclr()` `bgrt_map_t bgrt_atm_bclr (`
`bgrt_map_t * map_ptr,`
`bgrt_map_t msk)`

Сбросить биты по маске.

Аргументы

<code>map_ptr</code>	Указатель на атомарную карту.
<code>msk</code>	Маска.

Возвращает

Последнее состояние маскированных битов.

5.3.2.2 `bgrt_atm_bget()` `bgrt_map_t bgrt_atm_bget (`
`bgrt_map_t * map_ptr,`
`bgrt_map_t msk)`

Считать биты по маске.

Аргументы

<code>map_ptr</code>	Указатель на атомарную карту.
<code>msk</code>	Маска.

Возвращает

Состояние векторов прерываний.

5.3.2.3 `bgrt_atm_bset()` `void bgrt_atm_bset (`
`bgrt_map_t * map_ptr,`
`bgrt_map_t msk)`

Поставить биты в 1 по маске.

Предупреждения

Для вызова из обработчиков прерываний/критических секций!

Аргументы

<code>map_ptr</code>	Указатель на атомарную карту.
<code>msk</code>	Маска.

5.3.2.4 `bgrt_atm_init()` `void bgrt_atm_init (`
`bgrt_map_t * map_ptr)`

Инициализация атомарной карты.

Аргументы

<code>map_ptr</code>	Указатель на атомарную карту.
----------------------	-------------------------------

5.4 Файл `bugurtos/kernel/bugurt.h`

Главный заголовочный файл.

```
#include <bugurt_config.h>
#include "index.h"
#include "item.h"
#include "pcounter.h"
#include "xlist.h"
#include "pitem.h"
#include "crit_sec.h"
#include "proc.h"
#include "sched.h"
#include "sync.h"
#include "syscall.h"
#include "timer.h"
#include <bugurt_port.h>
#include "vint.h"
#include "kernel.h"
```

Макросы

- `#define BGRT_CDECL_BEGIN`
- `#define BGRT_CDECL_END`
- `#define BGRT_CONCAT(a, b) a##b`
- `#define BGRT_CONCAT2(a, b) BGRT_CONCAT(a,b)`
- `#define BGRT_CONCAT3(a, b) BGRT_CONCAT2(a,b)`
- `#define BGRT_ASSERT(c, msg) do{}while(0)`
- `#define BGRT_ST_OK ((bgrt_st_t)0)`
Удачное завершение.
- `#define BGRT_ST_ENULL ((bgrt_st_t)1)`
Передан нулевой указатель.
- `#define BGRT_ST_EOWN ((bgrt_st_t)2)`
Ошибка владения.
- `#define BGRT_ST_EEMPTY ((bgrt_st_t)3)`
Список процессов пуст.
- `#define BGRT_ST_ESYNC ((bgrt_st_t)4)`
Не тот объект типа `bgrt_sync_t`.
- `#define BGRT_ST_ETIMEOUT ((bgrt_st_t)5)`

- Истек таймаут `bgrt_sync_t`.
- `#define BGRT_ST_ESTAT ((bgrt_st_t)6)`
Ошибка состояния процесса.
- `#define BGRT_ST_EAGAIN ((bgrt_st_t)7)`
Попробуйте ещё раз.
- `#define BGRT_ST_SCALL ((bgrt_st_t)8)`
Неправильный системный вызов.
- `#define BGRT_ST_ROLL ((bgrt_st_t)9)`
Нужна следующая итерация.
- `#define BGRT_ST_IDLE ((bgrt_st_t)10)`
Простой системы.
- `#define BGRT_PRIO_LOWEST ((bgrt_prio_t)BGRT_BITS_IN_INDEX_T - (bgrt_prio_t)1)`
Низший приоритет.
- `#define BGRT_SPIN_INIT(arg) bgrt_spin_init(&((arg)->lock))`
Макрос-обёртка.
- `#define BGRT_SPIN_LOCK(arg) bgrt_spin_lock(&((arg)->lock))`
Макрос-обёртка.
- `#define BGRT_SPIN_FREE(arg) bgrt_spin_free(&((arg)->lock))`
Макрос-обёртка.
- `#define BGRT_RESCHEDED_PROC(proc) bgrt_resched(proc->core_id)`
Макрос-обёртка.
- `#define BGRT_KERNEL_PREEMPT() bgrt_kblock_do_work(&bgrt_kernel.kblock[bgrt_curr_cpu()])`

Определения типов

- `typedef void(* bgrt_code_t) (void *)`
Исполняемый код.

Функции

- `void bgrt_spin_init (bgrt_lock_t *lock)`
Инициализация спин-блокировки на многопроцессорной системе.
- `void bgrt_spin_lock (bgrt_lock_t *lock)`
Захват спин-блокировки на многопроцессорной системе.
- `void bgrt_spin_free (bgrt_lock_t *lock)`
Освобождение спин-блокировки на многопроцессорной системе.
- `bgrt_cpuid_t bgrt_curr_cpu (void)`
Возвращает id процессорного ядра.
- `void bgrt_stat_init (bgrt_ls_t *stat)`
Инициализация статистики
- `void bgrt_stat_inc (bgrt_proc_t *proc, bgrt_ls_t *stat)`
Обновление статистики при запуске процесса/ вставки процесса в очередь сигнала.
- `void bgrt_stat_dec (bgrt_proc_t *proc, bgrt_ls_t *stat)`
Обновление статистики при останове процесса/ удаления процесса из очереди сигнала.
- `void bgrt_stat_merge (bgrt_ls_t *src_stat, bgrt_ls_t *dst_stat)`
Перенос статистики.
- `bgrt_load_t bgrt_stat_calc_load (bgrt_prio_t prio, bgrt_ls_t *stat)`
Расчёт нагрузки.
- `void bgrt_resched (bgrt_cpuid_t core_id)`

- `bgrt_proc_t * bgrt_curr_proc (void)`
Текущий процесс.
- `bgrt_stack_t * bgrt_proc_stack_init (bgrt_stack_t *sstart, bgrt_code_t pmain, void *arg, void(*return_address)(void))`
Инициализация стека процесса.
- `void bgrt_init (void)`
Инициализация Ядра.
- `void bgrt_start (void)`
Запуск Ядра.
- `bgrt_st_t bgrt_syscall (bgrt_syscall_t num, void *arg)`
Системный вызов.
- `void bgrt_switch_to_proc (void)`
Переключение из контекста Ядра в контекст текущего процесса.

5.4.1 Подробное описание

Главный заголовочный файл.

В этот файл включены все заголовочные файлы BuguRTOS. В свою очередь все исходные тексты включают этот файл.

5.4.2 Макросы

5.4.2.1 `BGRT_ASSERT` `#define BGRT_ASSERT(`
 `c,`
 `msg) do{}while(0)`

5.4.2.2 `BGRT_CDECL_BEGIN` `#define BGRT_CDECL_BEGIN`

5.4.2.3 `BGRT_CDECL_END` `#define BGRT_CDECL_END`

5.4.2.4 `BGRT_CONCAT` `#define BGRT_CONCAT(`
 `a,`
 `b) a##b`

5.4.2.5 `BGRT_CONCAT2` `#define BGRT_CONCAT2(`
 `a,`
 `b) BGRT_CONCAT(a,b)`

5.4.2.6 `BGRT_CONCAT3` `#define BGRT_CONCAT3(`
 `a,`
 `b) BGRT_CONCAT2(a,b)`

5.4.2.7 `BGRT_KERNEL_PREEMPT` `#define BGRT_KERNEL_PREEMPT() bgrt_kblock_do_work(&bgrt↔`
 `_kernel.kblock[bgrt_curr_cpu()])`

5.4.2.8 `BGRT_PRIO_LOWEST` `#define BGRT_PRIO_LOWEST ((bgrt_prio_t)BGRT_BITS_IN_INDE↔`
 `X_T - (bgrt_prio_t)1)`

Низший приоритет.

5.4.2.9 `BGRT_RESCHEDED_PROC` `#define BGRT_RESCHEDED_PROC(`
 `proc) bgrt_resched(proc->core_id)`

Макрос-обёртка.

Обёртка функции `bgrt_resched`.

5.4.2.10 `BGRT_SPIN_FREE` `#define BGRT_SPIN_FREE(`
 `arg) bgrt_spin_free(&((arg)->lock))`

Макрос-обёртка.

Обёртка освобождения спин-блокировки `arg->lock`, на однопроцессорной системе - пустой макрос.

5.4.2.11 `BGRT_SPIN_INIT` `#define BGRT_SPIN_INIT(`
 `arg) bgrt_spin_init(&((arg)->lock))`

Макрос-обёртка.

Обёртка инициализации спин-блокировки `arg->lock`, на однопроцессорной системе - пустой макрос.

5.4.2.12 `BGRT_SPIN_LOCK` `#define BGRT_SPIN_LOCK(`
 `arg) bgrt_spin_lock(&((arg)->lock))`

Макрос-обёртка.

Обёртка захвата спин-блокировки `arg->lock`, на однопроцессорной системе - пустой макрос.

5.4.2.13 `BGRТ_ST_EAGAIN` `#define BGRТ_ST_EAGAIN ((bgrт_st_t)7)`

Попробуйте ещё раз.

5.4.2.14 `BGRТ_ST_EEMPTY` `#define BGRТ_ST_EEMPTY ((bgrт_st_t)3)`

Список процессов пуст.

5.4.2.15 `BGRТ_ST_ENULL` `#define BGRТ_ST_ENULL ((bgrт_st_t)1)`

Передан нулевой указатель.

5.4.2.16 `BGRТ_ST_EOWN` `#define BGRТ_ST_EOWN ((bgrт_st_t)2)`

Ошибка владения.

5.4.2.17 `BGRТ_ST_ESTAT` `#define BGRТ_ST_ESTAT ((bgrт_st_t)6)`

Ошибка состояния процесса.

5.4.2.18 `BGRТ_ST_ESYNC` `#define BGRТ_ST_ESYNC ((bgrт_st_t)4)`

Не тот объект типа `bgrт_sync_t`.

5.4.2.19 `BGRТ_ST_ETIMEOUT` `#define BGRТ_ST_ETIMEOUT ((bgrт_st_t)5)`

Истек таймаут `bgrт_sync_t`.

5.4.2.20 `BGRТ_ST_IDLE` `#define BGRТ_ST_IDLE ((bgrт_st_t)10)`

Простой системы.

5.4.2.21 `BGRT_ST_OK` `#define BGRT_ST_OK ((bgrt_st_t)0)`

Удачное завершение.

5.4.2.22 `BGRT_ST_ROLL` `#define BGRT_ST_ROLL ((bgrt_st_t)9)`

Нужна следующая итерация.

5.4.2.23 `BGRT_ST_SCALL` `#define BGRT_ST_SCALL ((bgrt_st_t)8)`

Неправильный системный вызов.

5.4.3 Типы

5.4.3.1 `bgrt_code_t` `typedef void(* bgrt_code_t) (void *)`

Исполняемый код.

Указатель на функцию типа `void`, принимающую в качестве аргумента указатель типа `void`.

5.4.4 Функции

5.4.4.1 `bgrt_curr_cpu()` `bgrt_cpuid_t bgrt_curr_cpu (`
`void)`

Возвращает id процессорного ядра.

Возвращает id процессорного ядра, на котором выполняется.

Предупреждения

Для внутреннего использования.

5.4.4.2 `bgrt_curr_proc()` `bgrt_proc_t*` `bgrt_curr_proc (`
`void)`

Текущий процесс.

Предупреждения

Для внутреннего использования.

Возвращает

Указатель на текущий процесс, исполняемый на локальном процессоре.

5.4.4.3 `bgrt_init()` `void` `bgrt_init (`
`void)`

Инициализация Ядра.

Подготовка Ядра к запуску.

5.4.4.4 `bgrt_proc_stack_init()` `bgrt_stack_t*` `bgrt_proc_stack_init (`
`bgrt_stack_t * sstart,`
`bgrt_code_t pmain,`
`void * arg,`
`void(*) (void) return_address)`

Инициализация стека процесса.

Подготовка стека к запуску процесса. Делает так, что после восстановления контекста процесса происходит вызов функции `pmain(arg)`.

Предупреждения

Для внутреннего использования.

Аргументы

<code>sstart</code>	Дно стека.
<code>pmain</code>	Функция, которая будет вызвана после восстановления контекста.
<code>arg</code>	Аргумент вызываемой функции.
<code>return_address</code>	адрес возврата из <code>pmain</code> .

Возвращает

Указатель на вершину подготовленного стека.

5.4.4.5 `bgrt_resched()` `void bgrt_resched (`
`bgrt_cpuid_t core_id)`

Перепланировка.

Запускает перепланировку на одном из процессорных ядер.

Предупреждения

Для внутреннего использования.

Аргументы

<code>core↔</code> <code>_id</code>	ID ядра, на котором надо перепланировать исполнение процессов.
--	--

5.4.4.6 `bgrt_spin_free()` `void bgrt_spin_free (`
`bgrt_lock_t * lock)`

Освобождение спин-блокировки на многопроцессорной системе.

Аргументы

<code>lock</code>	Указатель на спин-блокировку.
-------------------	-------------------------------

Предупреждения

Для внутреннего использования.

5.4.4.7 `bgrt_spin_init()` `void bgrt_spin_init (`
`bgrt_lock_t * lock)`

Инициализация спин-блокировки на многопроцессорной системе.

Инициализация спин-блокировки на многопроцессорной системе.

Предупреждения

Для внутреннего использования.

Аргументы

<code>lock</code>	Указатель на спин-блокировку.
-------------------	-------------------------------

5.4.4.8 `bgrt_spin_lock()` `void bgrt_spin_lock (`
`bgrt_lock_t * lock)`

Захват спин-блокировки на многопроцессорной системе.

Захват спин-блокировки на многопроцессорной системе.

Предупреждения

Для внутреннего использования.

Аргументы

lock	Указатель на спин-блокировку.
------	-------------------------------

5.4.4.9 `bgrt_start()` `void bgrt_start (`
`void)`

Запуск Ядра.

Запуск Ядра. После вызова этой функции можно ничего не писать - все равно исполняться не будет.

5.4.4.10 `bgrt_stat_calc_load()` `bgrt_load_t bgrt_stat_calc_load (`
`bgrt_prio_t prio,`
`bgrt_ls_t * stat)`

Расчёт нагрузки.

Расчёт нагрузки на одном процессорном ядре.

Предупреждения

Для внутреннего использования.

Аргументы

prio	Приоритет процесса, для которого считаем нагрузку.
stat	Указатель на структуру статистики Ядра.

Возвращает

Текущую оценку нагрузки на процессорном ядре, за которое отвечает stat.

5.4.4.11 `bgrt_stat_dec()` `void bgrt_stat_dec (`
 `bgrt_proc_t * proc,`
 `bgrt_ls_t * stat)`

Обновление статистики при останове процесса/ удаления процесса из очереди сигнала.

Обновление статистики при останове процесса/ удаления процесса из очереди сигнала.

Предупреждения

Для внутреннего использования.

Аргументы

<code>proc</code>	Указатель на процесс.
<code>stat</code>	Указатель на структуру статистики.

5.4.4.12 `bgrt_stat_inc()` `void bgrt_stat_inc (`
 `bgrt_proc_t * proc,`
 `bgrt_ls_t * stat)`

Обновление статистики при запуске процесса/ вставки процесса в очередь сигнала.

Обновление статистики при запуске процесса/ вставки процесса в очередь сигнала.

Предупреждения

Для внутреннего использования.

Аргументы

<code>proc</code>	Указатель на процесс.
<code>stat</code>	Указатель на структуру статистики.

5.4.4.13 `bgrt_stat_init()` `void bgrt_stat_init (`
 `bgrt_ls_t * stat)`

Инициализация статистики

Инициализирует структуру `bgrt_ls_t`, в которой хранится статистика.

Предупреждения

Для внутреннего использования.

Аргументы

stat	Указатель на структуру статистики.
------	------------------------------------

```
5.4.4.14 bgrt_stat_merge() void bgrt_stat_merge (
    bgrt_ls_t * src_stat,
    bgrt_ls_t * dst_stat )
```

Перенос статистики.

При передаче процессов из списка ожидающих сигнал в список готовых к выполнению надо обновить статистическую информацию в Ядре.

Предупреждения

Для внутреннего использования.

Аргументы

src_stat	Указатель на структуру статистики сигнала.
dst_stat	Указатель на структуру статистики Ядра.

```
5.4.4.15 bgrt_switch_to_proc() void bgrt_switch_to_proc (
    void )
```

Переключение из контекста Ядра в контекст текущего процесса.

```
5.4.4.16 bgrt_syscall() bgrt_st_t bgrt_syscall (
    bgrt_syscall_t num,
    void * arg )
```

Системный вызов.

Код Ядра всегда выполняется в контексте Ядра. Это нужно для экономии памяти в стеках процессов. Соответственно, если мы хотим выполнить какие либо операции над процессами, мьютексами, семафорами, сигналами, то нам нужно "попросить" Ядро сделать эту работу.

Именно для этого существует функция `bgrt_syscall`, которая передаёт управление Ядру для выполнения требуемой работы.

Предупреждения

Для внутреннего использования.

Аргументы

num	номер системного вызова (что именно надо выполнить).
arg	аргумент системного вызова (над чем это надо выполнить).

5.5 Файл bugurtos/kernel/crit_sec.h

Заголовок критических секций.

Макросы

- `#define BGRT_CRIT_SEC_ENTER()`
Макрос-обёртка.
- `#define BGRT_CRIT_SEC_EXIT() bgrt_priv_crit_sec_exit(current_core)`
Макрос-обёртка.

Функции

- `bgrt_cpuid_t bgrt_priv_crit_sec_enter (void)`
Вход в критическую секцию на многопроцессорной системе.
- `void bgrt_priv_crit_sec_exit (bgrt_cpuid_t core)`
Выход из критической секции на многопроцессорной системе

5.5.1 Подробное описание

Заголовок критических секций.

Критическая секция - область кода, в которой запрещены все прерывания. Критические секции используются, когда надо использовать общий ресурс в течение короткого времени.

Критические секции могут быть вложенные, в этом случае прерывания разрешаются, когда произошёл выход из всех критических секций.

5.5.2 Макросы

5.5.2.1 BGRT_CRIT_SEC_ENTER `#define BGRT_CRIT_SEC_ENTER()`

Макроопределение:

```
bgrt_cpuid_t current_core; \
current_core = bgrt_priv_crit_sec_enter()
```

Макрос-обёртка.

Вход в критическую секцию.

Предупреждения

Использовать в начале блока!

Все локальные переменные должны быть объявлены до BGRT_CRIT_SEC_ENTER

5.5.2.2 `BGRT_CRIT_SEC_EXIT` `#define BGRT_CRIT_SEC_EXIT() bgrt_priv_crit_sec_exit(current_↔ core)`

Макрос-обёртка.

Выход из критической секции.

Предупреждения

Использовать в конце блока!

5.5.3 Функции

5.5.3.1 `bgrt_priv_crit_sec_enter()` `bgrt_cpuid_t bgrt_priv_crit_sec_enter (void)`

Вход в критическую секцию на многопроцессорной системе.

Возвращает

ID процессорного ядра, на котором выполняется.

5.5.3.2 `bgrt_priv_crit_sec_exit()` `void bgrt_priv_crit_sec_exit (bgrt_cpuid_t core)`

Выход из критической секции на многопроцессорной системе

Аргументы

core	ID процессорного ядра, на котором выполняется.
------	--

Предупреждения

Передача ID процессорного ядра, отличного от того, где запускается функция приведёт к непредсказуемым последствиям.

5.6 Файл `bugurtos/kernel/default/syscall_api.h`

Заголовок системных вызовов.

Макросы

- `#define BGRT_PROC_RUN(pid) BGRT_SYSCALL_N(PROC_RUN, (void *) (pid))`

- Запуск процесса.

 - #define `BGRT_PROC_RESTART(pid) BGRT_SYSCALL_N(PROC_RESTART, (void *) (pid))`
- Перезапуск процесса.

 - #define `BGRT_PROC_STOP(pid) BGRT_SYSCALL_N(PROC_STOP, (void *) (pid))`
- Останов процесса.

 - #define `BGRT_PROC_SELF_STOP() BGRT_SYSCALL_N(PROC_SELF_STOP, (void *) 0)`
- Самоостанов процесса.

 - #define `BGRT_PROC_LOCK() BGRT_SYSCALL_N(PROC_LOCK, (void *) 0)`
- Установка флага `BGRT_PROC_FLG_LOCK` для вызывающего процесса.

 - #define `BGRT_PROC_FREE() BGRT_SYSCALL_N(PROC_FREE, (void *) 0)`
- Останов процесса по флагу `BGRT_PROC_FLG_PRE_STOP`.

 - #define `BGRT_PROC_RESET_WATCHDOG() BGRT_SYSCALL_N(PROC_RESET_WATCHDOG, (void *) 0)`
- Сброс watchdog для процесса реального времени.

 - #define `BGRT_PROC_GET_PRIO(pid, pri_ptr) (*(pri_ptr) = BGRT_PRIO_LOWEST+1, BGRT_SYSCALL_NVAR(PROC_GET_PRIO, (void *) (pri_ptr), (void *) (pid)))`
- Получить приоритет процесса.

 - #define `BGRT_PROC_SET_PRIO(pid, prio) BGRT_SYSCALL_NVAR(PROC_SET_PRIO, (void *) (pid), (int) (prio))`
- Управление приоритетом процесса.

 - #define `BGRT_PROC_GET_ID(pid_ptr) (*(pid_ptr) = BGRT_PID_NOTHING, BGRT_SYSCALL_N(PROC_GET_ID, (void *) (pid_ptr)))`
- Получить идентификатор текущего процесса.

 - #define `BGRT_SYNC_SET_OWNER(sync_ptr, pid) BGRT_SYSCALL_NVAR(SYNC_SET_OWNER, (void *) (sync_ptr), (void *) (pid))`
- Назначить хозяина объекта типа `bgrt_sync_t`.

 - #define `BGRT_SYNC_GET_OWNER(sync_ptr, pid_ptr) (*(pid_ptr) = BGRT_PID_NOTHING, BGRT_SYSCALL_NVAR(SYNC_GET_OWNER, (void *) (pid_ptr), (void *) (sync_ptr)))`
- Получить хозяина примитива.

 - #define `BGRT_SYNC_OWN(sync_ptr, touch) BGRT_SYSCALL_NVAR(SYNC_OWN, (void *) (sync_ptr), (int) (touch))`
- Завладеть объектом типа `bgrt_sync_t`.

 - #define `BGRT_SYNC_TOUCH(sync_ptr) BGRT_SYSCALL_N(SYNC_TOUCH, (void *) (sync_ptr))`
- Пометить объект `bgrt_sync_t`, как грязный.

 - #define `BGRT_SYNC_SLEEP(sync_ptr, touch_ptr) BGRT_SYSCALL_NVAR(SYNC_SLEEP, (void *) (sync_ptr), (void *) (touch_ptr))`
- "Уснуть" в ожидании синхронизации `bgrt_sync_t`.

 - #define `BGRT_SYNC_WAKE(sync_ptr, pid, chown) BGRT_SYSCALL_NVAR(SYNC_WAKE, (void *) (sync_ptr), (void *) (pid), (int) (chown))`
- "Разбудить" ожидающий процесс.

 - #define `BGRT_SYNC_WAIT(sync_ptr, pid_ptr, block) BGRT_SYSCALL_NVAR(SYNC_WAIT, (void *) (sync_ptr), (void *) (pid_ptr), (int) (block))`
- "Ожидать", блокировки процесса.

 - #define `BGRT_SYNC_PROC_TIMEOUT(pid) BGRT_SYSCALL_N(SYNC_PROC_TIMEOUT, (void *) (pid))`
- "Разбудить", процесс по таймауту.

5.6.1 Подробное описание

Заголовок системных вызовов.

Предупреждения

Все содержимое файла для внутреннего использования!

5.6.2 Макросы

5.6.2.1 `BGRT_PROC_FREE` `#define BGRT_PROC_FREE() BGRT_SYSCALL_N(PROC_FREE, (void *)0)`

Останов процесса по флагу `BGRT_PROC_FLG_PRE_STOP`.

5.6.2.2 `BGRT_PROC_GET_ID` `#define BGRT_PROC_GET_ID(pid_ptr) (*(pid_ptr) = BGRT_PID_NOTHING, BGRT_SYSCALL_N(PROC_GET_ID, (void *) (pid_ptr)))`

Получить идентификатор текущего процесса.

Аргументы

<code>pid_ptr</code>	- Указатель на буфер чтения идентификатора процесса.
----------------------	--

Возвращает

- Результат выполнения системного вызова.

5.6.2.3 `BGRT_PROC_GET_PRIO` `#define BGRT_PROC_GET_PRIO(pid, pri_ptr) (*(pri_ptr) = BGRT_PRIO_LOWEST+1, BGRT_SYSCALL_NVAR(PROC_GET_PRIO, (void *) (pri_ptr), (void *) (pid)))`

Получить приоритет процесса.

Аргументы

<code>pid</code>	- Идентификатор процесса.
<code>pri_ptr</code>	- Указатель на буфер для чтения приоритета.

Возвращает

- Результат выполнения системного вызова.

5.6.2.4 `BGRT_PROC_LOCK` `#define BGRT_PROC_LOCK() BGRT_SYSCALL_N(PROC_LOCK, (void *)0)`

Установка флага `BGRT_PROC_FLG_LOCK` для вызывающего процесса.

5.6.2.5 `BGRT_PROC_RESET_WATCHDOG` `#define BGRT_PROC_RESET_WATCHDOG() BGRT_SYSCALL_N(P←ROC_RESET_WATCHDOG, (void *)0)`

Сброс watchdog для процесса реального времени.

Если функцию вызывает процесс реального времени, то функция сбрасывает его таймер. Если процесс завис, и таймер не был вовремя сброшен, то планировщик остановит такой процесс и передаст управление другому.

5.6.2.6 `BGRT_PROC_RESTART` `#define BGRT_PROC_RESTART(pid) BGRT_SYSCALL_N(PROC_RESTART, (void *) (pid))`

Перезапуск процесса.

Если можно (процесс не запущен, завершил работу, не был "убит"), приводит структуру `proc` в состояние, которое было после вызова `bgrt_proc_init`, и ставит процесс в список готовых к выполнению, и производит перепланировку.

Аргументы

<code>pid</code>	- Идентификатор процесса.
------------------	---------------------------

Возвращает

`BGRT_ST_OK` - если процесс был вставлен в список готовых к выполнению, либо код ошибки.

5.6.2.7 `BGRT_PROC_RUN` `#define BGRT_PROC_RUN(pid) BGRT_SYSCALL_N(PROC_RUN, (void *) (pid))`

Запуск процесса.

Ставит процесс в список готовых к выполнению, если можно (процесс не запущен, ещё не завершил работу, не был "убит"), и производит перепланировку.

Аргументы

pid	- Идентификатор процесса.
-----	---------------------------

Возвращает

BGRT_ST_OK - если процесс был вставлен в список готовых к выполнению, либо код ошибки.

5.6.2.8 BGRT_PROC_SELF_STOP #define BGRT_PROC_SELF_STOP() BGRT_SYSCALL_N(PROC_SELF_STOP, (void *)0)

Самоостанов процесса.

Вырезает вызывающий процесс из списка готовых к выполнению и производит перепланировку.

5.6.2.9 BGRT_PROC_SET_PRIO #define BGRT_PROC_SET_PRIO(
pid,
prio) BGRT_SYSCALL_NVAR(PROC_SET_PRIO, (void *) (pid), (int) (prio))

Управление приоритетом процесса.

Устанавливает приоритет процесса, находящегося в любом состоянии.

Аргументы

pid	- Идентификатор процесса.
prio	- Новое значение приоритета.

Возвращает

- Результат выполнения системного вызова.

5.6.2.10 BGRT_PROC_STOP #define BGRT_PROC_STOP(
pid) BGRT_SYSCALL_N(PROC_STOP, (void *) (pid))

Останов процесса.

Вырезает процесс из списка готовых к выполнению и производит перепланировку.

Аргументы

pid	- Идентификатор процесса.
-----	---------------------------

Возвращает

BGRT_ST_OK - если процесс был вырезан из списка готовых к выполнению, либо код ошибки.

```
5.6.2.11 BGRT_SYNC_GET_OWNER #define BGRT_SYNC_GET_OWNER(
    sync_ptr,
    pid_ptr ) (*(pid_ptr) = BGRT_PID_NOTHING, BGRT_SYSCALL_NVAR(SYNC_GET_OWNER,
(void *) (pid_ptr), (void *) (sync_ptr)))
```

Получить хозяина примитива.

Аргументы

sync_ptr	- Указатель на интересующий объект типа bgrt_sync_t.
pid_ptr	- Указатель на буфер чтения идентификатора процесса.

Возвращает

Указатель на процесс-хозяин объекта типа bgrt_sync_t.

```
5.6.2.12 BGRT_SYNC_OWN #define BGRT_SYNC_OWN(
    sync_ptr,
    touch ) BGRT_SYSCALL_NVAR(SYNC_OWN, (void *) (sync_ptr), (int) (touch))
```

Завладеть объектом типа bgrt_sync_t.

Аргументы

sync_ptr	Указатель на объект типа bgrt_sync_t.
touch	Если не 0, - пометить sync, как "грязный" случае неудачи.

Возвращает

BGRT_ST_OK в случае успеха, либо код ошибки.

```
5.6.2.13 BGRT_SYNC_PROC_TIMEOUT #define BGRT_SYNC_PROC_TIMEOUT(
    pid ) BGRT_SYSCALL_N(SYNC_PROC_TIMEOUT, (void *) (pid))
```

"Разбудить", процесс по таймауту.

Аргументы

pid	Указатель на процесс, который надо подождать, если *pid==0, то вызывающий процесс будет ждать первой блокировки процесса на объекте типа bgrt_sync_t.
-----	---

Возвращает

BGRT_ST_OK в случае, если дождался разбудил целевой процесс, BGRT_ST_EAGAIN, если нужна следующая итерация, иначе - код ошибки.

```
5.6.2.14 BGRT_SYNC_SET_OWNER #define BGRT_SYNC_SET_OWNER(
    sync_ptr,
    pid ) BGRT_SYSCALL_NVAR(SYNC_SET_OWNER, (void *) (sync_ptr), (void *) (pid))
```

Назначить хозяина объекта типа bgrt_sync_t.

Аргументы

sync_ptr	Указатель на объект типа bgrt_sync_t.
pid	Уникальный идентификатор нового процесса-хозяин объекта типа bgrt_sync_t.

Возвращает

BGRT_ST_OK в случае успеха, либо код ошибки.

```
5.6.2.15 BGRT_SYNC_SLEEP #define BGRT_SYNC_SLEEP(
    sync_ptr,
    touch_ptr ) BGRT_SYSCALL_NVAR(SYNC_SLEEP, (void *) (sync_ptr), (void *) (touch_ptr))
```

"Уснуть" в ожидании синхронизации bgrt_sync_t.

Блокирует вызывающий процесс.

Аргументы

sync_ptr	Указатель на объект типа bgrt_sync_t.
touch_ptr	Указатель на переменную, хранящую флаг touch. Если там не 0, то sync был помечен, как "грязный".

Возвращает

BGRT_ST_OK в случае успеха, иначе - код ошибки.

```
5.6.2.16 BGRТ_SYNC_TOUCH #define BGRТ_SYNC_TOUCH(
    sync_ptr ) BGRТ_SYSCALL_N(SYNC_TOUCH, (void *) (sync_ptr))
```

Пометить объект `bgrt_sync_t`, как грязный.

Аргументы

<code>sync_ptr</code>	Указатель на объект типа <code>bgrt_sync_t</code> .
-----------------------	---

Возвращает

`BGRТ_ST_OK` в случае успеха, либо код ошибки.

```
5.6.2.17 BGRТ_SYNC_WAIT #define BGRТ_SYNC_WAIT(
    sync_ptr,
    pid_ptr,
    block ) BGRТ_SYSCALL_NVAR(SYNC_WAIT, (void *) (sync_ptr), (void *) (pid_ptr), (int) (block))
```

"Ожидать", блокировки процесса.

Подождать того момента, как целевой процесс будет заблокирован на целевом примитиве синхронизации.

Аргументы

<code>sync_ptr</code>	Указатель на объект типа <code>bgrt_sync_t</code> .
<code>pid_ptr</code>	Указатель на идентификатор процесса, который надо подождать, если <code>*pid==BGRТ_PID_NOTHING</code> , то вызывающий процесс будет ждать первой блокировки процесса на объекте типа <code>bgrt_sync_t</code> .
<code>block</code>	Флаг блокировки вызывающего процесса, если не 0 и нужно ждать, вызывающий процесс будет заблокирован.

Возвращает

`BGRТ_ST_OK` в случае если дождался блокировки целевого процесса, иначе - код ошибки.

```
5.6.2.18 BGRТ_SYNC_WAKE #define BGRТ_SYNC_WAKE(
    sync_ptr,
    pid,
    chown ) BGRТ_SYSCALL_NVAR(SYNC_WAKE, (void *) (sync_ptr), (void *) (pid), (int) (chown))
```

"Разбудить" ожидающий процесс.

Запускает ожидающий процесс. Может запустить "голову" списка ожидающих процессов, или какой-то конкретный процесс, в случае, если он заблокирован на целевом примитиве синхронизации.

Аргументы

sync_ptr	Указатель на объект типа bgrt_sync_t.
pid	Идентификатор процесса, который надо запустить, если BGRT_PID_NOTHING, то пытается запустить "голову" списка ожидающих.
chown	Флаг смены хозяина, если не 0, то запускаемый процесс станет новым хозяином примитива синхронизации.

Возвращает

BGRT_ST_OK в случае если удалось запустить процесс, иначе - код ошибки.

5.7 Файл bugurtos/kernel/default/syscall_routines.h

```
#include <bugurt.h>
```

Определения типов

- typedef bgrt_st_t(* bgrt_user_func_t) (bgrt_va_wr_t *)

Функции

- BGRT_SC_SR (PROC_RUN, void *arg)
- BGRT_SC_SR (PROC_RESTART, void *arg)
- BGRT_SC_SR (PROC_STOP, void *arg)
- BGRT_SC_SR (PROC_SELF_STOP, void *arg)
- BGRT_SC_SR (PROC_LOCK, void *arg)
- BGRT_SC_SR (PROC_FREE, void *arg)
- BGRT_SC_SR (PROC_RESET_WATCHDOG, void *arg)
- BGRT_SC_SR (PROC_GET_PRIO, bgrt_va_wr_t *va)
- BGRT_SC_SR (PROC_SET_PRIO, bgrt_va_wr_t *va)
- BGRT_SC_SR (PROC_GET_ID, void *arg)
- BGRT_SC_SR (SYNC_SET_OWNER, bgrt_va_wr_t *va)
- BGRT_SC_SR (SYNC_GET_OWNER, bgrt_va_wr_t *va)
- BGRT_SC_SR (SYNC_OWN, bgrt_va_wr_t *va)
- BGRT_SC_SR (SYNC_TOUCH, void *arg)
- BGRT_SC_SR (SYNC_SLEEP, bgrt_va_wr_t *va)
- BGRT_SC_SR (SYNC_WAKE, bgrt_va_wr_t *va)
- BGRT_SC_SR (SYNC_WAIT, bgrt_va_wr_t *va)
- BGRT_SC_SR (SYNC_PROC_TIMEOUT, void *arg)
- BGRT_SC_SR (USER, bgrt_va_wr_t *va)

5.7.1 Типы

5.7.1.1 bgrt_user_func_t typedef bgrt_st_t(* bgrt_user_func_t) (bgrt_va_wr_t *)

5.7.2 Функции

5.7.2.1 BGRT_SC_SR() [1/19] BGRT_SC_SR (PROC_FREE , void * arg)

5.7.2.2 BGRT_SC_SR() [2/19] BGRT_SC_SR (PROC_GET_ID , void * arg)

5.7.2.3 BGRT_SC_SR() [3/19] BGRT_SC_SR (PROC_GET_PRIO , bgrt_va_wr_t * va)

5.7.2.4 BGRT_SC_SR() [4/19] BGRT_SC_SR (PROC_LOCK , void * arg)

5.7.2.5 BGRT_SC_SR() [5/19] BGRT_SC_SR (PROC_RESET_WATCHDOG , void * arg)

5.7.2.6 BGRT_SC_SR() [6/19] BGRT_SC_SR (PROC_RESTART , void * arg)

5.7.2.7 BGRT_SC_SR() [7/19] BGRT_SC_SR (PROC_RUN , void * arg)

5.7.2.8 BGRT_SC_SR() [8/19] BGRT_SC_SR (PROC_SELF_STOP , void * arg)

5.7.2.9 `BGRT_SC_SR()` [9/19] `BGRT_SC_SR (`
 `PROC_SET_PRIO ,`
 `bgrt_va_wr_t * va)`

5.7.2.10 `BGRT_SC_SR()` [10/19] `BGRT_SC_SR (`
 `PROC_STOP ,`
 `void * arg)`

5.7.2.11 `BGRT_SC_SR()` [11/19] `BGRT_SC_SR (`
 `SYNC_GET_OWNER ,`
 `bgrt_va_wr_t * va)`

5.7.2.12 `BGRT_SC_SR()` [12/19] `BGRT_SC_SR (`
 `SYNC_OWN ,`
 `bgrt_va_wr_t * va)`

5.7.2.13 `BGRT_SC_SR()` [13/19] `BGRT_SC_SR (`
 `SYNC_PROC_TIMEOUT ,`
 `void * arg)`

5.7.2.14 `BGRT_SC_SR()` [14/19] `BGRT_SC_SR (`
 `SYNC_SET_OWNER ,`
 `bgrt_va_wr_t * va)`

5.7.2.15 `BGRT_SC_SR()` [15/19] `BGRT_SC_SR (`
 `SYNC_SLEEP ,`
 `bgrt_va_wr_t * va)`

5.7.2.16 `BGRT_SC_SR()` [16/19] `BGRT_SC_SR (`
 `SYNC_TOUCH ,`
 `void * arg)`

5.7.2.17 BGRT_SC_SR() [17/19] BGRT_SC_SR (
SYNC_WAIT ,
bgrt_va_wr_t * va)

5.7.2.18 BGRT_SC_SR() [18/19] BGRT_SC_SR (
SYNC_WAKE ,
bgrt_va_wr_t * va)

5.7.2.19 BGRT_SC_SR() [19/19] BGRT_SC_SR (
USER ,
bgrt_va_wr_t * va)

5.8 Файл bugurtos/kernel/index.h

Заголовок функции поиска в бинарном индексе.

Функции

- bgrt_prio_t bgrt_map_search (bgrt_map_t map)
Поиск в бинарном индексе.

5.8.1 Подробное описание

Заголовок функции поиска в бинарном индексе.

5.8.2 Функции

5.8.2.1 bgrt_map_search() bgrt_prio_t bgrt_map_search (
bgrt_map_t map)

Поиск в бинарном индексе.

Предупреждения

Для внутреннего использования.

Аргументы

map	Бинарный индекс.
-----	------------------

Возвращает

Наивысший (с минимальным значением) приоритет в индексе.

5.9 Файл `bugurtos/kernel/item.h`

Заголовок элементов 2-связного списка.

Структуры данных

- `struct bgrt_priv_item_t`
Элемент 2-связного списка.

Макросы

- `#define BGRT_ITEM_T_INIT(a) { (bgrt_item_t *)&a, (bgrt_item_t *)&a }`

Определения типов

- `typedef struct bgrt_priv_item_t bgrt_item_t`

Функции

- `void bgrt_item_init (bgrt_item_t *item)`
Инициализация объекта типа `bgrt_item_t`.
- `void bgrt_item_insert (bgrt_item_t *item, bgrt_item_t *head)`
Вставка элемента типа `bgrt_item_t` в список.
- `void bgrt_item_cut (bgrt_item_t *item)`
Вырезать элемент типа `bgrt_item_t` из списка.

5.9.1 Подробное описание

Заголовок элементов 2-связного списка.

5.9.2 Макросы

5.9.2.1 `BGRT_ITEM_T_INIT` `#define BGRT_ITEM_T_INIT(a) { (bgrt_item_t *)&a, (bgrt_item_t *)&a }`

Статическая инициализация объекта типа `bgrt_item_t`.

Предупреждения

Для внутреннего использования.

Аргументы

a	Имя переменной типа <code>bgrt_item_t</code> .
---	--

5.9.3 Типы

5.9.3.1 `bgrt_item_t` `typedef struct bgrt_priv_item_t bgrt_item_t`

Смотри `bgrt_priv_item_t`;

5.9.4 Функции

5.9.4.1 `bgrt_item_cut()` `void bgrt_item_cut (`
`bgrt_item_t * item)`

Вырезать элемент типа `bgrt_item_t` из списка.

Предупреждения

Для внутреннего использования.

Аргументы

item	Указатель на объект типа <code>bgrt_item_t</code> , который будем вырезать.
------	---

5.9.4.2 `bgrt_item_init()` `void bgrt_item_init (`
`bgrt_item_t * item)`

Инициализация объекта типа `bgrt_item_t`.

Предупреждения

Для внутреннего использования.

Аргументы

item	Указатель на объект <code>bgrt_item_t</code> .
------	--

```
5.9.4.3 bgrt_item_insert() void bgrt_item_insert (
    bgrt_item_t * item,
    bgrt_item_t * head )
```

Вставка элемента типа `bgrt_item_t` в список.

Предупреждения

Для внутреннего использования.

Аргументы

item	Указатель на объект типа <code>bgrt_item_t</code> , который будем вставлять.
head	Указатель на голову списка типа <code>bgrt_item_t</code> .

5.10 Файл `bugurtos/kernel/kernel.h`

Заголовок Ядра.

Структуры данных

- struct `bgrt_priv_kblock_t`
Блок ядра `BuguRTOS`.
- struct `bgrt_priv_kernel_t`
Ядро `BuguRTOS`.

Макросы

- `#define BGRT_KBLOCK_VSCALL ((bgrt_map_t)0x1)`
- `#define BGRT_KBLOCK_VTMR ((bgrt_map_t)0x2)`
- `#define BGRT_KBLOCK_VRESCH ((bgrt_map_t)0x4)`
- `#define BGRT_KBLOCK_VSCHMSK ((bgrt_map_t)0x6)`
- `#define BGRT_KBLOCK_PWRSV ((bgrt_map_t)0x8)`

Определения типов

- `typedef struct bgrt_priv_kblock_t bgrt_kblock_t`
- `typedef struct bgrt_priv_kernel_t bgrt_kernel_t`

Функции

- void `bgrt_kblock_init (bgrt_kblock_t *kblock)`
Инициализация объекта типа `bgrt_kblock_t`.
- void `bgrt_kblock_do_work (bgrt_kblock_t *kblock)`
Обработка программных прерываний.
- void `bgrt_kblock_main (bgrt_kblock_t *kblock)`
Главная функция потока Ядра.
- void `bgrt_kernel_init (void)`
Инициализация ядра.

Переменные

- `bgrt_kernel_t bgrt_kernel`
Ядро VuguRTOS.

5.10.1 Подробное описание

Заголовок Ядра.

5.10.2 Макросы

5.10.2.1 `BGRT_KBLOCK_PWRSV` `#define BGRT_KBLOCK_PWRSV ((bgrt_map_t)0x8)`

Вектор перехода в энергосбережение.

5.10.2.2 `BGRT_KBLOCK_VRESCH` `#define BGRT_KBLOCK_VRESCH ((bgrt_map_t)0x4)`

Вектор перепланировки.

5.10.2.3 `BGRT_KBLOCK_VSCALL` `#define BGRT_KBLOCK_VSCALL ((bgrt_map_t)0x1)`

Вектор системного вызова.

5.10.2.4 `BGRT_KBLOCK_VSCHMSK` `#define BGRT_KBLOCK_VSCHMSK ((bgrt_map_t)0x6)`

Маска векторов планировщика.

5.10.2.5 `BGRT_KBLOCK_VTMR` `#define BGRT_KBLOCK_VTMR ((bgrt_map_t)0x2)`

Вектор системног таймера.

5.10.3 Типы

5.10.3.1 `bgrt_kblock_t` `typedef struct bgrt_priv_kblock_t bgrt_kblock_t`

Смотри `bgrt_priv_kblock_t`;

5.10.3.2 `bgrt_kernel_t` `typedef struct bgrt_priv_kernel_t bgrt_kernel_t`

Смотри `bgrt_priv_kernel_t`;

5.10.4 Функции

5.10.4.1 `bgrt_kblock_do_work()` `void bgrt_kblock_do_work (`
`bgrt_kblock_t * kblock)`

Обработка программных прерываний.

Аргументы

kblock	Указатель на объект типа bgrt_kblock_t.
--------	---

5.10.4.2 bgrt_kblock_init() void bgrt_kblock_init (
bgrt_kblock_t * kblock)

Инициализация объекта типа bgrt_kblock_t.

Аргументы

kblock	Указатель на объект типа bgrt_kblock_t.
--------	---

5.10.4.3 bgrt_kblock_main() void bgrt_kblock_main (
bgrt_kblock_t * kblock)

Главная функция потока Ядра.

Аргументы

kblock	Указатель на объект типа bgrt_kblock_t.
--------	---

5.10.4.4 bgrt_kernel_init() void bgrt_kernel_init (
void)

Инициализация ядра.

Готовит ядро к запуску.

Предупреждения

Для внутреннего использования.

5.10.5 Переменные

5.10.5.1 bgrt_kernel bgrt_kernel_t bgrt_kernel

Ядро VuguRTOS.

Оно одно на всю систему!

5.11 Файл `bugurtos/kernel/pcounter.h`

Заголовок счётчиков захваченных ресурсов.

Структуры данных

- `struct bgrt_priv_pcounter_t`
Счётчик захваченных ресурсов.

Макросы

- `#define BGRT_CNT_INC(cnt) (cnt = bgrt_cnt_inc(cnt))`
- `#define BGRT_CNT_DEC(cnt) (cnt = bgrt_cnt_dec(cnt))`
- `#define BGRT_CNT_ADD(cnt, delta) (cnt = bgrt_cnt_add(cnt, delta))`
- `#define BGRT_CNT_SUB(cnt, delta) (cnt = bgrt_cnt_sub(cnt, delta))`

Определения типов

- `typedef struct bgrt_priv_pcounter_t bgrt_pcounter_t`

Функции

- `bgrt_cnt_t bgrt_cnt_inc (bgrt_cnt_t val)`
Инкремент счётчика.
- `bgrt_cnt_t bgrt_cnt_dec (bgrt_cnt_t val)`
Декремент счётчика.
- `bgrt_cnt_t bgrt_cnt_add (bgrt_cnt_t a, bgrt_cnt_t b)`
Добавление к значению счётчика.
- `bgrt_cnt_t bgrt_cnt_sub (bgrt_cnt_t a, bgrt_cnt_t b)`
Вычитание из значения счётчика.
- `void bgrt_pcounter_init (bgrt_pcounter_t *pcounter)`
Инициализация счётчика.
- `void bgrt_pcounter_inc (bgrt_pcounter_t *pcounter, bgrt_prio_t prio)`
Инкремент счётчика.
- `bgrt_map_t bgrt_pcounter_dec (bgrt_pcounter_t *pcounter, bgrt_prio_t prio)`
Декремент счётчика.
- `void bgrt_pcounter_plus (bgrt_pcounter_t *pcounter, bgrt_prio_t prio, bgrt_cnt_t count)`
Увеличение счётчика на произвольное количество единиц.
- `bgrt_map_t bgrt_pcounter_minus (bgrt_pcounter_t *pcounter, bgrt_prio_t prio, bgrt_cnt_t count)`
Уменьшение счётчика на произвольное количество единиц.

5.11.1 Подробное описание

Заголовок счётчиков захваченных ресурсов.

5.11.2 Макросы

5.11.2.1 `BGRT_CNT_ADD` `#define BGRT_CNT_ADD(
cnt,
delta) (cnt = bgrt_cnt_add(cnt, delta))`

Обёртка над `bgrt_cnt_add`;

5.11.2.2 `BGRT_CNT_DEC` `#define BGRT_CNT_DEC(
cnt) (cnt = bgrt_cnt_dec(cnt))`

Обёртка над `bgrt_cnt_dec`;

5.11.2.3 `BGRT_CNT_INC` `#define BGRT_CNT_INC(
cnt) (cnt = bgrt_cnt_inc(cnt))`

Обёртка над `bgrt_cnt_inc`;

5.11.2.4 `BGRT_CNT_SUB` `#define BGRT_CNT_SUB(
cnt,
delta) (cnt = bgrt_cnt_sub(cnt, delta))`

Обёртка над `bgrt_cnt_sub`;

5.11.3 Типы

5.11.3.1 `bgrt_pcounter_t` `typedef struct bgrt_priv_pcounter_t bgrt_pcounter_t`

Смотри `bgrt_priv_pcounter_t`;

5.11.4 Функции

5.11.4.1 `bgrt_cnt_add()` `bgrt_cnt_t bgrt_cnt_add (
bgrt_cnt_t a,
bgrt_cnt_t b)`

Добавление к значению счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

a	Текущее значение счётчика.
b	Изменение.

Возвращает

Новое значение счётчика.

```
5.11.4.2 bgrt_cnt_dec() bgrt_cnt_t bgrt_cnt_dec (
    bgrt_cnt_t val )
```

Декремент счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

val	Текущее значение счётчика.
-----	----------------------------

Возвращает

Новое значение счётчика.

```
5.11.4.3 bgrt_cnt_inc() bgrt_cnt_t bgrt_cnt_inc (
    bgrt_cnt_t val )
```

Инкремент счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

val	Текущее значение счётчика.
-----	----------------------------

Возвращает

Новое значение счётчика.

5.11.4.4 `bgrt_cnt_sub()` `bgrt_cnt_t bgrt_cnt_sub (`
`bgrt_cnt_t a,`
`bgrt_cnt_t b)`

Вычитание из значения счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

a	Текущее значение счётчика.
b	Изменение.

Возвращает

Новое значение счётчика.

5.11.4.5 `bgrt_pcounter_dec()` `bgrt_map_t bgrt_pcounter_dec (`
`bgrt_pcounter_t * pcounter,`
`bgrt_prio_t prio)`

Декремент счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

pcounter	Указатель на счётчик.
prio	Приоритет.

5.11.4.6 `bgrt_pcounter_inc()` `void bgrt_pcounter_inc (`
`bgrt_pcounter_t * pcounter,`
`bgrt_prio_t prio)`

Инкремент счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

<code>pcounter</code>	Указатель на счётчик.
<code>prio</code>	Приоритет.

5.11.4.7 `bgrt_pcounter_init()` `void bgrt_pcounter_init (`
`bgrt_pcounter_t * pcounter)`

Инициализация счётчика.

Предупреждения

Для внутреннего использования.

Аргументы

<code>pcounter</code>	Указатель на счётчик.
-----------------------	-----------------------

5.11.4.8 `bgrt_pcounter_minus()` `bgrt_map_t bgrt_pcounter_minus (`
`bgrt_pcounter_t * pcounter,`
`bgrt_prio_t prio,`
`bgrt_cnt_t count)`

Уменьшение счётчика на произвольное количество единиц.

Предупреждения

Для внутреннего использования.

Аргументы

<code>pcounter</code>	Указатель на счётчик.
<code>prio</code>	Приоритет.
<code>count</code>	Количество единиц.

Возвращает

0 - если соответствующая часть счётчика обнулилась, не 0 - в других случаях.

```
5.11.4.9 bgrt_pcounter_plus() void bgrt_pcounter_plus (
    bgrt_pcounter_t * pcounter,
    bgrt_prio_t prio,
    bgrt_cnt_t count )
```

Увеличение счётчика на произвольное количество единиц.

Предупреждения

Для внутреннего использования.

Аргументы

pcounter	Указатель на счётчик.
prio	Приоритет.
count	Количество единиц.

5.12 Файл bugurtos/kernel/pitem.h

Заголовок элементов списка с приоритетами.

Структуры данных

- struct `bgrt_priv_pitem_t`
Элемент списка с приоритетами

Макросы

- `#define BGRT_PITEM_T_INIT(a, p) { BGRT_ITEM_T_INIT(a), (bgrt_xlist_t *)0, (bgrt_prio_t)p }`

Определения типов

- `typedef struct bgrt_priv_pitem_t bgrt_pitem_t`

Функции

- void `bgrt_pitem_init (bgrt_pitem_t *pitem, bgrt_prio_t prio)`
Инициализация объект a типа `bgrt_pitem_t`.
- void `bgrt_pitem_insert (bgrt_pitem_t *pitem, bgrt_xlist_t *xlist)`
Вставка элемента типа `bgrt_pitem_t` в список типа `bgrt_xlist_t`.
- void `bgrt_pitem_fast_cut (bgrt_pitem_t *pitem)`
Быстро вырезать из списка.
- void `bgrt_pitem_cut (bgrt_pitem_t *pitem)`
Вырезать из списка.
- `bgrt_pitem_t * bgrt_pitem_xlist_chain (bgrt_xlist_t *src)`
"Сцепить" список типа `bgrt_xlist_t`.

5.12.1 Подробное описание

Заголовок элементов списка с приоритетами.

5.12.2 Макросы

5.12.2.1 `BGRT_PITEM_T_INIT` `#define BGRT_PITEM_T_INIT(`
`a,`
`p) { BGRT_ITEM_T_INIT(a), (bgrt_xlist_t *)0, (bgrt_prio_t)p }`

Статическая инициализация объекта типа `bgrt_pitem_t`

Предупреждения

Для внутреннего использования.

Аргументы

a	Имя переменной.
p	Приоритет.

5.12.3 Типы

5.12.3.1 `bgrt_pitem_t` `typedef struct bgrt_priv_pitem_t bgrt_pitem_t`

5.12.4 Функции

5.12.4.1 `bgrt_pitem_cut()` `void bgrt_pitem_cut (`
`bgrt_pitem_t * pitem)`

Вырезать из списка.

Вызывает `bgrt_pitem_fast_cut` и обнуляет указатель `pitem->list`.

Предупреждения

Для внутреннего использования.

Аргументы

pitem	Указатель на объект bgrt_pitem_t.
-------	-----------------------------------

5.12.4.2 `bgrt_pitem_fast_cut()` `void bgrt_pitem_fast_cut (`
`bgrt_pitem_t * pitem)`

Быстро вырезать из списка.

Вырезает объект типа `bgrt_pitem_t`, из списка типа `bgrt_xlist_t`, не обнуляет указатель `pitem->list`.

Предупреждения

Для внутреннего использования.

Аргументы

pitem	Указатель на объект bgrt_pitem_t.
-------	-----------------------------------

5.12.4.3 `bgrt_pitem_init()` `void bgrt_pitem_init (`
`bgrt_pitem_t * pitem,`
`bgrt_prio_t prio)`

Инициализация объект а типа `bgrt_pitem_t`.

Предупреждения

Для внутреннего использования.

Аргументы

pitem	Указатель на объект bgrt_pitem_t.
prio	Приоритет элемента.

5.12.4.4 `bgrt_pitem_insert()` `void bgrt_pitem_insert (`
`bgrt_pitem_t * pitem,`
`bgrt_xlist_t * xlist)`

Вставка элемента типа `bgrt_pitem_t` в список типа `bgrt_xlist_t`.

Предупреждения

Для внутреннего использования.

Аргументы

pittem	Указатель на объект bgrt_pittem_t.
xlist	Указатель на список.

5.12.4.5 `bgrt_pittem_xlist_chain()` `bgrt_pittem_t*` `bgrt_pittem_xlist_chain (bgrt_xlist_t * src)`

"Сцепить" список типа `bgrt_xlist_t`.

Вырезать из списка типа `bgrt_xlist_t` все элементы типа `bgrt_pittem_t` и сделать из них простой 2-связный список.

Предупреждения

Для внутреннего использования.

Аргументы

src	Указатель на объект <code>bgrt_xlist_t</code> .
-----	---

Возвращает

Указатель на голову 2-связного списка.

5.13 Файл bugurtos/kernel/proc.h

Заголовок процессов.

Структуры данных

- struct `bgrt_priv_uspd_t`
- struct `bgrt_priv_proc_t`

Процесс.

Макросы

- `#define BGRT_PROC_LRES_INIT(a) bgrt_pcounter_init(&((a)->lres))`
Макрос-обёртка.
- `#define BGRT_PROC_LRES_INC(a, b) bgrt_pcounter_inc(&((a)->lres), (bgrt_prio_t)b)`
Макрос-обёртка.
- `#define BGRT_PROC_LRES_DEC(a, b) bgrt_pcounter_dec(&((a)->lres), (bgrt_prio_t)b)`
Макрос-обёртка.
- `#define BGRT_PID_T bgrt_proc_t *`
Уникальный идентификатор процесса.

- #define `BGRT_PID_TO_PROC(p)` (p)
Преобразование идентификатора процесса в указатель на процесс.
- #define `BGRT_PROC_TO_PID(p)` (p)
Преобразование указателя на процесс в идентификатор процесса.
- #define `BGRT_PID_NOTHING` ((`BGRT_PID_T`)0)
Пустой идентификатор процесса.
- #define `BGRT_USPD_PROC_T` struct `bgrt_priv_uspd_t`
- #define `BGRT_USPD_T` `BGRT_USPD_PROC_T *`
- #define `BGRT_GET_USPD()` (&(bgrt_curr_proc()->udata))
- #define `BGRT_USPD_INIT(proc)`
- #define `BGRT_PROC_FLG_RT` ((`bgrt_flag_t`)0x80)
Флаг реального времени.
- #define `BGRT_PROC_FLG_RR` ((`bgrt_flag_t`)0x40)
Флаг карусельной многозадачности.
- #define `BGRT_PROC_FLG_LOCK` ((`bgrt_flag_t`)0x20)
Флаг блокировки останова процесса.
- #define `BGRT_PROC_FLG_PRE_STOP` ((`bgrt_flag_t`)0x10)
Флаг запроса останова.
- #define `BGRT_PROC_FLG_LOCK_MASK` ((`bgrt_flag_t`)(`BGRT_PROC_FLG_LOCK`))
Маска `BGRT_PROC_FLG_LOCK`.
- #define `BGRT_PROC_STATE_CLEAR_MASK` ((`bgrt_flag_t`)0xF0)
Маска очистки состояния исполнения процесса.
- #define `BGRT_PROC_STATE_CLEAR_RUN_MASK` ((`bgrt_flag_t`)0xFC)
Маска очистки состояния исполнения процесса.
- #define `BGRT_PROC_STATE_MASK` ((`bgrt_flag_t`)0x0F)
Маска состояния исполнения процесса.
- #define `BGRT_PROC_STATE_RESTART_MASK` ((`bgrt_flag_t`)0x8)
Маска проверки состояния процесса.
- #define `BGRT_PROC_STATE_RUN_MASK` ((`bgrt_flag_t`)0x3)
Маска проверки состояния процесса.
- #define `BGRT_PROC_STATE_WAIT_MASK` ((`bgrt_flag_t`)0x8)
Маска проверки состояния процесса.
- #define `BGRT_PROC_STATE_STOPED` ((`bgrt_flag_t`)0x0)
Начальное состояние, остановлен.
- #define `BGRT_PROC_STATE_END` ((`bgrt_flag_t`)0x1)
Завершен.
- #define `BGRT_PROC_STATE_READY` ((`bgrt_flag_t`)0x2)
Готов к выполнению.
- #define `BGRT_PROC_STATE_RUNNING` ((`bgrt_flag_t`)0x3)
Выполняется.
- #define `BGRT_PROC_STATE_WD_STOPED` ((`bgrt_flag_t`)0x4)
Остановлен по вачдог.
- #define `BGRT_PROC_STATE_DEAD` ((`bgrt_flag_t`)0x5)
Завершен до завершения ирс-транзакций.
- #define `BGRT_PROC_STATE_TO_READY` ((`bgrt_flag_t`)0x6)
Готов к выполнению.
- #define `BGRT_PROC_STATE_TO_RUNNING` ((`bgrt_flag_t`)0x7)
Выполняется.
- #define `BGRT_PROC_STATE_SYNC_WAIT` ((`bgrt_flag_t`)0x8)
Ожидает приема спящих процессов.
- #define `BGRT_PROC_STATE_SYNC_SLEEP` ((`bgrt_flag_t`)0x9)

- Ожидает пробуждения.
- `#define BGRT_PROC_STATE_SYNC_READY ((bgrt_flag_t)0xA)`
Готов к выполнению.
- `#define BGRT_PROC_STATE_SYNC_RUNNING ((bgrt_flag_t)0xB)`
Выполняется.
- `#define BGRT_PROC_STATE_PI_PEND ((bgrt_flag_t)0xC)`
Ожидает смены приоритета
- `#define BGRT_PROC_STATE_PI_DONE ((bgrt_flag_t)0xD)`
Запущен при смене приоритета
- `#define BGRT_PROC_STATE_PI_READY ((bgrt_flag_t)0xE)`
Готов к выполнению.
- `#define BGRT_PROC_STATE_PI_RUNNING ((bgrt_flag_t)0xF)`
Выполняется.
- `#define BGRT_PROC_PRE_STOP_TEST(a) (((a)->flags) & BGRT_PROC_FLG_PRE_STOP && (!(a)->flags) & BGRT_PROC_FLG_LOCK_MASK))`
Макрос проверки условий останова по флагу BGRT_PROC_FLG_PRE_STOP.
- `#define BGRT_PROC_RUN_TEST(a) (((a)->flags & BGRT_PROC_STATE_RUN_MASK)>= BGRT_PROC_STATE_READY)`
Проверяет, запущен ли процесс.
- `#define BGRT_PROC_GET_STATE(a) ((a)->flags & BGRT_PROC_STATE_MASK)`
Читает состояние процесса.
- `#define BGRT_PROC_SET_STATE(a, b) ((a)->flags &= BGRT_PROC_STATE_CLEAR_MASK, (a)->flags |= b)`
Устанавливает состояние процесса.

Определения типов

- `typedef struct bgrt_priv_proc_t bgrt_proc_t`

Функции

- `void bgrt_priv_proc_lres_inc (bgrt_proc_t *proc, bgrt_prio_t prio)`
Управление приоритетом процесса.
- `void bgrt_priv_proc_lres_dec (bgrt_proc_t *proc, bgrt_prio_t prio)`
Управление приоритетом процесса.
- `void bgrt_priv_proc_stop_ensure (bgrt_proc_t *proc, bgrt_flag_t state)`
Останов процесса.
- `bgrt_st_t bgrt_priv_proc_init (bgrt_proc_t *proc, bgrt_code_t pmain, bgrt_code_t sv_hook, bgrt_code_t rs_hook, void *arg, bgrt_stack_t *sstart, bgrt_prio_t prio, bgrt_tmr_t time_← quant, bgrt_bool_t is_rt, bgrt_aff_t affinity)`
Инициализация процесса из обработчика прерывания, либо из критической секции.
- `bgrt_st_t bgrt_proc_init (bgrt_proc_t *proc, bgrt_code_t pmain, bgrt_code_t sv_hook, bgrt_code_t rs_hook, void *arg, bgrt_stack_t *sstart, bgrt_prio_t prio, bgrt_tmr_t time_← quant, bgrt_bool_t is_rt, bgrt_aff_t affinity)`
Инициализация процесса.
- `void bgrt_proc_terminate (void)`
Завершение работы процесса после возврата из `proc->pmain`.
- `void bgrt_priv_proc_terminate (void)`
Завершение работы процесса после возврата из `proc->pmain`.
- `bgrt_st_t bgrt_priv_proc_run (bgrt_proc_t *proc)`
Запуск процесса из критической секции, либо обработчика прерывания.

- `bgrt_st_t bgrt_priv_proc_restart (bgrt_proc_t *proc)`
Перезапуск процесса из критической секции или обработчика прерывания.
- `bgrt_st_t bgrt_priv_proc_stop (bgrt_proc_t *proc)`
Останов процесса из критической секции или обработчика прерывания.
- `void bgrt_priv_proc_self_stop (void)`
Самоостанов процесса.
- `void bgrt_priv_proc_reset_watchdog (void)`
Сброс watchdog для процесса реального времени из обработчика прерывания.
- `void bgrt_priv_proc_lock (void)`
Установка флага `BGRT_PROC_FLG_LOCK` для вызывающего процесса.
- `void bgrt_priv_proc_free (void)`
Останов процесса по флагу `BGRT_PROC_FLG_PRE_STOP` из критической секции или обработчика прерывания.
- `void bgrt_priv_proc_set_prio (bgrt_proc_t *proc, bgrt_prio_t prio)`
Управление приоритетом процесса.
- `bgrt_prio_t bgrt_priv_proc_get_prio (bgrt_proc_t *proc)`
Получить приоритет процесса.

5.13.1 Подробное описание

Заголовок процессов.

5.13.2 Макросы

5.13.2.1 `BGRT_GET_USPD` `#define BGRT_GET_USPD() (&(bgrt_curr_proc()->udata))`

Получить указатель на данные пространства пользователя текущего процесса.

5.13.2.2 `BGRT_PID_NOTHING` `#define BGRT_PID_NOTHING ((BGRT_PID_T)0)`

Пустой идентификатор процесса.

5.13.2.3 `BGRT_PID_T` `#define BGRT_PID_T bgrt_proc_t *`

Уникальный идентификатор процесса.

5.13.2.4 `BGRT_PID_TO_PROC` `#define BGRT_PID_TO_PROC(
p) (p)`

Преобразование идентификатора процесса в указатель на процесс.

Заметки

Может проверять идентификатор процесса на допустимость.

Аргументы

p	Уникальный идентификатор процесса.
---	------------------------------------

Возвращает

Указатель на объект типа `bgrt_proc_t`, или нулевой указатель.

5.13.2.5 `BGRT_PROC_FLG_LOCK` `#define BGRT_PROC_FLG_LOCK ((bgrt_flag_t)0x20)`

Флаг блокировки останова процесса.

В данный момент процесс нельзя останавливать.

5.13.2.6 `BGRT_PROC_FLG_LOCK_MASK` `#define BGRT_PROC_FLG_LOCK_MASK ((bgrt_flag_t)↔(BGRT_PROC_FLG_LOCK))`

Маска `BGRT_PROC_FLG_LOCK`.

Нужна, чтобы определить, удерживает ли процесс общие ресурсы.

5.13.2.7 `BGRT_PROC_FLG_PRE_STOP` `#define BGRT_PROC_FLG_PRE_STOP ((bgrt_flag_t)0x10)`

Флаг запроса останова.

Произошёл запрос на останов процесса. Процесс будет остановлен при первой же возможности.

5.13.2.8 `BGRT_PROC_FLG_RR` `#define BGRT_PROC_FLG_RR ((bgrt_flag_t)0x40)`

Флаг карусельной многозадачности.

Если выставлен этот флаг, то используется карусельная многозадачность, если нет - ФИФО.

5.13.2.9 `BGRT_PROC_FLG_RT` `#define BGRT_PROC_FLG_RT ((bgrt_flag_t)0x80)`

Флаг реального времени.

Для этого процесса используется политика планирования жёсткого реального времени.

5.13.2.10 `BGRT_PROC_GET_STATE` `#define BGRT_PROC_GET_STATE(a) ((a)->flags & BGRT_PROC_STATE_MASK)`

Читает состояние процесса.

Предупреждения

Для внутреннего использования.

5.13.2.11 `BGRT_PROC_LRES_DEC` `#define BGRT_PROC_LRES_DEC(a, b) bgrt_pcounter_dec(&(a)->lres, (bgrt_prio_t)b)`

Макрос-обёртка.

Декремент счётчика `proc->lres`.

Аргументы

a	указатель на процесс.
b	приоритет объекта типа <code>bgrt_sync_t</code> .

```
5.13.2.12 BGRТ_PROC_LRES_INC #define BGRТ_PROC_LRES_INC(
    a,
    b ) bgrt_pcounter_inc(&((a)->lres), (bgrt_prio_t)b)
```

Макрос-обёртка.

Инкремент счётчика `proc->lres`.

Аргументы

a	указатель на процесс.
b	приоритет объекта типа <code>bgrt_sync_t</code> .

```
5.13.2.13 BGRТ_PROC_LRES_INIT #define BGRТ_PROC_LRES_INIT(
    a ) bgrt_pcounter_init(&((a)->lres))
```

Макрос-обёртка.

Иницирует поле `proc->lres` процесса.

Аргументы

a	указатель на процесс.
---	-----------------------

```
5.13.2.14 BGRТ_PROC_PRE_STOP_TEST #define BGRТ_PROC_PRE_STOP_TEST(
    a ) (((a)->flags) & BGRТ_PROC_FLG_PRE_STOP) && !(((a)->flags) & BGRТ_PROC_FLG_LOCK_MASK))
```

Макрос проверки условий останова по флагу `BGRТ_PROC_FLG_PRE_STOP`.

Используется для проверки процессов на возможность останова по флагу `BGRТ_PROC_FLG_PRE_STOP`. Процесс не должен удерживать общие ресурсы в момент останова по флагу.

Предупреждения

Для внутреннего использования.

```
5.13.2.15 BGRТ_PROC_RUN_TEST #define BGRТ_PROC_RUN_TEST(  
    a) (((a)->flags & BGRТ_PROC_STATE_RUN_MASK)>= BGRТ_PROC_STATE_READY)
```

Проверяет, запущен ли процесс.

Предупреждения

Для внутреннего использования.

```
5.13.2.16 BGRТ_PROC_SET_STATE #define BGRТ_PROC_SET_STATE(  
    a,  
    b) ((a)->flags &= BGRТ_PROC_STATE_CLEAR_MASK, (a)->flags |= b)
```

Устанавливает состояние процесса.

Предупреждения

Для внутреннего использования.

```
5.13.2.17 BGRТ_PROC_STATE_CLEAR_MASK #define BGRТ_PROC_STATE_CLEAR_MA←  
SK ((bgrт_flag_t)0xF0)
```

Маска очистки состояния исполнения процесса.

Нужна, чтобы очистить биты состояния выполнения процесса в поле `proc->flags`.

```
5.13.2.18 BGRТ_PROC_STATE_CLEAR_RUN_MASK #define BGRТ_PROC_STATE_CLEAR_R←  
UN_MASK ((bgrт_flag_t)0xFC)
```

Маска очистки состояния исполнения процесса.

Нужна, чтобы очистить младшие биты состояния выполнения процесса в поле `proc->flags`.

```
5.13.2.19 BGRТ_PROC_STATE_DEAD #define BGRТ_PROC_STATE_DEAD ((bgrт_flag_t)0x5)
```

Завершен до завершения `ipc`-транзакций.

```
5.13.2.20 BGRТ_PROC_STATE_END #define BGRТ_PROC_STATE_END ((bgrт_flag_t)0x1)
```

Завершен.

5.13.2.21 `BGRT_PROC_STATE_MASK` `#define BGRT_PROC_STATE_MASK ((bgrt_flag_t)0x0F)`

Маска состояния исполнения процесса.

5.13.2.22 `BGRT_PROC_STATE_PI_DONE` `#define BGRT_PROC_STATE_PI_DONE ((bgrt_flag_t)0xD)`

Запущен при смене приоритета

5.13.2.23 `BGRT_PROC_STATE_PI_PEND` `#define BGRT_PROC_STATE_PI_PEND ((bgrt_flag_t)0xC)`

Ожидает смены приоритета

5.13.2.24 `BGRT_PROC_STATE_PI_READY` `#define BGRT_PROC_STATE_PI_READY ((bgrt_flag_t)0xE)`

Готов к выполнению.

5.13.2.25 `BGRT_PROC_STATE_PI_RUNNING` `#define BGRT_PROC_STATE_PI_RUNNING ((bgrt_flag_t)0xF)`

Выполняется.

5.13.2.26 `BGRT_PROC_STATE_READY` `#define BGRT_PROC_STATE_READY ((bgrt_flag_t)0x2)`

Готов к выполнению.

5.13.2.27 `BGRT_PROC_STATE_RESTART_MASK` `#define BGRT_PROC_STATE_RESTART_MASK ((bgrt_flag_t)0x8)`

Маска проверки состояния процесса.

Используется функцией `bgrt_priv_proc_restart`, для проверки возможности перезапуска.

5.13.2.28 `BGRT_PROC_STATE_RUN_MASK` `#define BGRT_PROC_STATE_RUN_MASK ((bgrt_flag_t)0x3)`

Маска проверки состояния процесса.

Используется для того, чтобы проверить, запущен ли процесс.

5.13.2.29 BGRT_PROC_STATE_RUNNING #define BGRT_PROC_STATE_RUNNING ((bgrt_flag_↔
t)0x3)

Выполняется.

5.13.2.30 BGRT_PROC_STATE_STOPPED #define BGRT_PROC_STATE_STOPPED ((bgrt_flag_t)0x0)

Начальное состояние, остановлен.

5.13.2.31 BGRT_PROC_STATE_SYNC_READY #define BGRT_PROC_STATE_SYNC_REA↔
DY ((bgrt_flag_t)0xA)

Готов к выполнению.

5.13.2.32 BGRT_PROC_STATE_SYNC_RUNNING #define BGRT_PROC_STATE_SYNC_RUNNI↔
NG ((bgrt_flag_t)0xB)

Выполняется.

5.13.2.33 BGRT_PROC_STATE_SYNC_SLEEP #define BGRT_PROC_STATE_SYNC_SLE↔
EP ((bgrt_flag_t)0x9)

Ожидает пробуждения.

5.13.2.34 BGRT_PROC_STATE_SYNC_WAIT #define BGRT_PROC_STATE_SYNC_WAIT ((bgrt_↔
_flag_t)0x8)

Ожидает приема спящих процессов.

5.13.2.35 BGRT_PROC_STATE_TO_READY #define BGRT_PROC_STATE_TO_READY ((bgrt_↔
flag_t)0x6)

Готов к выполнению.

```
5.13.2.36 BGRT_PROC_STATE_TO_RUNNING #define BGRT_PROC_STATE_TO_RUNNI↔
NG ((bgrt_flag_t)0x7)
```

Выполняется.

```
5.13.2.37 BGRT_PROC_STATE_WAIT_MASK #define BGRT_PROC_STATE_WAIT_MASK ((bgrt↔
_flag_t)0x8)
```

Маска проверки состояния процесса.

Используется для того, чтобы проверить, ожидает ли процесс синхронизации.

```
5.13.2.38 BGRT_PROC_STATE_WD_STOPED #define BGRT_PROC_STATE_WD_STOPED ((bgrt↔
_flag_t)0x4)
```

Остановлен по вачдог.

```
5.13.2.39 BGRT_PROC_TO_PID #define BGRT_PROC_TO_PID(
p) (p)
```

Преобразование указателя на процесс в идентификатор процесса.

Заметки

Может проверять допустимость значения указателя.

Аргументы

p	Указатель на объект типа <code>bgrt_proc_t</code> .
---	---

Возвращает

Уникальный идентификатор процесса.

```
5.13.2.40 BGRT_USPD_INIT #define BGRT_USPD_INIT(
proc)
```

Макроопределение:

```
do{\
    proc->udata.scarg = (void *)0;\
    proc->udata.senum = (bgrt_syscall_t)BGRT_SC_ENUM_END;\
}while (0)
```

Инициализация.

5.13.2.41 `BGR_T_USPD_PROC_T` `#define BGR_T_USPD_PROC_T struct bgrt_priv_uspd_t`

Данные процесса из пространства пользователя.

5.13.2.42 `BGR_T_USPD_T` `#define BGR_T_USPD_T BGR_T_USPD_PROC_T *`

Данные процесса из пространства пользователя.

5.13.3 Типы

5.13.3.1 `bgrt_proc_t` `typedef struct bgrt_priv_proc_t bgrt_proc_t`

Смотри `bgrt_priv_proc_t`;

5.13.4 Функции

5.13.4.1 `bgrt_priv_proc_free()` `void bgrt_priv_proc_free (`
`void)`

Останов процесса по флагу `BGR_T_PROC_FLG_PRE_STOP` из критической секции или обработчика прерывания.

Предупреждения

Для внутреннего использования.

5.13.4.2 `bgrt_priv_proc_get_prio()` `bgrt_prio_t bgrt_priv_proc_get_prio (`
`bgrt_proc_t * proc)`

Получить приоритет процесса.

Предупреждения

Для внутреннего использования.

Аргументы

<code>proc</code>	- Указатель на процесс.
-------------------	-------------------------

Возвращает

- Значение приоритета процесса.

```
5.13.4.3 bgrt_priv_proc_init() bgrt_st_t bgrt_priv_proc_init (
    bgrt_proc_t * proc,
    bgrt_code_t pmain,
    bgrt_code_t sv_hook,
    bgrt_code_t rs_hook,
    void * arg,
    bgrt_stack_t * sstart,
    bgrt_prio_t prio,
    bgrt_tmr_t time_quant,
    bgrt_bool_t is_rt,
    bgrt_aff_t affinity )
```

Инициализация процесса из обработчика прерывания, либо из критической секции.

Аргументы

proc	Указатель на иницилируемый процесс.
pmain	Указатель на главную функцию процесса.
sv_hook	Указатель на хук proc->sv_hook.
rs_hook	Указатель на хук proc->rs_hook.
arg	Указатель на аргумент.
sstart	Указатель на дно стека процесса.
prio	Приоритет.
time_quant	Квант времени.
is_rt	Флаг реального времени, если true, значит процесс будет иметь поведение RT.
affinity	Аффинность.

```
5.13.4.4 bgrt_priv_proc_lock() void bgrt_priv_proc_lock (
    void )
```

Установка флага BGRT_PROC_FLG_LOCK для вызывающего процесса.

Предупреждения

Для внутреннего использования.

```
5.13.4.5 bgrt_priv_proc_lres_dec() void bgrt_priv_proc_lres_dec (
    bgrt_proc_t * proc,
    bgrt_prio_t prio )
```

Управление приоритетом процесса.

Декрементирует счётчик `proc->lres`, сбрасывает флаг `BGRT_PROC_FLG_LOCK` при необходимости.

Предупреждения

Для внутреннего использования.

Аргументы

<code>proc</code>	- Указатель на процесс.
<code>prio</code>	- Новое значение приоритета.

```
5.13.4.6 bgrt_priv_proc_lres_inc() void bgrt_priv_proc_lres_inc (
    bgrt_proc_t * proc,
    bgrt_prio_t prio )
```

Управление приоритетом процесса.

Инкрементирует счётчик `proc->lres`, устанавливает флаг `BGRT_PROC_FLG_LOCK`.

Предупреждения

Для внутреннего использования.

Аргументы

<code>proc</code>	- Указатель на процесс.
<code>prio</code>	- Новое значение приоритета.

```
5.13.4.7 bgrt_priv_proc_reset_watchdog() void bgrt_priv_proc_reset_watchdog (
    void )
```

Сброс `watchdog` для процесса реального времени из обработчика прерывания.

Если функцию вызывает процесс реального времени, то функция сбрасывает его таймер. Если процесс завис, и таймер не был вовремя сброшен, то планировщик остановит такой процесс и передаст управление другому.

Предупреждения

Для внутреннего использования.

5.13.4.8 `bgrt_priv_proc_restart()` `bgrt_st_t bgrt_priv_proc_restart (`
`bgrt_proc_t * proc)`

Перезапуск процесса из критической секции или обработчика прерывания.

Если можно (процесс не запущен, завершил работу, не был "убит"), приводит структуру `proc` в состояние, которое было после вызова `bgrt_proc_init`, и ставит процесс в список готовых к выполнению, производит перепланировку.

Аргументы

<code>proc</code>	- Указатель на запускаемый процесс.
-------------------	-------------------------------------

Возвращает

`BGRT_ST_OK` - если процесс был вставлен в список готовых к выполнению, либо код ошибки.

5.13.4.9 `bgrt_priv_proc_run()` `bgrt_st_t bgrt_priv_proc_run (`
`bgrt_proc_t * proc)`

Запуск процесса из критической секции, либо обработчика прерывания.

Ставит процесс в список готовых к выполнению, если можно (процесс не запущен, ещё не завершил работу, не был "убит"), и производит перепланировку.

Аргументы

<code>proc</code>	- Указатель на запускаемый процесс.
-------------------	-------------------------------------

Возвращает

`BGRT_ST_OK` - если процесс был вставлен в список готовых к выполнению, либо код ошибки.

5.13.4.10 `bgrt_priv_proc_self_stop()` `void bgrt_priv_proc_self_stop (`
`void)`

Самоостанов процесса.

Вырезает вызывающий процесс из списка готовых к выполнению и производит перепланировку.

Предупреждения

Для внутреннего использования.

```
5.13.4.11 bgrt_priv_proc_set_prio() void bgrt_priv_proc_set_prio (
    bgrt_proc_t * proc,
    bgrt_prio_t prio )
```

Управление приоритетом процесса.

Устанавливает приоритет процесса, находящегося в любом состоянии.

Предупреждения

Для внутреннего использования.

Аргументы

proc	- Указатель на процесс.
prio	- Новое значение приоритета.

```
5.13.4.12 bgrt_priv_proc_stop() bgrt_st_t bgrt_priv_proc_stop (
    bgrt_proc_t * proc )
```

Останов процесса из критической секции или обработчика прерывания.

Вырезает процесс из списка готовых к выполнению и производит перепланировку.

Аргументы

proc	- Указатель на останавливаемый процесс.
------	---

Возвращает

BGRT_ST_OK - если процесс был вырезан из списка готовых к выполнению, либо код ошибки.

```
5.13.4.13 bgrt_priv_proc_stop_ensure() void bgrt_priv_proc_stop_ensure (
    bgrt_proc_t * proc,
    bgrt_flag_t state )
```

Останов процесса.

Гарантированно останавливает процесс.

Предупреждения

Для внутреннего использования.

Аргументы

proc	- Указатель на процесс.
state	- Новое состояние процесса.

5.13.4.14 `bgrt_priv_proc_terminate()` `void bgrt_priv_proc_terminate (void)`

Завершение работы процесса после возврата из `proc->pmain`.

Предупреждения

Для внутреннего использования.

5.13.4.15 `bgrt_proc_init()` `bgrt_st_t bgrt_proc_init (bgrt_proc_t * proc, bgrt_code_t pmain, bgrt_code_t sv_hook, bgrt_code_t rs_hook, void * arg, bgrt_stack_t * sstart, bgrt_prio_t prio, bgrt_tmr_t time_quant, bgrt_bool_t is_rt, bgrt_aff_t affinity)`

Инициализация процесса.

Аргументы

proc	Указатель на иницилируемый процесс.
pmain	Указатель на главную функцию процесса.
sv_hook	Указатель на хук <code>proc->sv_hook</code> .
rs_hook	Указатель на хук <code>proc->rs_hook</code> .
arg	Указатель на аргумент.
sstart	Указатель на дно стека процесса.
prio	Приоритет.
time_quant	Квант времени.
is_rt	Флаг реального времени, если true, значит процесс будет иметь поведение RT.
affinity	Аффинность.

5.13.4.16 `bgrt_proc_terminate()` `void bgrt_proc_terminate (void)`

Завершение работы процесса после возврата из `proc->pmain`.

5.14 Файл bugurtos/kernel/sched.h

Заголовок планировщика

Структуры данных

- `struct bgrt_priv_sched_t`
Планировщик.
- `struct bgrt_priv_kstat_t`

Макросы

- `#define BGRT_SCHED_PROC_SET_CORE(proc) bgrt_priv_sched_proc_set_core(proc)`

Определения типов

- `typedef struct bgrt_priv_sched_t bgrt_sched_t`
- `typedef struct bgrt_priv_kstat_t bgrt_kstat_t`

Функции

- `void bgrt_sched_init (bgrt_sched_t *sched)`
Инициализация планировщика.
- `bgrt_st_t bgrt_sched_run (bgrt_bool_t is_periodic)`
Функция планирования.
- `void bgrt_sched_proc_run (bgrt_proc_t *proc, bgrt_flag_t state)`
"Низкоуровневый" запуск процесса, для внутреннего использования.
- `void bgrt_sched_proc_stop (bgrt_proc_t *proc, bgrt_flag_t state)`
"Низкоуровневый" останов процесса, для внутреннего использования.
- `bgrt_bool_t bgrt_priv_sched_proc_yield (void)`
Передача управления следующему процессу (для внутреннего использования).
- `bgrt_bool_t bgrt_sched_proc_yield (void)`
Передача управления следующему процессу.
- `bgrt_cpuid_t bgrt_sched_load_balancer (bgrt_proc_t *proc, bgrt_ls_t *stat)`
Балансировщик нагрузки.
- `bgrt_cpuid_t bgrt_sched_highest_load_core (bgrt_ls_t *stat)`
Функция поиска процессорного Ядра с максимальной нагрузкой.
- `void bgrt_priv_sched_proc_set_core (bgrt_proc_t *proc)`
Назначить ядро процессора.
- `void bgrt_sched_lazy_local_load_balancer (void)`
Ленивая балансировка нагрузки, локальный балансировщик.

5.14.1 Подробное описание

Заголовок планировщика

Предупреждения

Все функции в этом файле для внутреннего использования!!!

5.14.2 Макросы

5.14.2.1 `BGRT_SCHED_PROC_SET_CORE` `#define BGRT_SCHED_PROC_SET_CORE(proc) bgrt_priv_sched_proc_set_core(proc)`

5.14.3 Типы

5.14.3.1 `bgrt_kstat_t` `typedef struct bgrt_priv_kstat_t bgrt_kstat_t`

Статистика для балансировки нагрузки, на Hotplug работать не собираемся, все будет статично.

5.14.3.2 `bgrt_sched_t` `typedef struct bgrt_priv_sched_t bgrt_sched_t`

Смотри `bgrt_priv_sched_t`;

5.14.4 Функции

5.14.4.1 `bgrt_priv_sched_proc_set_core()` `void bgrt_priv_sched_proc_set_core (bgrt_proc_t * proc)`

Назначить ядро процессора.

Выбирает и устанавливает ядро, на котором будет выполняться процесс.

Предупреждения

Для внутреннего использования.

Аргументы

proc	Указатель процесс
------	-------------------

5.14.4.2 `bgrt_priv_sched_proc_yield()` `bgrt_bool_t bgrt_priv_sched_proc_yield (void)`

Передача управления следующему процессу (для внутреннего использования).

Передаёт управление следующему процессу, если такой процесс есть.

Предупреждения

Для внутреннего использования.

Возвращает

0 если нет других выполняющихся процессов, не 0 - если есть.

5.14.4.3 `bgrt_sched_highest_load_core()` `bgrt_cpuid_t bgrt_sched_highest_load_core (bgrt_ls_t * stat)`

Функция поиска процессорного Ядра с максимальной нагрузкой.

Используется в глобальном ленивом балансирующем нагрузке и функции `sig_signal`.

Предупреждения

Для внутреннего использования.

Аргументы

<code>stat</code>	Указатель на массив статистики Ядра, либо сигнала.
-------------------	--

Возвращает

ID процессорного ядра с наибольшей нагрузкой.

5.14.4.4 `bgrt_sched_init()` `void bgrt_sched_init (bgrt_sched_t * sched)`

Инициализация планировщика.

Готовит планировщик к запуску.

Предупреждения

Для внутреннего использования.

Аргументы

sched	- Указатель на планировщик.
-------	-----------------------------

5.14.4.5 `bgrt_sched_lazy_local_load_balancer()` `void bgrt_sched_lazy_local_load_balancer (void)`

Ленивая балансировка нагрузки, локальный балансировщик.

Переносит 1 процесс с ядра, на котором выполняется на самое не нагруженное процессорное ядро в системе.

5.14.4.6 `bgrt_sched_load_balancer()` `bgrt_cpuid_t bgrt_sched_load_balancer (bgrt_proc_t * proc, bgrt_ls_t * stat)`

Балансировщик нагрузки.

Используется для балансировки нагрузки в Ядре, а также для предварительной балансировки нагрузки в сигналах.

Предупреждения

Для внутреннего использования.

Аргументы

proc	Указатель на процесс, который надо перенести на новое процессорное ядро.
stat	Указатель на массив статистики Ядра, либо сигнала.

Возвращает

ID процессорного ядра с наименьшей нагрузкой.

5.14.4.7 `bgrt_sched_proc_run()` `void bgrt_sched_proc_run (bgrt_proc_t * proc, bgrt_flag_t state)`

"Низкоуровневый" запуск процесса, для внутреннего использования.

5.14.4.8 `bgrt_sched_proc_stop()` `void bgrt_sched_proc_stop (bgrt_proc_t * proc, bgrt_flag_t state)`

"Низкоуровневый" останов процесса, для внутреннего использования.

5.14.4.9 `bgrt_sched_proc_yield()` `bgrt_bool_t bgrt_sched_proc_yield (void)`

Передача управления следующему процессу.

Передаёт управление следующему процессу, если такой процесс есть.

Возвращает

0 если нет других выполняющихся процессов, не 0 - если есть.

5.14.4.10 `bgrt_sched_run()` `bgrt_st_t bgrt_sched_run (bgrt_bool_t is_periodic)`

Функция планирования.

Предупреждения

Для внутреннего использования.

Аргументы

<code>is_periodic</code>	- Флаг работы по периодическому прерыванию.
--------------------------	---

Возвращает

`BGRT_ST_OK` - если есть новый процесс, `BGRT_ST_EEMPTY` - если нет готовых процессов.

5.15 Файл bugurtos/kernel/sync.h

Заголовок базового примитива синхронизации.

Структуры данных

- `struct bgrt_priv_sync_t`
Базовый примитив синхронизации.

Макросы

- `#define BGRT_SYNC_PRIO(s) bgrt_priv_sync_prio(s)`
Считает приоритет объекта типа `bgrt_sync_t`.
- `#define BGRT_SYNC_INIT(s, p) bgrt_sync_init((bgrt_sync_t *)s, (bgrt_priv_t)p)`
Смотри `bgrt_sync_init`.
- `#define BGRT_PRIV_SYNC_INIT(s, p) bgrt_priv_sync_init((bgrt_sync_t *)s, (bgrt_priv_t)p)`
Смотри `bgrt_priv_sync_init`.

Определения типов

- typedef struct `bgrt_priv_sync_t` `bgrt_sync_t`

Функции

- `bgrt_prio_t bgrt_priv_sync_prio (bgrt_sync_t *sync)`
Возвращает текущий приоритет объекта типа `bgrt_sync_t`.
- `bgrt_st_t bgrt_sync_init (bgrt_sync_t *sync, bgrt_prio_t prio)`
Инициализация базового примитива синхронизации.
- `bgrt_st_t bgrt_priv_sync_init (bgrt_sync_t *sync, bgrt_prio_t prio)`
Инициализация из критической секции, или обработчика прерываний.
- `bgrt_st_t bgrt_priv_sync_set_owner (bgrt_sync_t *sync, bgrt_proc_t *proc)`
Смотри `BGRT_SYNC_SET_OWNER`.
- `bgrt_proc_t * bgrt_priv_sync_get_owner (bgrt_sync_t *sync)`
Смотри `BGRT_SYNC_GET_OWNER`.
- `bgrt_st_t bgrt_priv_sync_own (bgrt_sync_t *sync, bgrt_flag_t touch)`
Смотри `BGRT_SYNC_OWN`.
- `bgrt_st_t bgrt_priv_sync_touch (bgrt_sync_t *sync)`
Смотри `BGRT_SYNC_TOUCH`.
- `bgrt_st_t bgrt_priv_sync_wake (bgrt_sync_t *sync, bgrt_proc_t *proc, bgrt_flag_t chown)`
Смотри `BGRT_SYNC_WAKE`.
- `bgrt_st_t bgrt_priv_sync_sleep (bgrt_sync_t *sync, bgrt_flag_t *touch)`
Смотри `BGRT_SYNC_SLEEP`.
- `bgrt_st_t bgrt_priv_sync_wait (bgrt_sync_t *sync, bgrt_proc_t **proc, bgrt_flag_t block)`
Смотри `BGRT_SYNC_WAIT`.
- `bgrt_st_t bgrt_priv_sync_proc_timeout (bgrt_proc_t *proc)`
Смотри `BGRT_SYNC_PROC_TIMEOUT`.

5.15.1 Подробное описание

Заголовок базового примитива синхронизации.

5.15.2 Макросы

5.15.2.1 `BGRT_PRIV_SYNC_INIT` `#define BGRT_PRIV_SYNC_INIT(
s,
p) bgrt_priv_sync_init((bgrt_sync_t *)s, (bgrt_prio_t)p)`

Смотри `bgrt_priv_sync_init`.

```
5.15.2.2 BGRТ_SYNC_INIT #define BGRТ_SYNC_INIT(
    s,
    p ) bgrт_sync_init((bgrт_sync_t *)s, (bgrт_prio_t)p)
```

Смотри `bgrт_sync_init`.

```
5.15.2.3 BGRТ_SYNC_PRIO #define BGRТ_SYNC_PRIO(
    s ) bgrт_priv_sync_prio(s)
```

Считает приоритет объекта типа `bgrт_sync_t`.

5.15.3 Типы

```
5.15.3.1 bgrт_sync_t typedef struct bgrт_priv_sync_t bgrт_sync_t
```

Смотри `bgrт_priv_sync_t`;

5.15.4 Функции

```
5.15.4.1 bgrт_priv_sync_get_owner() bgrт_proc_t* bgrт_priv_sync_get_owner (
    bgrт_sync_t * sync )
```

Смотри `BGRТ_SYNC_GET_OWNER`.

Предупреждения

Для внутреннего использования.

```
5.15.4.2 bgrт_priv_sync_init() bgrт_st_t bgrт_priv_sync_init (
    bgrт_sync_t * sync,
    bgrт_prio_t prio )
```

Инициализация из критической секции, или обработчика прерываний.

Да, инициировать из обработчика прерывания можно!

Аргументы

<code>sync</code>	Указатель на базовый примитив синхронизации.
<code>prio</code>	Приоритет.

5.15.4.3 `bgrt_priv_sync_own()` `bgrt_st_t bgrt_priv_sync_own (`
`bgrt_sync_t * sync,`
`bgrt_flag_t touch)`

Смотри `BGRT_SYNC_OWN`.

Предупреждения

Для внутреннего использования.

5.15.4.4 `bgrt_priv_sync_prio()` `bgrt_prio_t bgrt_priv_sync_prio (`
`bgrt_sync_t * sync)`

Возвращает текущий приоритет объекта типа `bgrt_sync_t`.

Предупреждения

Для внутреннего использования.

5.15.4.5 `bgrt_priv_sync_proc_timeout()` `bgrt_st_t bgrt_priv_sync_proc_timeout (`
`bgrt_proc_t * proc)`

Смотри `BGRT_SYNC_PROC_TIMEOUT`.

Предупреждения

Для внутреннего использования.

5.15.4.6 `bgrt_priv_sync_set_owner()` `bgrt_st_t bgrt_priv_sync_set_owner (`
`bgrt_sync_t * sync,`
`bgrt_proc_t * proc)`

Смотри `BGRT_SYNC_SET_OWNER`.

Предупреждения

Для внутреннего использования.

5.15.4.7 `bgrt_priv_sync_sleep()` `bgrt_st_t bgrt_priv_sync_sleep (`
`bgrt_sync_t * sync,`
`bgrt_flag_t * touch)`

Смотри `BGRT_SYNC_SLEEP`.

Предупреждения

Для внутреннего использования.

5.15.4.8 `bgrt_priv_sync_touch()` `bgrt_st_t bgrt_priv_sync_touch (`
`bgrt_sync_t * sync)`

Смотри `BGRT_SYNC_TOUCH`.

Предупреждения

Для внутреннего использования.

5.15.4.9 `bgrt_priv_sync_wait()` `bgrt_st_t bgrt_priv_sync_wait (`
`bgrt_sync_t * sync,`
`bgrt_proc_t ** proc,`
`bgrt_flag_t block)`

Смотри `BGRT_SYNC_WAIT`.

Предупреждения

Для внутреннего использования.

5.15.4.10 `bgrt_priv_sync_wake()` `bgrt_st_t bgrt_priv_sync_wake (`
`bgrt_sync_t * sync,`
`bgrt_proc_t * proc,`
`bgrt_flag_t chown)`

Смотри `BGRT_SYNC_WAKE`.

Предупреждения

Для внутреннего использования.

5.15.4.11 `bgrt_sync_init()` `bgrt_st_t bgrt_sync_init (`
`bgrt_sync_t * sync,`
`bgrt_prio_t prio)`

Инициализация базового примитива синхронизации.

Аргументы

sync	Указатель на объект типа <code>bgrt_sync_t</code> .
prio	Приоритет.

5.16 Файл `bugurtos/kernel/syscall.h`

Заголовок системных вызовов.

```
#include <stdarg.h>
#include <syscall_table.h>
#include <default/syscall_api.h>
```

Структуры данных

- `struct _bgrt_va_wr_t`

Макросы

- `#define BGRT_SC_ID(syscall) BGRT_CONCAT(BGRT_SC_ENUM_, syscall)`
Получить идентификатор системного вызова по названию.
- `#define BGRT_SC_TBL_ENTRY(syscall, arg) BGRT_SC_ID(syscall),`
- `#define BGRT_SC_SR_NAME(syscall) BGRT_CONCAT2(BGRT_SC_, BGRT_CONCAT(syscall, _SR))`
Имя обработчика системного вызова.
- `#define BGRT_SC_SR(syscall, arg) bgrt_st_t BGRT_SC_SR_NAME(syscall)(arg)`
Обработчик системного вызова.
- `#define BGRT_SYSCALL_N(sc_name, arg) bgrt_syscall(BGRT_SC_ID(sc_name), arg)`
Системный вызов в виде макроса, см. `bgrt_syscall`.
- `#define BGRT_SYSCALL_NVAR(sc_name, ...) bgrt_syscall_var(BGRT_SC_ID(sc_name), ↵
__VA_ARGS__)`
Системный вызов в виде макроса, см. `bgrt_syscall_var`.

Определения типов

- `typedef enum _bgrt_sc_enum bgrt_sc_enum`
Идентификаторы системных вызовов.
- `typedef bgrt_st_t(* bgrt_scsr_t) (void *)`
Указатель на обработчик системного вызова.
- `typedef struct _bgrt_va_wr_t bgrt_va_wr_t`
Обёртка для `va_list`.

Перечисления

- `enum _bgrt_sc_enum { BGRT_SC_ENUM_END }`

Функции

- `bgrt_st_t bgrt_priv_do_syscall` (`bgrt_syscall_t syscall_num, void *syscall_arg`)
Обработка системного вызова.
- `bgrt_st_t bgrt_syscall_var` (`bgrt_syscall_t num,...`)
Системный вызов.

5.16.1 Подробное описание

Заголовок системных вызовов.

Предупреждения

Все содержимое файла для внутреннего использования!

5.16.2 Макросы

5.16.2.1 `BGRT_SC_ID` `#define BGRT_SC_ID(syscall) BGRT_CONCAT(BGRT_SC_ENUM_, syscall)`

Получить идентификатор системного вызова по названию.

5.16.2.2 `BGRT_SC_SR` `#define BGRT_SC_SR(syscall, arg) bgrt_st_t BGRT_SC_SR_NAME(syscall)(arg)`

Обработчик системного вызова.

5.16.2.3 `BGRT_SC_SR_NAME` `#define BGRT_SC_SR_NAME(syscall) BGRT_CONCAT2(BGRT_SC_, BGRT_CONCAT(syscall, _SR))`

Имя обработчика системного вызова.

5.16.2.4 `BGRT_SC_TBL_ENTRY` `#define BGRT_SC_TBL_ENTRY(syscall, arg) BGRT_SC_ID(syscall),`

5.16.2.5 `BGRT_SYSCALL_N` `#define BGRT_SYSCALL_N(sc_name, arg) bgrt_syscall(BGRT_SC_ID(sc_name), arg)`

Системный вызов в виде макроса, см. `bgrt_syscall`.

Предупреждения

Для внутреннего использования.

Аргументы

sc_name	имя системного вызова (что именно надо выполнить).
arg	Указатель на аргумент.

```
5.16.2.6 BGRТ_SYSCALL_NVAR #define BGRТ_SYSCALL_NVAR(
    sc_name,
    ... ) bgrт_syscall_var(BGRТ_SC_ID(sc_name), __VA_ARGS__)
```

Системный вызов в виде макроса, см. `bgrт_syscall_var`.

Предупреждения

Для внутреннего использования.

Аргументы

sc_name	имя системного вызова (что именно надо выполнить).
---------	--

5.16.3 Типы

```
5.16.3.1 bgrт_sc_enum typedef enum _bgrт_sc_enum bgrт_sc_enum
```

Идентификаторы системных вызовов.

```
5.16.3.2 bgrт_scsr_t typedef bgrт_st_t(* bgrт_scsr_t) (void *)
```

Указатель на обработчик системного вызова.

```
5.16.3.3 bgrт_va_wr_t typedef struct _bgrт_va_wr_t bgrт_va_wr_t
```

Обёртка для `va_list`.

5.16.4 Перечисления

```
5.16.4.1 _bgrт_sc_enum enum _bgrт_sc_enum
```

Элементы перечислений

BGRT_SC_ENUM_END	
------------------	--

5.16.5 Функции

5.16.5.1 `bgrt_priv_do_syscall()` `bgrt_st_t bgrt_priv_do_syscall (`
`bgrt_syscall_t syscall_num,`
`void * syscall_arg)`

Обработка системного вызова.

Запускает обработчик системного вызова и передаёт ему аргумент.

Аргументы

<code>syscall_num</code>	Номер системного вызова.
<code>syscall_arg</code>	Аргумент системного вызова.

Возвращает

Результат выполнения системного вызова.

5.16.5.2 `bgrt_syscall_var()` `bgrt_st_t bgrt_syscall_var (`
`bgrt_syscall_t num,`
`...)`

Системный вызов.

Код Ядра всегда выполняется в контексте Ядра. Это нужно для экономии памяти в стеках процессов. Соответственно, если мы хотим выполнить какие либо операции над процессами, мьютексами, семафорами, сигналами, то нам нужно "попросить" Ядро сделать эту работу.

Именно для этого существует функция `bgrt_syscall`, которая передаёт управление Ядру для выполнения требуемой работы.

Предупреждения

Для внутреннего использования.

Аргументы

<code>num</code>	номер системного вызова (что именно надо выполнить).
------------------	--

5.17 Файл `bugurtos/kernel/timer.h`

Заголовок программных таймеров.

Структуры данных

- `struct bgrt_priv_ktimer_t`

Макросы

- `#define BGRT_CLEAR_TIMER(t) bgrt_priv_clear_timer((bgrt_tmr_t *)&t)`
Сброс программного таймера.
- `#define BGRT_SET_TIMER(t, s) (t += s)`
Установка программного таймера.
- `#define BGRT_TIMER(t) (bgrt_tmr_t)bgrt_priv_timer((bgrt_tmr_t)t)`
Получить значение программного таймера, для внутреннего использования.
- `#define BGRT_WAIT_INTERVAL(tmr, time) (bgrt_priv_wait_interval(&tmr, time))`
Вызывающий процесс ожидает в цикле, когда системное время становится больше или равно, чем `tmr + time`.

Определения типов

- `typedef struct bgrt_priv_ktimer_t bgrt_ktimer_t`

Функции

- `void bgrt_wait_time (bgrt_tmr_t time)`
Подождать заданный интервал времени.
- `void bgrt_priv_clear_timer (bgrt_tmr_t *t)`
Сброс программного таймера.
- `bgrt_tmr_t bgrt_priv_timer (bgrt_tmr_t t)`
Получить значение программного таймера.
- `void bgrt_priv_wait_interval (bgrt_tmr_t *tmr, bgrt_tmr_t time)`
Вызывающий процесс ожидает в цикле, когда системное время становится больше или равно, чем `*tmr + time`.

5.17.1 Подробное описание

Заголовок программных таймеров.

Программные таймеры используются для синхронизации процессов по времени.

Предупреждения

Программные таймеры нельзя использовать для точного измерения интервалов времени!

5.17.2 Макросы

5.17.2.1 `BGRT_CLEAR_TIMER` `#define BGRT_CLEAR_TIMER(
t) bgrt_priv_clear_timer((bgrt_tmr_t *)&t)`

Сброс программного таймера.

Аргументы

t	Имя переменной таймера.
---	-------------------------

5.17.2.2 `BGRT_SET_TIMER` `#define BGRT_SET_TIMER(
t,
s) (t += s)`

Установка программного таймера.

Может быть использована вместо `BGRT_CLEAR_TIMER` для периодического вызова кода, в этом случае поведение таймеров будет детерминированным, не будет наблюдаться рассинхронизации.

Аргументы

t	Имя переменной таймера.
s	Шаг таймера.

5.17.2.3 `BGRT_TIMER` `#define BGRT_TIMER(
t) (bgrt_tmr_t)bgrt_priv_timer((bgrt_tmr_t)t)`

Получить значение программного таймера, для внутреннего использования.

Аргументы

t	Значение таймера.
---	-------------------

5.17.2.4 `BGRT_WAIT_INTERVAL` `#define BGRT_WAIT_INTERVAL(
tmr,
time) (bgrt_priv_wait_interval(&tmr, time))`

Вызывающий процесс ожидает в цикле, когда системное время становится больше или равно, чем `tmr + time`.

Аргументы

tmr	Имя переменной таймера.
time	Интервал времени ожидания.

5.17.3 Типы

5.17.3.1 `bgrt_ktimer_t` typedef struct `bgrt_priv_ktimer_t` `bgrt_ktimer_t`

Системный таймер (используется для подсчёта времени Ядром).

5.17.4 Функции

5.17.4.1 `bgrt_priv_clear_timer()` void `bgrt_priv_clear_timer` (
 `bgrt_tmr_t * t`)

Сброс программного таймера.

Предупреждения

Для внутреннего использования.

Аргументы

t	Указатель на таймер.
---	----------------------

5.17.4.2 `bgrt_priv_timer()` `bgrt_tmr_t` `bgrt_priv_timer` (
 `bgrt_tmr_t t`)

Получить значение программного таймера.

Предупреждения

Для внутреннего использования.

Аргументы

t	Значение таймера.
---	-------------------

5.17.4.3 `bgrt_priv_wait_interval()` void `bgrt_priv_wait_interval` (
 `bgrt_tmr_t * tmr,`
 `bgrt_tmr_t time`)

Вызывающий процесс ожидает в цикле, когда системное время становится больше или равно, чем `*tmr + time`.

Предупреждения

Для внутреннего использования.

Аргументы

<code>tmr</code>	Указатель на таймер.
<code>time</code>	Интервал времени ожидания.

5.17.4.4 `bgrt_wait_time()` `void bgrt_wait_time (`
`bgrt_tmr_t time)`

Подождать заданный интервал времени.

Просто ждёт в цикле пока пройдёт время `time`.

Аргументы

<code>time</code>	Время ожидания.
-------------------	-----------------

5.18 Файл `bugurtos/kernel/vint.h`

Заголовок виртуальных прерываний.

Структуры данных

- `struct bgrt_priv_vint_t`
 Виртуальное прерывание.
- `struct bgrt_priv_vic_t`
 Виртуальный контроллер прерываний.

Макросы

- `#define BGRT_VINT_CS_START() BGRT_INT_LOCK()`
- `#define BGRT_VINT_CS_END() BGRT_INT_FREE()`

Определения типов

- `typedef struct bgrt_priv_vint_t bgrt_vint_t`
- `typedef struct bgrt_priv_vic_t bgrt_vic_t`

Функции

- `void bgrt_vint_init (bgrt_vint_t *vint, bgrt_prio_t prio, bgrt_code_t func, void *arg)`
 Инициализация объект `a` типа `bgrt_vint_t`.
- `bgrt_st_t bgrt_vint_push_isr (bgrt_vint_t *vint, bgrt_vic_t *vic)`
 Поставить объект типа `bgrt_vint_t` на обработку из обработчика прерывания.
- `bgrt_st_t bgrt_vint_push (bgrt_vint_t *vint, bgrt_vic_t *vic)`
 Поставить объект типа `bgrt_vint_t` на обработку.
- `void bgrt_vic_init (bgrt_vic_t *vic)`
 Инициализация виртуального контроллера прерываний.
- `bgrt_st_t bgrt_vic_iterator (bgrt_vic_t *vic)`
 Обработка виртуальных прерываний.
- `void bgrt_vic_do_work (bgrt_vic_t *vic)`
 Обработка виртуальных прерываний.

5.18.1 Подробное описание

Заголовок виртуальных прерываний.

5.18.2 Макросы

5.18.2.1 `BGRT_VINT_CS_END` `#define BGRT_VINT_CS_END() BGRT_INT_FREE()`

5.18.2.2 `BGRT_VINT_CS_START` `#define BGRT_VINT_CS_START() BGRT_INT_LOCK()`

5.18.3 Типы

5.18.3.1 `bgrt_vic_t` `typedef struct bgrt_priv_vic_t bgrt_vic_t`

5.18.3.2 `bgrt_vint_t` `typedef struct bgrt_priv_vint_t bgrt_vint_t`

5.18.4 Функции

5.18.4.1 `bgrt_vic_do_work()` `void bgrt_vic_do_work (`
`bgrt_vic_t * vic)`

Обработка виртуальных прерываний.

Предупреждения

Для внутреннего использования.

Аргументы

<code>vic</code>	Указатель на виртуальный контроллер прерываний.
------------------	---


```
5.18.4.2 bgrt_vic_init() void bgrt_vic_init (  
    bgrt_vic_t * vic )
```

Инициализация виртуального контроллера прерываний.

Предупреждения

Для внутреннего использования.

Аргументы

vic	Указатель на виртуальный контроллер прерываний.
-----	---

```
5.18.4.3 bgrt_vic_iterator() bgrt_st_t bgrt_vic_iterator (  
    bgrt_vic_t * vic )
```

Обработка виртуальных прерываний.

Предупреждения

Для внутреннего использования.

Аргументы

vic	Указатель на виртуальный контроллер прерываний.
-----	---

Возвращает

BGRT_ST_ROLL если нужна ещё итерация, BGRT_ST_OK если вся работа выполнена.

```
5.18.4.4 bgrt_vint_init() void bgrt_vint_init (  
    bgrt_vint_t * vint,  
    bgrt_prio_t prio,  
    bgrt_code_t func,  
    void * arg )
```

Инициализация объект а типа bgrt_vint_t.

Предупреждения

Для внутреннего использования.

Аргументы

vint	Указатель на объект bgrt_vint_t.
prio	Приоритет элемента.
func	Указатель на обработчик.
arg	Указатель на аргумент.

5.18.4.5 `bgrt_vint_push()` `bgrt_st_t bgrt_vint_push (`
`bgrt_vint_t * vint,`
`bgrt_vic_t * vic)`

Поставить объект типа `bgrt_vint_t` на обработку.

Предупреждения

Для внутреннего использования.

Аргументы

vint	Указатель на объект bgrt_vint_t.
vic	Указатель на виртуальный контроллер прерываний.

5.18.4.6 `bgrt_vint_push_isr()` `bgrt_st_t bgrt_vint_push_isr (`
`bgrt_vint_t * vint,`
`bgrt_vic_t * vic)`

Поставить объект типа `bgrt_vint_t` на обработку из обработчика прерывания.

Предупреждения

Для внутреннего использования.

Аргументы

vint	Указатель на объект bgrt_vint_t.
vic	Указатель на виртуальный контроллер прерываний.

5.19 Файл `bugurtos/kernel/xlist.h`

Заголовок списков с приоритетами.

Структуры данных

- `struct bgrt_priv_xlist_t`
Список с приоритетами.

Определения типов

- `typedef struct bgrt_priv_xlist_t bgrt_xlist_t`

Функции

- `void bgrt_xlist_init (bgrt_xlist_t *xlist)`
Инициализация списка.
- `bgrt_item_t * bgrt_xlist_head (bgrt_xlist_t *xlist)`
Поиск головы списка.
- `void bgrt_xlist_switch (bgrt_xlist_t *xlist, bgrt_prio_t prio)`
Переключение списка.

5.19.1 Подробное описание

Заголовок списков с приоритетами.

5.19.2 Типы

5.19.2.1 `bgrt_xlist_t` `typedef struct bgrt_priv_xlist_t bgrt_xlist_t`

Смотри `bgrt_priv_xlist_t`;

5.19.3 Функции

5.19.3.1 `bgrt_xlist_head()` `bgrt_item_t* bgrt_xlist_head (bgrt_xlist_t * xlist)`

Поиск головы списка.

Предупреждения

Для внутреннего использования.

Аргументы

<code>xlist</code>	Указатель на список.
--------------------	----------------------

Возвращает

Указатель на голову - самый приоритетный элемент в массиве указателей.

5.19.3.2 `bgrt_xlist_init()` `void bgrt_xlist_init (`
`bgrt_xlist_t * xlist)`

Инициализация списка.

Предупреждения

Для внутреннего использования.

Аргументы

xlist	Указатель на список.
-------	----------------------

5.19.3.3 `bgrt_xlist_switch()` `void bgrt_xlist_switch (`
`bgrt_xlist_t * xlist,`
`bgrt_prio_t prio)`

Переключение списка.

Изменяет указатель `xlist->item[prio]` на `xlist->item[prio]->next`.

Предупреждения

Для внутреннего использования.

Аргументы

xlist	Указатель на список.
prio	Приоритет переключаемой части списка.

Предметный указатель

- [_bgrt_sc_enum](#)
 - [syscall.h, 86](#)
 - [_bgrt_va_wr_t, 3](#)
 - [list, 3](#)
- [affinity](#)
 - [bgrt_priv_proc_t, 9](#)
- [arg](#)
 - [bgrt_priv_proc_t, 9](#)
 - [bgrt_priv_vint_t, 15](#)
- [atm_cortex_m34_1.h](#)
 - [BGRT_ATM_BCLR_ISR, 16](#)
 - [BGRT_ATM_BGET_ISR, 16](#)
 - [BGRT_ATM_BSET_ISR, 17](#)
 - [BGRT_ATM_INIT_ISR, 17](#)
 - [BGRT_VINT_PUSH_ISR, 17](#)
- [atm_gen_1.h](#)
 - [BGRT_ATM_BCLR_ISR, 17](#)
 - [BGRT_ATM_BGET_ISR, 17](#)
 - [BGRT_ATM_BSET_ISR, 17](#)
 - [BGRT_ATM_INIT_ISR, 17](#)
 - [BGRT_VINT_PUSH_ISR, 18](#)
- [base_prio](#)
 - [bgrt_priv_proc_t, 9](#)
- [BGRT_ASSERT](#)
 - [bugurt.h, 24](#)
- [bgrt_atm_bclr](#)
 - [bugurt_port.h, 20](#)
- [BGRT_ATM_BCLR_ISR](#)
 - [atm_cortex_m34_1.h, 16](#)
 - [atm_gen_1.h, 17](#)
 - [bugurt_port.h, 18](#)
- [bgrt_atm_bget](#)
 - [bugurt_port.h, 21](#)
- [BGRT_ATM_BGET_ISR](#)
 - [atm_cortex_m34_1.h, 16](#)
 - [atm_gen_1.h, 17](#)
 - [bugurt_port.h, 19](#)
- [bgrt_atm_bset](#)
 - [bugurt_port.h, 21](#)
- [BGRT_ATM_BSET_ISR](#)
 - [atm_cortex_m34_1.h, 17](#)
 - [atm_gen_1.h, 17](#)
 - [bugurt_port.h, 19](#)
- [bgrt_atm_init](#)
 - [bugurt_port.h, 22](#)
- [BGRT_ATM_INIT_ISR](#)
 - [atm_cortex_m34_1.h, 17](#)
 - [atm_gen_1.h, 17](#)
 - [bugurt_port.h, 19](#)
- [BGRT_CDECL_BEGIN](#)
 - [bugurt.h, 24](#)
- [BGRT_CDECL_END](#)
 - [bugurt.h, 24](#)
- [BGRT_CLEAR_TIMER](#)
 - [timer.h, 88](#)
- [BGRT_CNT_ADD](#)
 - [pcounter.h, 52](#)
- [bgrt_cnt_add](#)
 - [pcounter.h, 52](#)
- [BGRT_CNT_DEC](#)
 - [pcounter.h, 52](#)
- [bgrt_cnt_dec](#)
 - [pcounter.h, 53](#)
- [BGRT_CNT_INC](#)
 - [pcounter.h, 52](#)
- [bgrt_cnt_inc](#)
 - [pcounter.h, 53](#)
- [BGRT_CNT_SUB](#)
 - [pcounter.h, 52](#)
- [bgrt_cnt_sub](#)
 - [pcounter.h, 53](#)
- [bgrt_code_t](#)
 - [bugurt.h, 27](#)
- [BGRT_CONCAT](#)
 - [bugurt.h, 24](#)
- [BGRT_CONCAT2](#)
 - [bugurt.h, 24](#)
- [BGRT_CONCAT3](#)
 - [bugurt.h, 25](#)
- [BGRT_CRIT_SEC_ENTER](#)
 - [crit_sec.h, 33](#)
- [BGRT_CRIT_SEC_EXIT](#)
 - [crit_sec.h, 33](#)
- [bgrt_curr_cpu](#)
 - [bugurt.h, 27](#)
- [BGRT_CURR_PROC](#)
 - [bugurt_port.h, 20](#)
- [bgrt_curr_proc](#)
 - [bugurt.h, 27](#)
- [BGRT_GET_USPD](#)
 - [proc.h, 62](#)
- [bgrt_init](#)
 - [bugurt.h, 28](#)
- [BGRT_INT_FREE](#)
 - [bugurt_port.h, 20](#)
- [BGRT_INT_LOCK](#)
 - [bugurt_port.h, 20](#)
- [BGRT_ISR](#)
 - [bugurt_port.h, 20](#)
- [bgrt_item_cut](#)
 - [item.h, 47](#)
- [bgrt_item_init](#)
 - [item.h, 47](#)
- [bgrt_item_insert](#)
 - [item.h, 48](#)
- [bgrt_item_t](#)
 - [item.h, 47](#)
- [BGRT_ITEM_T_INIT](#)
 - [item.h, 46](#)

- BGRT_KBLOCK
 - bugurt_port.h, 20
- bgrt_kblock_do_work
 - kernel.h, 49
- bgrt_kblock_init
 - kernel.h, 50
- bgrt_kblock_main
 - kernel.h, 50
- BGRT_KBLOCK_PWRSV
 - kernel.h, 49
- bgrt_kblock_t
 - kernel.h, 49
- BGRT_KBLOCK_VRESCH
 - kernel.h, 49
- BGRT_KBLOCK_VSCALL
 - kernel.h, 49
- BGRT_KBLOCK_VSCHMSK
 - kernel.h, 49
- BGRT_KBLOCK_VTMR
 - kernel.h, 49
- bgrt_kernel
 - kernel.h, 50
- bgrt_kernel_init
 - kernel.h, 50
- BGRT_KERNEL_PREEMPT
 - bugurt.h, 25
- bgrt_kernel_t
 - kernel.h, 49
- bgrt_kstat_t
 - sched.h, 76
- bgrt_ktimer_t
 - timer.h, 89
- bgrt_map_search
 - index.h, 45
- bgrt_pcounter_dec
 - pcounter.h, 54
- bgrt_pcounter_inc
 - pcounter.h, 54
- bgrt_pcounter_init
 - pcounter.h, 55
- bgrt_pcounter_minus
 - pcounter.h, 55
- bgrt_pcounter_plus
 - pcounter.h, 55
- bgrt_pcounter_t
 - pcounter.h, 52
- BGRT_PID_NOTHING
 - proc.h, 62
- BGRT_PID_T
 - proc.h, 62
- BGRT_PID_TO_PROC
 - proc.h, 62
- bgrt_pitem_cut
 - pitem.h, 57
- bgrt_pitem_fast_cut
 - pitem.h, 58
- bgrt_pitem_init
 - pitem.h, 58
- bgrt_pitem_insert
 - pitem.h, 58
- bgrt_pitem_t
 - pitem.h, 57
- BGRT_PITEM_T_INIT
 - pitem.h, 57
- bgrt_pitem_xlist_chain
 - pitem.h, 59
- BGRT_PRIO_LOWEST
 - bugurt.h, 25
- bgrt_priv_clear_timer
 - timer.h, 90
- bgrt_priv_crit_sec_enter
 - crit_sec.h, 34
- bgrt_priv_crit_sec_exit
 - crit_sec.h, 34
- bgrt_priv_do_syscall
 - syscall.h, 87
- bgrt_priv_item_t, 3
 - next, 4
 - prev, 4
- bgrt_priv_kblock_t, 4
 - hpmap, 4
 - lpmap, 4
- bgrt_priv_kernel_t, 5
 - kblock, 5
 - sched, 5
 - stat, 5
 - timer, 5
- bgrt_priv_kstat_t, 5
 - lock, 6
 - val, 6
- bgrt_priv_ktimer_t, 6
 - lock, 6
 - tick, 6
 - val, 6
- bgrt_priv_pcounter_t, 7
 - counter, 7
 - map, 7
- bgrt_priv_pitem_t, 7
 - list, 8
 - parent, 8
 - prio, 8
- bgrt_priv_proc_free
 - proc.h, 69
- bgrt_priv_proc_get_prio
 - proc.h, 69
- bgrt_priv_proc_init
 - proc.h, 70
- bgrt_priv_proc_lock
 - proc.h, 70
- bgrt_priv_proc_lres_dec
 - proc.h, 70
- bgrt_priv_proc_lres_inc
 - proc.h, 71
- bgrt_priv_proc_reset_watchdog
 - proc.h, 71
- bgrt_priv_proc_restart

- proc.h, 71
- bgrt_priv_proc_run
 - proc.h, 72
- bgrt_priv_proc_self_stop
 - proc.h, 72
- bgrt_priv_proc_set_prio
 - proc.h, 72
- bgrt_priv_proc_stop
 - proc.h, 73
- bgrt_priv_proc_stop_ensure
 - proc.h, 73
- bgrt_priv_proc_t, 8
 - affinity, 9
 - arg, 9
 - base_prio, 9
 - cnt_lock, 9
 - core_id, 9
 - flags, 9
 - lock, 9
 - lres, 10
 - parent, 10
 - pmain, 10
 - rs_hook, 10
 - spointer, 10
 - sstart, 10
 - sv_hook, 10
 - sync, 10
 - time_quant, 10
 - timer, 10
 - udata, 10
- bgrt_priv_proc_terminate
 - proc.h, 74
- bgrt_priv_sched_proc_set_core
 - sched.h, 76
- bgrt_priv_sched_proc_yield
 - sched.h, 77
- bgrt_priv_sched_t, 11
 - current_proc, 11
 - expired, 11
 - lock, 11
 - nested_crit_sec, 11
 - plst, 12
 - ready, 12
- bgrt_priv_sync_get_owner
 - sync.h, 81
- BGRT_PRIV_SYNC_INIT
 - sync.h, 80
- bgrt_priv_sync_init
 - sync.h, 81
- bgrt_priv_sync_own
 - sync.h, 82
- bgrt_priv_sync_prio
 - sync.h, 82
- bgrt_priv_sync_proc_timeout
 - sync.h, 82
- bgrt_priv_sync_set_owner
 - sync.h, 82
- bgrt_priv_sync_sleep
 - sync.h, 82
- bgrt_priv_sync_t, 12
 - dirty, 12
 - lock, 13
 - owner, 13
 - prio, 13
 - pwake, 13
 - sleep, 13
 - snum, 13
- bgrt_priv_sync_touch
 - sync.h, 83
- bgrt_priv_sync_wait
 - sync.h, 83
- bgrt_priv_sync_wake
 - sync.h, 83
- bgrt_priv_timer
 - timer.h, 90
- bgrt_priv_uspd_t, 13
 - scarg, 13
 - senum, 14
 - secret, 14
- bgrt_priv_vic_t, 14
 - list, 14
 - prio, 14
- bgrt_priv_vint_t, 15
 - arg, 15
 - func, 15
 - parent, 15
- bgrt_priv_wait_interval
 - timer.h, 90
- bgrt_priv_xlist_t, 15
 - item, 16
 - map, 16
- BGRT_PROC_FLG_LOCK
 - proc.h, 63
- BGRT_PROC_FLG_LOCK_MASK
 - proc.h, 63
- BGRT_PROC_FLG_PRE_STOP
 - proc.h, 63
- BGRT_PROC_FLG_RR
 - proc.h, 63
- BGRT_PROC_FLG_RT
 - proc.h, 63
- BGRT_PROC_FREE
 - syscall_api.h, 36
- BGRT_PROC_GET_ID
 - syscall_api.h, 36
- BGRT_PROC_GET_PRIO
 - syscall_api.h, 36
- BGRT_PROC_GET_STATE
 - proc.h, 63
- bgrt_proc_init
 - proc.h, 74
- BGRT_PROC_LOCK
 - syscall_api.h, 37
- BGRT_PROC_LRES_DEC
 - proc.h, 63
- BGRT_PROC_LRES_INC

- proc.h, 64
- BGRT_PROC_LRES_INIT
 - proc.h, 64
- BGRT_PROC_PRE_STOP_TEST
 - proc.h, 64
- BGRT_PROC_RESET_WATCHDOG
 - syscall_api.h, 37
- BGRT_PROC_RESTART
 - syscall_api.h, 37
- BGRT_PROC_RUN
 - syscall_api.h, 37
- BGRT_PROC_RUN_TEST
 - proc.h, 64
- BGRT_PROC_SELF_STOP
 - syscall_api.h, 38
- BGRT_PROC_SET_PRIO
 - syscall_api.h, 38
- BGRT_PROC_SET_STATE
 - proc.h, 65
- bgrt_proc_stack_init
 - bugurt.h, 28
- BGRT_PROC_STATE_CLEAR_MASK
 - proc.h, 65
- BGRT_PROC_STATE_CLEAR_RUN_MASK
 - proc.h, 65
- BGRT_PROC_STATE_DEAD
 - proc.h, 65
- BGRT_PROC_STATE_END
 - proc.h, 65
- BGRT_PROC_STATE_MASK
 - proc.h, 65
- BGRT_PROC_STATE_PI_DONE
 - proc.h, 66
- BGRT_PROC_STATE_PI_PEND
 - proc.h, 66
- BGRT_PROC_STATE_PI_READY
 - proc.h, 66
- BGRT_PROC_STATE_PI_RUNNING
 - proc.h, 66
- BGRT_PROC_STATE_READY
 - proc.h, 66
- BGRT_PROC_STATE_RESTART_MASK
 - proc.h, 66
- BGRT_PROC_STATE_RUN_MASK
 - proc.h, 66
- BGRT_PROC_STATE_RUNNING
 - proc.h, 66
- BGRT_PROC_STATE_STOPPED
 - proc.h, 67
- BGRT_PROC_STATE_SYNC_READY
 - proc.h, 67
- BGRT_PROC_STATE_SYNC_RUNNING
 - proc.h, 67
- BGRT_PROC_STATE_SYNC_SLEEP
 - proc.h, 67
- BGRT_PROC_STATE_SYNC_WAIT
 - proc.h, 67
- BGRT_PROC_STATE_TO_READY
 - proc.h, 67
- BGRT_PROC_STATE_TO_RUNNING
 - proc.h, 67
- BGRT_PROC_STATE_WAIT_MASK
 - proc.h, 68
- BGRT_PROC_STATE_WD_STOPPED
 - proc.h, 68
- BGRT_PROC_STOP
 - syscall_api.h, 38
- bgrt_proc_t
 - proc.h, 69
- bgrt_proc_terminate
 - proc.h, 74
- BGRT_PROC_TO_PID
 - proc.h, 68
- bgrt_resched
 - bugurt.h, 28
- BGRT_RESCHED_PROC
 - bugurt.h, 25
- bgrt_sc_enum
 - syscall.h, 86
- BGRT_SC_ENUM_END
 - syscall.h, 87
- BGRT_SC_ID
 - syscall.h, 85
- BGRT_SC_SR
 - syscall.h, 85
- syscall_routines.h, 43–45
- BGRT_SC_SR_NAME
 - syscall.h, 85
- BGRT_SC_TBL_ENTRY
 - syscall.h, 85
- bgrt_sched_highest_load_core
 - sched.h, 77
- bgrt_sched_init
 - sched.h, 77
- bgrt_sched_lazy_local_load_balancer
 - sched.h, 78
- bgrt_sched_load_balancer
 - sched.h, 78
- bgrt_sched_proc_run
 - sched.h, 78
- BGRT_SCHED_PROC_SET_CORE
 - sched.h, 76
- bgrt_sched_proc_stop
 - sched.h, 78
- bgrt_sched_proc_yield
 - sched.h, 78
- bgrt_sched_run
 - sched.h, 79
- bgrt_sched_t
 - sched.h, 76
- bgrt_scsr_t
 - syscall.h, 86
- BGRT_SET_TIMER
 - timer.h, 89
- BGRT_SPIN_FREE
 - bugurt.h, 25

- bgrt_spin_free
 - bugurt.h, 29
- BGRT_SPIN_INIT
 - bugurt.h, 25
- bgrt_spin_init
 - bugurt.h, 29
- BGRT_SPIN_LOCK
 - bugurt.h, 25
- bgrt_spin_lock
 - bugurt.h, 30
- BGRT_ST_EAGAIN
 - bugurt.h, 25
- BGRT_ST_EEMPTY
 - bugurt.h, 26
- BGRT_ST_ENULL
 - bugurt.h, 26
- BGRT_ST_EOWN
 - bugurt.h, 26
- BGRT_ST_ESTAT
 - bugurt.h, 26
- BGRT_ST_ESYNC
 - bugurt.h, 26
- BGRT_ST_ETIMEOUT
 - bugurt.h, 26
- BGRT_ST_IDLE
 - bugurt.h, 26
- BGRT_ST_OK
 - bugurt.h, 26
- BGRT_ST_ROLL
 - bugurt.h, 27
- BGRT_ST_SCALL
 - bugurt.h, 27
- bgrt_start
 - bugurt.h, 30
- bgrt_stat_calc_load
 - bugurt.h, 30
- bgrt_stat_dec
 - bugurt.h, 30
- bgrt_stat_inc
 - bugurt.h, 31
- bgrt_stat_init
 - bugurt.h, 31
- bgrt_stat_merge
 - bugurt.h, 32
- bgrt_switch_to_proc
 - bugurt.h, 32
- BGRT_SYNC_GET_OWNER
 - syscall_api.h, 39
- BGRT_SYNC_INIT
 - sync.h, 80
- bgrt_sync_init
 - sync.h, 83
- BGRT_SYNC_OWN
 - syscall_api.h, 39
- BGRT_SYNC_PRIO
 - sync.h, 81
- BGRT_SYNC_PROC_TIMEOUT
 - syscall_api.h, 39
- BGRT_SYNC_SET_OWNER
 - syscall_api.h, 40
- BGRT_SYNC_SLEEP
 - syscall_api.h, 40
- bgrt_sync_t
 - sync.h, 81
- BGRT_SYNC_TOUCH
 - syscall_api.h, 40
- BGRT_SYNC_WAIT
 - syscall_api.h, 41
- BGRT_SYNC_WAKE
 - syscall_api.h, 41
- bgrt_syscall
 - bugurt.h, 32
- BGRT_SYSCALL_N
 - syscall.h, 85
- BGRT_SYSCALL_NVAR
 - syscall.h, 86
- bgrt_syscall_var
 - syscall.h, 87
- BGRT_TIMER
 - timer.h, 89
- bgrt_user_func_t
 - syscall_routines.h, 42
- BGRT_USPD_INIT
 - proc.h, 68
- BGRT_USPD_PROC_T
 - proc.h, 68
- BGRT_USPD_T
 - proc.h, 69
- bgrt_va_wr_t
 - syscall.h, 86
- bgrt_vic_do_work
 - vint.h, 92
- bgrt_vic_init
 - vint.h, 92
- bgrt_vic_iterator
 - vint.h, 93
- bgrt_vic_t
 - vint.h, 92
- BGRT_VINT_CS_END
 - vint.h, 92
- BGRT_VINT_CS_START
 - vint.h, 92
- bgrt_vint_init
 - vint.h, 93
- bgrt_vint_push
 - vint.h, 94
- BGRT_VINT_PUSH_ISR
 - atm_cortex_m34_1.h, 17
 - atm_gen_1.h, 18
- bgrt_vint_push_isr
 - vint.h, 94
- bgrt_vint_t
 - vint.h, 92
- BGRT_WAIT_INTERVAL
 - timer.h, 89
- bgrt_wait_time

- timer.h, 91
- bgrt_xlist_head
 - xlist.h, 95
- bgrt_xlist_init
 - xlist.h, 96
- bgrt_xlist_switch
 - xlist.h, 96
- bgrt_xlist_t
 - xlist.h, 95
- bugurt.h
 - BGRT_ASSERT, 24
 - BGRT_CDECL_BEGIN, 24
 - BGRT_CDECL_END, 24
 - bgrt_code_t, 27
 - BGRT_CONCAT, 24
 - BGRT_CONCAT2, 24
 - BGRT_CONCAT3, 25
 - bgrt_curr_cpu, 27
 - bgrt_curr_proc, 27
 - bgrt_init, 28
 - BGRT_KERNEL_PREEMPT, 25
 - BGRT_PRIO_LOWEST, 25
 - bgrt_proc_stack_init, 28
 - bgrt_resched, 28
 - BGRT_RESCHED_PROC, 25
 - BGRT_SPIN_FREE, 25
 - bgrt_spin_free, 29
 - BGRT_SPIN_INIT, 25
 - bgrt_spin_init, 29
 - BGRT_SPIN_LOCK, 25
 - bgrt_spin_lock, 30
 - BGRT_ST_EAGAIN, 25
 - BGRT_ST_EEMPTY, 26
 - BGRT_ST_ENULL, 26
 - BGRT_ST_EOWN, 26
 - BGRT_ST_ESTAT, 26
 - BGRT_ST_ESYNC, 26
 - BGRT_ST_ETIMEOUT, 26
 - BGRT_ST_IDLE, 26
 - BGRT_ST_OK, 26
 - BGRT_ST_ROLL, 27
 - BGRT_ST_SCALL, 27
 - bgrt_start, 30
 - bgrt_stat_calc_load, 30
 - bgrt_stat_dec, 30
 - bgrt_stat_inc, 31
 - bgrt_stat_init, 31
 - bgrt_stat_merge, 32
 - bgrt_switch_to_proc, 32
 - bgrt_syscall, 32
- bugurt_port.h
 - bgrt_atm_bclr, 20
 - BGRT_ATM_BCLR_ISR, 18
 - bgrt_atm_bget, 21
 - BGRT_ATM_BGET_ISR, 19
 - bgrt_atm_bset, 21
 - BGRT_ATM_BSET_ISR, 19
 - bgrt_atm_init, 22
 - BGRT_ATM_INIT_ISR, 19
 - BGRT_CURR_PROC, 20
 - BGRT_INT_FREE, 20
 - BGRT_INT_LOCK, 20
 - BGRT_ISR, 20
 - BGRT_KBLOCK, 20
- bugurtos/arch/common/atm_cortex_m34_1.h, 16
- bugurtos/arch/common/atm_gen_1.h, 17
- bugurtos/doc/doxygen/bugurt_port.h, 18
- bugurtos/kernel/bugurt.h, 22
- bugurtos/kernel/crit_sec.h, 33
- bugurtos/kernel/default/syscall_api.h, 34
- bugurtos/kernel/default/syscall_routines.h, 42
- bugurtos/kernel/index.h, 45
- bugurtos/kernel/item.h, 46
- bugurtos/kernel/kernel.h, 48
- bugurtos/kernel/pcounter.h, 51
- bugurtos/kernel/pitem.h, 56
- bugurtos/kernel/proc.h, 59
- bugurtos/kernel/sched.h, 75
- bugurtos/kernel/sync.h, 79
- bugurtos/kernel/syscall.h, 84
- bugurtos/kernel/timer.h, 88
- bugurtos/kernel/vint.h, 91
- bugurtos/kernel/xlist.h, 94
- cnt_lock
 - bgrt_priv_proc_t, 9
- core_id
 - bgrt_priv_proc_t, 9
- counter
 - bgrt_priv_pcounter_t, 7
- crit_sec.h
 - BGRT_CRIT_SEC_ENTER, 33
 - BGRT_CRIT_SEC_EXIT, 33
 - bgrt_priv_crit_sec_enter, 34
 - bgrt_priv_crit_sec_exit, 34
- current_proc
 - bgrt_priv_sched_t, 11
- dirty
 - bgrt_priv_sync_t, 12
- expired
 - bgrt_priv_sched_t, 11
- flags
 - bgrt_priv_proc_t, 9
- func
 - bgrt_priv_vint_t, 15
- hpmmap
 - bgrt_priv_kblock_t, 4
- index.h
 - bgrt_map_search, 45
- item
 - bgrt_priv_xlist_t, 16
- item.h

- bgrt_item_cut, 47
- bgrt_item_init, 47
- bgrt_item_insert, 48
- bgrt_item_t, 47
- BGRT_ITEM_T_INIT, 46
- kblock
 - bgrt_priv_kernel_t, 5
- kernel.h
 - bgrt_kblock_do_work, 49
 - bgrt_kblock_init, 50
 - bgrt_kblock_main, 50
 - BGRT_KBLOCK_PWRSV, 49
 - bgrt_kblock_t, 49
 - BGRT_KBLOCK_VRESCH, 49
 - BGRT_KBLOCK_VSCALL, 49
 - BGRT_KBLOCK_VSCHMSK, 49
 - BGRT_KBLOCK_VTMR, 49
 - bgrt_kernel, 50
 - bgrt_kernel_init, 50
 - bgrt_kernel_t, 49
- list
 - _bgrt_va_wr_t, 3
 - bgrt_priv_pitem_t, 8
 - bgrt_priv_vic_t, 14
- lock
 - bgrt_priv_kstat_t, 6
 - bgrt_priv_ktimer_t, 6
 - bgrt_priv_proc_t, 9
 - bgrt_priv_sched_t, 11
 - bgrt_priv_sync_t, 13
- lpmap
 - bgrt_priv_kblock_t, 4
- lres
 - bgrt_priv_proc_t, 10
- map
 - bgrt_priv_pcounter_t, 7
 - bgrt_priv_xlist_t, 16
- nested_crit_sec
 - bgrt_priv_sched_t, 11
- next
 - bgrt_priv_item_t, 4
- owner
 - bgrt_priv_sync_t, 13
- parent
 - bgrt_priv_pitem_t, 8
 - bgrt_priv_proc_t, 10
 - bgrt_priv_vint_t, 15
- pcounter.h
 - BGRT_CNT_ADD, 52
 - bgrt_cnt_add, 52
 - BGRT_CNT_DEC, 52
 - bgrt_cnt_dec, 53
 - BGRT_CNT_INC, 52
 - bgrt_cnt_inc, 53
 - BGRT_CNT_SUB, 52
 - bgrt_cnt_sub, 53
 - bgrt_pcounter_dec, 54
 - bgrt_pcounter_inc, 54
 - bgrt_pcounter_init, 55
 - bgrt_pcounter_minus, 55
 - bgrt_pcounter_plus, 55
 - bgrt_pcounter_t, 52
- pitem.h
 - bgrt_pitem_cut, 57
 - bgrt_pitem_fast_cut, 58
 - bgrt_pitem_init, 58
 - bgrt_pitem_insert, 58
 - bgrt_pitem_t, 57
 - BGRT_PITEM_T_INIT, 57
 - bgrt_pitem_xlist_chain, 59
- plst
 - bgrt_priv_sched_t, 12
- pmain
 - bgrt_priv_proc_t, 10
- prev
 - bgrt_priv_item_t, 4
- prio
 - bgrt_priv_pitem_t, 8
 - bgrt_priv_sync_t, 13
 - bgrt_priv_vic_t, 14
- proc.h
 - BGRT_GET_USPD, 62
 - BGRT_PID_NOTHING, 62
 - BGRT_PID_T, 62
 - BGRT_PID_TO_PROC, 62
 - bgrt_priv_proc_free, 69
 - bgrt_priv_proc_get_prio, 69
 - bgrt_priv_proc_init, 70
 - bgrt_priv_proc_lock, 70
 - bgrt_priv_proc_lres_dec, 70
 - bgrt_priv_proc_lres_inc, 71
 - bgrt_priv_proc_reset_watchdog, 71
 - bgrt_priv_proc_restart, 71
 - bgrt_priv_proc_run, 72
 - bgrt_priv_proc_self_stop, 72
 - bgrt_priv_proc_set_prio, 72
 - bgrt_priv_proc_stop, 73
 - bgrt_priv_proc_stop_ensure, 73
 - bgrt_priv_proc_terminate, 74
 - BGRT_PROC_FLG_LOCK, 63
 - BGRT_PROC_FLG_LOCK_MASK, 63
 - BGRT_PROC_FLG_PRE_STOP, 63
 - BGRT_PROC_FLG_RR, 63
 - BGRT_PROC_FLG_RT, 63
 - BGRT_PROC_GET_STATE, 63
 - bgrt_proc_init, 74
 - BGRT_PROC_LRES_DEC, 63
 - BGRT_PROC_LRES_INC, 64
 - BGRT_PROC_LRES_INIT, 64
 - BGRT_PROC_PRE_STOP_TEST, 64
 - BGRT_PROC_RUN_TEST, 64
 - BGRT_PROC_SET_STATE, 65

- BGRT_PROC_STATE_CLEAR_MASK, 65
- BGRT_PROC_STATE_CLEAR_RUN_MASK, 65
- BGRT_PROC_STATE_DEAD, 65
- BGRT_PROC_STATE_END, 65
- BGRT_PROC_STATE_MASK, 65
- BGRT_PROC_STATE_PI_DONE, 66
- BGRT_PROC_STATE_PI_PEND, 66
- BGRT_PROC_STATE_PI_READY, 66
- BGRT_PROC_STATE_PI_RUNNING, 66
- BGRT_PROC_STATE_READY, 66
- BGRT_PROC_STATE_RESTART_MASK, 66
- BGRT_PROC_STATE_RUN_MASK, 66
- BGRT_PROC_STATE_RUNNING, 66
- BGRT_PROC_STATE_STOPED, 67
- BGRT_PROC_STATE_SYNC_READY, 67
- BGRT_PROC_STATE_SYNC_RUNNING, 67
- BGRT_PROC_STATE_SYNC_SLEEP, 67
- BGRT_PROC_STATE_SYNC_WAIT, 67
- BGRT_PROC_STATE_TO_READY, 67
- BGRT_PROC_STATE_TO_RUNNING, 67
- BGRT_PROC_STATE_WAIT_MASK, 68
- BGRT_PROC_STATE_WD_STOPED, 68
- bgrt_proc_t, 69
- bgrt_proc_terminate, 74
- BGRT_PROC_TO_PID, 68
- BGRT_USPD_INIT, 68
- BGRT_USPD_PROC_T, 68
- BGRT_USPD_T, 69
- pwake
 - bgrt_priv_sync_t, 13
- ready
 - bgrt_priv_sched_t, 12
- rs_hook
 - bgrt_priv_proc_t, 10
- scarg
 - bgrt_priv_uspd_t, 13
- sched
 - bgrt_priv_kernel_t, 5
- sched.h
 - bgrt_kstat_t, 76
 - bgrt_priv_sched_proc_set_core, 76
 - bgrt_priv_sched_proc_yield, 77
 - bgrt_sched_highest_load_core, 77
 - bgrt_sched_init, 77
 - bgrt_sched_lazy_local_load_balancer, 78
 - bgrt_sched_load_balancer, 78
 - bgrt_sched_proc_run, 78
 - BGRT_SCHED_PROC_SET_CORE, 76
 - bgrt_sched_proc_stop, 78
 - bgrt_sched_proc_yield, 78
 - bgrt_sched_run, 79
 - bgrt_sched_t, 76
- scnum
 - bgrt_priv_uspd_t, 14
- secret
- bgrt_priv_uspd_t, 14
- sleep
 - bgrt_priv_sync_t, 13
- snum
 - bgrt_priv_sync_t, 13
- spointer
 - bgrt_priv_proc_t, 10
- sstart
 - bgrt_priv_proc_t, 10
- stat
 - bgrt_priv_kernel_t, 5
- sv_hook
 - bgrt_priv_proc_t, 10
- sync
 - bgrt_priv_proc_t, 10
- sync.h
 - bgrt_priv_sync_get_owner, 81
 - BGRT_PRIV_SYNC_INIT, 80
 - bgrt_priv_sync_init, 81
 - bgrt_priv_sync_own, 82
 - bgrt_priv_sync_prio, 82
 - bgrt_priv_sync_proc_timeout, 82
 - bgrt_priv_sync_set_owner, 82
 - bgrt_priv_sync_sleep, 82
 - bgrt_priv_sync_touch, 83
 - bgrt_priv_sync_wait, 83
 - bgrt_priv_sync_wake, 83
 - BGRT_SYNC_INIT, 80
 - bgrt_sync_init, 83
 - BGRT_SYNC_PRIO, 81
 - bgrt_sync_t, 81
- syscall.h
 - _bgrt_sc_enum, 86
 - bgrt_priv_do_syscall, 87
 - bgrt_sc_enum, 86
 - BGRT_SC_ENUM_END, 87
 - BGRT_SC_ID, 85
 - BGRT_SC_SR, 85
 - BGRT_SC_SR_NAME, 85
 - BGRT_SC_TBL_ENTRY, 85
 - bgrt_scsr_t, 86
 - BGRT_SYSCALL_N, 85
 - BGRT_SYSCALL_NVAR, 86
 - bgrt_syscall_var, 87
 - bgrt_va_wr_t, 86
- syscall_api.h
 - BGRT_PROC_FREE, 36
 - BGRT_PROC_GET_ID, 36
 - BGRT_PROC_GET_PRIO, 36
 - BGRT_PROC_LOCK, 37
 - BGRT_PROC_RESET_WATCHDOG, 37
 - BGRT_PROC_RESTART, 37
 - BGRT_PROC_RUN, 37
 - BGRT_PROC_SELF_STOP, 38
 - BGRT_PROC_SET_PRIO, 38
 - BGRT_PROC_STOP, 38
 - BGRT_SYNC_GET_OWNER, 39

BGRT_SYNC_OWN, 39
BGRT_SYNC_PROC_TIMEOUT, 39
BGRT_SYNC_SET_OWNER, 40
BGRT_SYNC_SLEEP, 40
BGRT_SYNC_TOUCH, 40
BGRT_SYNC_WAIT, 41
BGRT_SYNC_WAKE, 41
syscall_routines.h
 BGRT_SC_SR, 43–45
 bgrt_user_func_t, 42
tick
 bgrt_priv_ktimer_t, 6
time_quant
 bgrt_priv_proc_t, 10
timer
 bgrt_priv_kernel_t, 5
 bgrt_priv_proc_t, 10
timer.h
 BGRT_CLEAR_TIMER, 88
 bgrt_ktimer_t, 89
 bgrt_priv_clear_timer, 90
 bgrt_priv_timer, 90
 bgrt_priv_wait_interval, 90
 BGRT_SET_TIMER, 89
 BGRT_TIMER, 89
 BGRT_WAIT_INTERVAL, 89
 bgrt_wait_time, 91
udata
 bgrt_priv_proc_t, 10
val
 bgrt_priv_kstat_t, 6
 bgrt_priv_ktimer_t, 6
vint.h
 bgrt_vic_do_work, 92
 bgrt_vic_init, 92
 bgrt_vic_iterator, 93
 bgrt_vic_t, 92
 BGRT_VINT_CS_END, 92
 BGRT_VINT_CS_START, 92
 bgrt_vint_init, 93
 bgrt_vint_push, 94
 bgrt_vint_push_isr, 94
 bgrt_vint_t, 92
xlist.h
 bgrt_xlist_head, 95
 bgrt_xlist_init, 96
 bgrt_xlist_switch, 96
 bgrt_xlist_t, 95