

BuguRTOS

4.1.0

Generated by Doxygen 1.8.17

---

<b>1 Main Page</b>	<b>1</b>
<b>2 Data Structure Index</b>	<b>1</b>
2.1 Data Structures	1
<b>3 File Index</b>	<b>2</b>
3.1 File List	2
<b>4 Data Structure Documentation</b>	<b>3</b>
4.1 <code>_bgrt_va_wr_t</code> Struct Reference	3
4.1.1 Field Documentation	3
4.2 <code>bgrt_priv_item_t</code> Struct Reference	3
4.2.1 Detailed Description	3
4.2.2 Field Documentation	4
4.3 <code>bgrt_priv_kblock_t</code> Struct Reference	4
4.3.1 Detailed Description	4
4.3.2 Field Documentation	4
4.4 <code>bgrt_priv_kernel_t</code> Struct Reference	5
4.4.1 Detailed Description	5
4.4.2 Field Documentation	5
4.5 <code>bgrt_priv_kstat_t</code> Struct Reference	6
4.5.1 Field Documentation	6
4.6 <code>bgrt_priv_ktimer_t</code> Struct Reference	6
4.6.1 Field Documentation	7
4.7 <code>bgrt_priv_pcounter_t</code> Struct Reference	7
4.7.1 Detailed Description	7
4.7.2 Field Documentation	7
4.8 <code>bgrt_priv_pitem_t</code> Struct Reference	8
4.8.1 Detailed Description	8
4.8.2 Field Documentation	8
4.9 <code>bgrt_priv_proc_t</code> Struct Reference	9
4.9.1 Detailed Description	9
4.9.2 Field Documentation	9
4.10 <code>bgrt_priv_sched_t</code> Struct Reference	11
4.10.1 Detailed Description	11
4.10.2 Field Documentation	12
4.11 <code>bgrt_priv_sync_t</code> Struct Reference	12
4.11.1 Detailed Description	13
4.11.2 Field Documentation	13
4.12 <code>bgrt_priv_uspd_t</code> Struct Reference	14
4.12.1 Field Documentation	14
4.13 <code>bgrt_priv_vic_t</code> Struct Reference	14
4.13.1 Detailed Description	14

---

4.13.2 Field Documentation . . . . .	15
4.14 bgrt_priv_vint_t Struct Reference . . . . .	15
4.14.1 Detailed Description . . . . .	15
4.14.2 Field Documentation . . . . .	15
4.15 bgrt_priv_xlist_t Struct Reference . . . . .	16
4.15.1 Detailed Description . . . . .	16
4.15.2 Field Documentation . . . . .	16
<b>5 File Documentation</b> . . . . .	<b>17</b>
5.1 bugurtos/arch/common/atm_cortex_m34_1.h File Reference . . . . .	17
5.1.1 Macro Definition Documentation . . . . .	17
5.2 bugurtos/arch/common/atm_gen_1.h File Reference . . . . .	18
5.2.1 Macro Definition Documentation . . . . .	18
5.3 bugurtos/doc/doxygen/bugurt_port.h File Reference . . . . .	19
5.3.1 Macro Definition Documentation . . . . .	19
5.3.2 Function Documentation . . . . .	21
5.4 bugurtos/kernel/bugurt.h File Reference . . . . .	23
5.4.1 Detailed Description . . . . .	25
5.4.2 Macro Definition Documentation . . . . .	25
5.4.3 Typedef Documentation . . . . .	28
5.4.4 Function Documentation . . . . .	28
5.5 bugurtos/kernel/crit_sec.h File Reference . . . . .	33
5.5.1 Detailed Description . . . . .	34
5.5.2 Macro Definition Documentation . . . . .	34
5.5.3 Function Documentation . . . . .	34
5.6 bugurtos/kernel/default/syscall_api.h File Reference . . . . .	35
5.6.1 Detailed Description . . . . .	36
5.6.2 Macro Definition Documentation . . . . .	36
5.7 bugurtos/kernel/default/syscall_routines.h File Reference . . . . .	42
5.7.1 Typedef Documentation . . . . .	43
5.7.2 Function Documentation . . . . .	43
5.8 bugurtos/kernel/index.h File Reference . . . . .	45
5.8.1 Detailed Description . . . . .	45
5.8.2 Function Documentation . . . . .	45
5.9 bugurtos/kernel/item.h File Reference . . . . .	46
5.9.1 Detailed Description . . . . .	46
5.9.2 Macro Definition Documentation . . . . .	46
5.9.3 Typedef Documentation . . . . .	47
5.9.4 Function Documentation . . . . .	47
5.10 bugurtos/kernel/kernel.h File Reference . . . . .	48
5.10.1 Detailed Description . . . . .	49
5.10.2 Macro Definition Documentation . . . . .	49

---

5.10.3 Typedef Documentation . . . . .	49
5.10.4 Function Documentation . . . . .	49
5.10.5 Variable Documentation . . . . .	50
5.11 bugurtos/kernel/pcounter.h File Reference . . . . .	51
5.11.1 Detailed Description . . . . .	51
5.11.2 Macro Definition Documentation . . . . .	52
5.11.3 Typedef Documentation . . . . .	52
5.11.4 Function Documentation . . . . .	52
5.12 bugurtos/kernel/pitem.h File Reference . . . . .	56
5.12.1 Detailed Description . . . . .	57
5.12.2 Macro Definition Documentation . . . . .	57
5.12.3 Typedef Documentation . . . . .	57
5.12.4 Function Documentation . . . . .	57
5.13 bugurtos/kernel/proc.h File Reference . . . . .	59
5.13.1 Detailed Description . . . . .	62
5.13.2 Macro Definition Documentation . . . . .	62
5.13.3 Typedef Documentation . . . . .	69
5.13.4 Function Documentation . . . . .	69
5.14 bugurtos/kernel/sched.h File Reference . . . . .	74
5.14.1 Detailed Description . . . . .	75
5.14.2 Macro Definition Documentation . . . . .	75
5.14.3 Typedef Documentation . . . . .	76
5.14.4 Function Documentation . . . . .	76
5.15 bugurtos/kernel/sync.h File Reference . . . . .	79
5.15.1 Detailed Description . . . . .	80
5.15.2 Macro Definition Documentation . . . . .	80
5.15.3 Typedef Documentation . . . . .	80
5.15.4 Function Documentation . . . . .	80
5.16 bugurtos/kernel/syscall.h File Reference . . . . .	83
5.16.1 Detailed Description . . . . .	84
5.16.2 Macro Definition Documentation . . . . .	84
5.16.3 Typedef Documentation . . . . .	85
5.16.4 Enumeration Type Documentation . . . . .	85
5.16.5 Function Documentation . . . . .	86
5.17 bugurtos/kernel/timer.h File Reference . . . . .	87
5.17.1 Detailed Description . . . . .	87
5.17.2 Macro Definition Documentation . . . . .	87
5.17.3 Typedef Documentation . . . . .	88
5.17.4 Function Documentation . . . . .	89
5.18 bugurtos/kernel/vint.h File Reference . . . . .	90
5.18.1 Detailed Description . . . . .	91
5.18.2 Macro Definition Documentation . . . . .	91

5.18.3 Typedef Documentation . . . . .	91
5.18.4 Function Documentation . . . . .	91
5.19 bugurtos/kernel/xlist.h File Reference . . . . .	94
5.19.1 Detailed Description . . . . .	94
5.19.2 Typedef Documentation . . . . .	94
5.19.3 Function Documentation . . . . .	94
<b>Index</b>	<b>97</b>

## 1 Main Page

The BuguRTOS is a RTOS bgrt\_kernel. It is written by anonymous JUST FOR FUN.

### Warning

BuguRTOS license is modified GPLv3, look at exception.txt for more info.

## 2 Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_bgrt_va_wr_t</a>	<b>3</b>
<a href="#">bgrt_priv_item_t</a> A list item	<b>3</b>
<a href="#">bgrt_priv_kblock_t</a> A BuguRTOS kernel block structure	<b>4</b>
<a href="#">bgrt_priv_kernel_t</a> A BuguRTOS kernel structure	<b>5</b>
<a href="#">bgrt_priv_kstat_t</a>	<b>6</b>
<a href="#">bgrt_priv_ktimer_t</a>	<b>6</b>
<a href="#">bgrt_priv_pcounter_t</a> A locked resource counter	<b>7</b>
<a href="#">bgrt_priv_pitem_t</a> A prioritized list item	<b>8</b>
<a href="#">bgrt_priv_proc_t</a> A process	<b>9</b>
<a href="#">bgrt_priv_sched_t</a> A scheduler	<b>11</b>
<a href="#">bgrt_priv_sync_t</a> Basic synchronization primitive	<b>12</b>

<a href="#">bgrt_priv_uspd_t</a>	14
<a href="#">bgrt_priv_vic_t</a> A virtual interrupt controller	14
<a href="#">bgrt_priv_vint_t</a> A virtual interrupt	15
<a href="#">bgrt_priv_xlist_t</a> A prioritized list	16

## 3 File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">bugurtos/arch/common/atm_cortex_m34_1.h</a>	17
<a href="#">bugurtos/arch/common/atm_gen_1.h</a>	18
<a href="#">bugurtos/doc/doxygen/bugurt_port.h</a>	19
<a href="#">bugurtos/kernel/bugurt.h</a> The top header file	23
<a href="#">bugurtos/kernel/crit_sec.h</a> A critical section header	33
<a href="#">bugurtos/kernel/index.h</a> An map search header	45
<a href="#">bugurtos/kernel/item.h</a> A list item header	46
<a href="#">bugurtos/kernel/kernel.h</a> A kernel header	48
<a href="#">bugurtos/kernel/pcounter.h</a> A locked resource counter header	51
<a href="#">bugurtos/kernel/pitem.h</a> A prioritized list item header	56
<a href="#">bugurtos/kernel/proc.h</a> A process header	59
<a href="#">bugurtos/kernel/sched.h</a> A scheduler header	74
<a href="#">bugurtos/kernel/sync.h</a> A sync header	79
<a href="#">bugurtos/kernel/syscall.h</a> System call header	83
<a href="#">bugurtos/kernel/timer.h</a> A software timer headers	87

<a href="#">bugurtos/kernel/vint.h</a> A virtual interrupt header	90
<a href="#">bugurtos/kernel/xlist.h</a> A prioritized list header	94
<a href="#">bugurtos/kernel/default/syscall_api.h</a> System call header	35
<a href="#">bugurtos/kernel/default/syscall_routines.h</a>	42

## 4 Data Structure Documentation

### 4.1 [\\_bgrt\\_va\\_wr\\_t](#) Struct Reference

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

#### Data Fields

- [va\\_list list](#)

#### 4.1.1 Field Documentation

**4.1.1.1 list** [va\\_list \\_bgrt\\_va\\_wr\\_t::list](#)

The documentation for this struct was generated from the following file:

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

### 4.2 [bgrt\\_priv\\_item\\_t](#) Struct Reference

A list item.

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

#### Data Fields

- [bgrt\\_item\\_t \\* next](#)
- [bgrt\\_item\\_t \\* prev](#)

#### 4.2.1 Detailed Description

A list item.

All structures, that must be listed, will inherit [bgrt\\_item\\_t](#) properties and methods.

## 4.2.2 Field Documentation

### 4.2.2.1 `next` `bgrt_item_t*` `bgrt_priv_item_t::next`

Next item in a list.

### 4.2.2.2 `prev` `bgrt_item_t*` `bgrt_priv_item_t::prev`

Previous item in a list.

The documentation for this struct was generated from the following file:

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

## 4.3 `bgrt_priv_kblock_t` Struct Reference

A BuguRTOS kernel block structure.

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

### Data Fields

- `bgrt_map_t` [hpmmap](#)
- `bgrt_map_t` [lpmap](#)

### 4.3.1 Detailed Description

A BuguRTOS kernel block structure.

A kernel block is responsible for virtual interrupt processing, system call processing and process scheduling in certain CPU core.

### 4.3.2 Field Documentation

#### 4.3.2.1 `hpmmap` `bgrt_map_t` `bgrt_priv_kblock_t::hpmmap`

A high priority fast virtual interrupt controller.



#### 4.3.2.2 lpmap `bgrt_map_t bgrt_priv_kblock_t::lpmap`

A low priority fast virtual interrupt controller.

The documentation for this struct was generated from the following file:

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

## 4.4 bgrt\_priv\_kernel\_t Struct Reference

A BuguRTOS kernel structure.

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

### Data Fields

- [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 Detailed Description

A BuguRTOS kernel structure.

The kernel stores information about launched processes, system time and other important information.

### 4.4.2 Field Documentation

#### 4.4.2.1 kblock `bgrt_kblock_t bgrt_priv_kernel_t::kblock` [BGRT\_MAX\_CPU]

Software interrupt controllers.

#### 4.4.2.2 sched `bgrt_sched_t bgrt_priv_kernel_t::sched` [BGRT\_MAX\_CPU]

A separate scheduler for every CPU core.

#### 4.4.2.3 stat `bgrt_kstat_t bgrt_priv_kernel_t::stat`

A statistic for load balancing, CPU hotplug is not supported.

#### 4.4.2.4 timer `bgrt_ktimer_t bgrt_priv_kernel_t::timer`

The system timer.

The documentation for this struct was generated from the following file:

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

## 4.5 `bgrt_priv_kstat_t` Struct Reference

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

### Data Fields

- `bgrt_ls_t val` [BGRT\_MAX\_CPU]
- `bgrt_lock_t lock`

### 4.5.1 Field Documentation

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

A spin-lock.

#### 4.5.1.2 val `bgrt_ls_t bgrt_priv_kstat_t::val` [BGRT\_MAX\_CPU]

A values.

The documentation for this struct was generated from the following file:

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

## 4.6 `bgrt_priv_ktimer_t` Struct Reference

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

### Data Fields

- `void(* tick)(void)`
- `bgrt_tmr_t val`
- `bgrt_lock_t lock`

### 4.6.1 Field Documentation

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

A spin-lock.

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

A hook pointer.

**4.6.1.3 val** `bgrt_tmr_t bgrt_priv_ktimer_t::val`

A value.

The documentation for this struct was generated from the following file:

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

## 4.7 bgrt\_priv\_pcounter\_t Struct Reference

A locked resource counter.

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

### Data Fields

- `bgrt_cnt_t counter` [BGRT\_BITS\_IN\_INDEX\_T]
- `bgrt_map_t map`

### 4.7.1 Detailed Description

A locked resource counter.

`bgrt_pcounter_t` objects are used to store information about inherited priorities.

### 4.7.2 Field Documentation

**4.7.2.1 counter** `bgrt_cnt_t bgrt_priv_pcounter_t::counter` [BGRT\_BITS\_IN\_INDEX\_T]

A counter array.

#### 4.7.2.2 **map** `bgrt_map_t bgrt_priv_pcounter_t::map`

An map to speedup search.

The documentation for this struct was generated from the following file:

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

## 4.8 **bgrt\_priv\_pitem\_t** Struct Reference

A prioritized list item.

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

### Data Fields

- [bgrt\\_item\\_t parent](#)
- [bgrt\\_xlist\\_t \\* list](#)
- [bgrt\\_prio\\_t prio](#)

#### 4.8.1 Detailed Description

A prioritized list item.

#### 4.8.2 Field Documentation

##### 4.8.2.1 **list** `bgrt_xlist_t* bgrt_priv_pitem_t::list`

A pointer to an `bgrt_xlist_t` object.

##### 4.8.2.2 **parent** `bgrt_item_t bgrt_priv_pitem_t::parent`

A parent - `bgrt_item_t`.

##### 4.8.2.3 **prio** `bgrt_prio_t bgrt_priv_pitem_t::prio`

A priority.

The documentation for this struct was generated from the following file:

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

## 4.9 bgrt\_priv\_proc\_t Struct Reference

A process.

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

### Data Fields

- [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 Detailed Description

A process.

There are many OSES, so It may be called a process, a thread, a task etc. The point of all these names is: independent sequence of CPU instructions.

So a process is a part of your program, that has its own "main" routine (stored in pmain field of bgrt\_proc\_t object). A process "main" routine can be written in a way as if there were no other processes!

It's possible to use one "main" routine for many processes, as different processes are independent, but you have to remember one thing about static variables in such "main" routine.

#### Warning

Be careful with static variables, these variables are common for all processes sharing one routine! You must access such static variables using process synchronization facilities.

### 4.9.2 Field Documentation

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

An Affinity of a process.

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

An argument for `pmain`, `sv_hook`, `rs_hook`, may be used to store process local data.

**4.9.2.3 base\_prio** `bgrt_prio_t bgrt_priv_proc_t::base_prio`

A base process priority.

**4.9.2.4 cnt\_lock** `bgrt_cnt_t bgrt_priv_proc_t::cnt_lock`

A counter of `BGRT_PROC_LOCK` nesting.

**4.9.2.5 core\_id** `bgrt_cpuid_t bgrt_priv_proc_t::core_id`

An ID of a CPU that runs a process.

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

Process state flags (to treat process state quickly).

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

A process spin-lock.

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

A locked resource counter.

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

A parent is `bgrt_pitem_t`.

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

A pointer to a process "main" routine.

**4.9.2.11 rs\_hook** `bgrt_code_t bgrt_priv_proc_t::rs_hook`

A context restore hook, it is run before restoring a process context.

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

A process stack top pointer.

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

A process stack bottom pointer.

**4.9.2.14 sv\_hook** `bgrt_code_t bgrt_priv_proc_t::sv_hook`

A context save hook, it is run after saving a process context.

**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`

A process time slice.

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

A process timer, it is used as watchdog for real time processes

**4.9.2.18 udata** `BGRT_USPD_PROC_T bgrt_priv_proc_t::udata`

User space process data.

The documentation for this struct was generated from the following file:

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

## 4.10 bgrt\_priv\_sched\_t Struct Reference

A scheduler.

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

### Data Fields

- `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 Detailed Description

A scheduler.

A scheduler object contains an information about processes, running on some CPU core.

## 4.10.2 Field Documentation

**4.10.2.1 current\_proc** `bgrt_proc_t* bgrt_priv_sched_t::current_proc`

A currently running process.

**4.10.2.2 expired** `bgrt_xlist_t* bgrt_priv_sched_t::expired`

A pointer to an expired process list.

**4.10.2.3 lock** `bgrt_lock_t bgrt_priv_sched_t::lock`

A scheduler spin-lock.

**4.10.2.4 nested\_crit\_sec** `bgrt_cnt_t bgrt_priv_sched_t::nested_crit_sec`

A critical section nesting count.

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

A storage for a ready and for an expired process lists.

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

A pointer to a ready process list.

The documentation for this struct was generated from the following file:

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

## 4.11 bgrt\_priv\_sync\_t Struct Reference

Basic synchronization primitive.

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

### Data Fields

- [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 Detailed Description

Basic synchronization primitive.

A basic type that handles blocking process synchronization. By wrapping this type one can get traditional synchronization primitives (mutexes, semaphores, conditional variables, message-FIFOs, IPC-endpoints, etc.).

Basic priority inheritance protocol is supported.

### 4.11.2 Field Documentation

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

Dirty priority inheritance transaction counter.

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

A sync spin-lock.

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

A pointer to a process, that holds a sync.

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

Priority.

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

Pending wakeup counter.

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

A list of waiting processes.

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

Sleeping process counter.

The documentation for this struct was generated from the following file:

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

## 4.12 bgrt\_priv\_uspd\_t Struct Reference

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

### Data Fields

- void \* [scarg](#)
- bgrt\_syscall\_t [scnum](#)
- bgrt\_st\_t [scret](#)

### 4.12.1 Field Documentation

**4.12.1.1 scarg** void\* bgrt\_priv\_uspd\_t::scarg

A system call pointer.

**4.12.1.2 scnum** bgrt\_syscall\_t bgrt\_priv\_uspd\_t::scnum

A system call number.

**4.12.1.3 scret** bgrt\_st\_t bgrt\_priv\_uspd\_t::scret

A system call result.

The documentation for this struct was generated from the following file:

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

## 4.13 bgrt\_priv\_vic\_t Struct Reference

A virtual interrupt controller.

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

### Data Fields

- [bgrt\\_xlist\\_t](#) list
- [bgrt\\_prio\\_t](#) prio

### 4.13.1 Detailed Description

A virtual interrupt controller.

### 4.13.2 Field Documentation

**4.13.2.1 list** [bgrt\\_xlist\\_t](#) `bgrt_priv_vic_t::list`

A parent - [bgrt\\_xlist\\_t](#).

**4.13.2.2 prio** [bgrt\\_prio\\_t](#) `bgrt_priv_vic_t::prio`

Current priority.

The documentation for this struct was generated from the following file:

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

## 4.14 bgrt\_priv\_vint\_t Struct Reference

A virtual interrupt.

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

### Data Fields

- [bgrt\\_pitem\\_t](#) `parent`
- [bgrt\\_code\\_t](#) `func`
- `void * arg`

### 4.14.1 Detailed Description

A virtual interrupt.

### 4.14.2 Field Documentation

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

A virtual ISR arg.

**4.14.2.2 func** [bgrt\\_code\\_t](#) `bgrt_priv_vint_t::func`

A virtual ISR pointer.

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

A parent - `bgrt_item_t`.

The documentation for this struct was generated from the following file:

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

### 4.15 `bgrt_priv_xlist_t` Struct Reference

A prioritized list.

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

#### Data Fields

- `bgrt_item_t * item` [BGRT\_BITS\_IN\_INDEX\_T]
- `bgrt_map_t map`

#### 4.15.1 Detailed Description

A prioritized list.

A container type, `bgrt_xlist_t` objects store lists of `bgrt_item_t` objects. In fact these containers store lists of `bgrt_pitem_t` or other compatible objects.

#### 4.15.2 Field Documentation

##### 4.15.2.1 `item` `bgrt_item_t*` `bgrt_priv_xlist_t::item`[BGRT\_BITS\_IN\_INDEX\_T]

An array of list head pointers.

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

Index for fast search.

The documentation for this struct was generated from the following file:

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

## 5 File Documentation

### 5.1 bugurtos/arch/common/atm\_cortex\_m34\_1.h File Reference

#### Macros

- #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 Macro Definition Documentation

**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 bugurtos/arch/common/atm\_gen\_1.h File Reference

### Macros

- #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 Macro Definition Documentation

**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 bugurtos/doc/doxygen/bugurt\_port.h File Reference

### Macros

- #define [BGRT\\_INT\\_LOCK\(\)](#)  
*Disable interrupts.*
- #define [BGRT\\_INT\\_FREE\(\)](#)  
*Enable interrupts.*
- #define [BGRT\\_KBLOCK](#)  
*Current kernel block.*
- #define [BGRT\\_CURR\\_PROC](#)  
*Current process.*
- #define [BGRT\\_ISR\(v\)](#)  
*Interrupt service routine declaration template.*
- #define [BGRT\\_ATM\\_INIT\\_ISR\(map\\_ptr\)](#)  
*Atomic map initialization.*
- #define [BGRT\\_ATM\\_BSET\\_ISR\(map\\_ptr, msk\)](#)  
*Set masked bits.*
- #define [BGRT\\_ATM\\_BGET\\_ISR\(map\\_ptr, msk\)](#)  
*Read masked bits.*
- #define [BGRT\\_ATM\\_BCLR\\_ISR\(map\\_ptr, msk\)](#)  
*Clear masked bits.*

### Functions

- void [bgrt\\_atm\\_init](#) (bgrt\_map\_t \*map\_ptr)  
*Atomic map initialization.*
- void [bgrt\\_atm\\_bset](#) (bgrt\_map\_t \*map\_ptr, bgrt\_map\_t msk)  
*Set bits using mask.*
- bgrt\_map\_t [bgrt\\_atm\\_bget](#) (bgrt\_map\_t \*map\_ptr, bgrt\_map\_t msk)  
*Read masked bits.*
- bgrt\_map\_t [bgrt\\_atm\\_bclr](#) (bgrt\_map\_t \*map\_ptr, bgrt\_map\_t msk)  
*Clear masked bits.*

### 5.3.1 Macro Definition Documentation

**5.3.1.1 BGRT\_ATM\_BCLR\_ISR** #define BGRT\_ATM\_BCLR\_ISR(  
    map\_ptr,  
    msk )

Clear masked bits.

#### Warning

For ISR/crit\_sec usage!

**Parameters**

<i>map_ptr</i>	A pointer to atomic map.
<i>msk</i>	A mask.

**Returns**

Last masked bits state.

**5.3.1.2 BGRT\_ATM\_BGET\_ISR** `#define BGRT_ATM_BGET_ISR(  
    map_ptr,  
    msk )`

Read masked bits.

**Warning**

For ISR/crit\_sec usage!

**Parameters**

<i>map_ptr</i>	A pointer to atomic map.
<i>msk</i>	A mask.

**Returns**

Masked vectors state.

**5.3.1.3 BGRT\_ATM\_BSET\_ISR** `#define BGRT_ATM_BSET_ISR(  
    map_ptr,  
    msk )`

Set masked bits.

**Parameters**

<i>map_ptr</i>	A pointer to atomic map.
<i>msk</i>	A mask.

**5.3.1.4 BGRT\_ATM\_INIT\_ISR** `#define BGRT_ATM_INIT_ISR(  
    map_ptr )`

Atomic map initialization.



### Warning

For ISR/crit\_sec usage!

### Parameters

<i>map_ptr</i>	A pointer to atomic map.
----------------	--------------------------

#### 5.3.1.5 **BGRT\_CURR\_PROC** `#define BGRT_CURR_PROC`

Current process.

#### 5.3.1.6 **BGRT\_INT\_FREE** `#define BGRT_INT_FREE( )`

Enable interrupts.

#### 5.3.1.7 **BGRT\_INT\_LOCK** `#define BGRT_INT_LOCK( )`

Disable interrupts.

#### 5.3.1.8 **BGRT\_ISR** `#define BGRT_ISR( v )`

Interrupt service routine declaration template.

### Parameters

<i>v</i>	An interrupt vector id.
----------	-------------------------

#### 5.3.1.9 **BGRT\_KBLOCK** `#define BGRT_KBLOCK`

Current kernel block.

## 5.3.2 Function Documentation

**5.3.2.1 bgrt\_atm\_bclr()** `bgrt_map_t bgrt_atm_bclr (`  
    `bgrt_map_t * map_ptr,`  
    `bgrt_map_t msk )`

Clear masked bits.

**Parameters**

<i>map_ptr</i>	A pointer to atomic map.
<i>msk</i>	A mask.

**Returns**

Last masked bits state.

**5.3.2.2 bgrt\_atm\_bget()** `bgrt_map_t bgrt_atm_bget (`  
    `bgrt_map_t * map_ptr,`  
    `bgrt_map_t msk )`

Read masked bits.

**Parameters**

<i>map_ptr</i>	A pointer to atomic map.
<i>msk</i>	A mask.

**Returns**

Masked vectors state.

**5.3.2.3 bgrt\_atm\_bset()** `void bgrt_atm_bset (`  
    `bgrt_map_t * map_ptr,`  
    `bgrt_map_t msk )`

Set bits using mask.

**Warning**

For ISR/crit\_sec usage!

**Parameters**

<i>map_ptr</i>	A pointer to atomic map.
<i>msk</i>	A mask.

**5.3.2.4 bgrt\_atm\_init()** void bgrt\_atm\_init (  
     bgrt\_map\_t \* map\_ptr )

Atomic map initialization.

Parameters

<i>map_ptr</i>	A pointer to atomic map.
----------------	--------------------------

## 5.4 bugurtos/kernel/bugurt.h File Reference

The top header file.

```
#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"
```

### Macros

- #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)  
     *Success.*
- #define [BGRT\\_ST\\_ENULL](#) ((bgrt\_st\_t)1)  
     *Null pointer argument.*
- #define [BGRT\\_ST\\_EOWN](#) ((bgrt\_st\_t)2)  
     *Ownership error.*
- #define [BGRT\\_ST\\_EEMPTY](#) ((bgrt\_st\_t)3)  
     *The process list is empty.*
- #define [BGRT\\_ST\\_ESYNC](#) ((bgrt\_st\_t)4)  
     *Wrong bgrt\_sync\_t object.*
- #define [BGRT\\_ST\\_ETIMEOUT](#) ((bgrt\_st\_t)5)

- *Timeout expired.*
- #define `BGRT_ST_ESTAT` ((bgrt\_st\_t)6)
- *Process state error.*
- #define `BGRT_ST_EAGAIN` ((bgrt\_st\_t)7)
- *Try again.*
- #define `BGRT_ST_SCALL` ((bgrt\_st\_t)8)
- *Wrong system call.*
- #define `BGRT_ST_ROLL` ((bgrt\_st\_t)9)
- *Next iteration needed.*
- #define `BGRT_ST_IDLE` ((bgrt\_st\_t)10)
- *The system is IDLE.*
- #define `BGRT_PRIO_LOWEST` ((bgrt\_prio\_t)BGRT\_BITS\_IN\_INDEX\_T - (bgrt\_prio\_t)1)
- *Lowest priority level.*
- #define `BGRT_SPIN_INIT`(arg) `bgrt_spin_init(&((arg)->lock))`
- *Wrapper macro.*
- #define `BGRT_SPIN_LOCK`(arg) `bgrt_spin_lock(&((arg)->lock))`
- *Wrapper macro.*
- #define `BGRT_SPIN_FREE`(arg) `bgrt_spin_free(&((arg)->lock))`
- *Wrapper macro.*
- #define `BGRT_RESCHED_PROC`(proc) `bgrt_resched(proc->core_id)`
- *Wrapper macro.*
- #define `BGRT_KERNEL_PREEMPT`() `bgrt_kblock_do_work(&bgrt_kernel.kblock[bgrt_curr_cpu()])`

## Typedefs

- typedef void(\* `bgrt_code_t`) (void \*)
- *Executable code.*

## Functions

- void `bgrt_spin_init` (bgrt\_lock\_t \*lock)
- *Spin-lock initialization for MP system.*
- void `bgrt_spin_lock` (bgrt\_lock\_t \*lock)
- *Lock spin-lock on MP system.*
- void `bgrt_spin_free` (bgrt\_lock\_t \*lock)
- *Unlock spin-lock on MP system.*
- bgrt\_cpuid\_t `bgrt_curr_cpu` (void)
- *Returns processor core id.*
- void `bgrt_stat_init` (bgrt\_ls\_t \*stat)
- *Statistic initialization.*
- void `bgrt_stat_inc` (bgrt\_proc\_t \*proc, bgrt\_ls\_t \*stat)
- *Statistic update on load increase.*
- void `bgrt_stat_dec` (bgrt\_proc\_t \*proc, bgrt\_ls\_t \*stat)
- *Statistic update on load decrease.*
- void `bgrt_stat_merge` (bgrt\_ls\_t \*src\_stat, bgrt\_ls\_t \*dst\_stat)
- *Statistic merge.*
- bgrt\_load\_t `bgrt_stat_calc_load` (bgrt\_prio\_t prio, bgrt\_ls\_t \*stat)
- void `bgrt_resched` (bgrt\_cpuid\_t core\_id)
- *Rescheduling.*
- `bgrt_proc_t * bgrt_curr_proc` (void)

*Current process.*

- `bgrt_stack_t * bgrt_proc_stack_init (bgrt_stack_t *sstart, bgrt_code_t pmain, void *arg, void(*return_↔ address)(void))`

*A process stack initialization.*

- `void bgrt_init (void)`

*The Kernel initiation.*

- `void bgrt_start (void)`

*The OS start.*

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

*A system call.*

- `void bgrt_switch_to_proc (void)`

*Kernel to process context switch.*

### 5.4.1 Detailed Description

The top header file.

All other BuguRTOS headers are included here. On the other hand all BuguRTOS source files include this file.

### 5.4.2 Macro Definition Documentation

**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\_INDEX\_T -  
(bgrt\_prio\_t)1)

Lowest priority level.

**5.4.2.9 BGRT\_RESCHED\_PROC** #define BGRT\_RESCHED\_PROC(  
    *proc* ) bgrt\_resched(*proc*->core\_id)

Wrapper macro.

A wrapper for bgrt\_resched function.

**5.4.2.10 BGRT\_SPIN\_FREE** #define BGRT\_SPIN\_FREE(  
    *arg* ) bgrt\_spin\_free(&((*arg*)->lock))

Wrapper macro.

Lock wrapper for *arg*->lock spinlock. Empty macro in single core system.

**5.4.2.11 BGRT\_SPIN\_INIT** #define BGRT\_SPIN\_INIT(  
    *arg* ) bgrt\_spin\_init(&((*arg*)->lock))

Wrapper macro.

Initialization wrapper for *arg*->lock spinlock. Empty macro in single core system.

**5.4.2.12 BGRT\_SPIN\_LOCK** #define BGRT\_SPIN\_LOCK(  
    *arg* ) bgrt\_spin\_lock(&((*arg*)->lock))

Wrapper macro.

Lock wrapper for *arg*->lock spinlock. Empty macro in single core system.

**5.4.2.13 BGRT\_ST\_EAGAIN** #define BGRT\_ST\_EAGAIN ((bgrt\_st\_t)7)

Try again.

**5.4.2.14 BGRT\_ST\_EEMPTY** `#define BGRT_ST_EEMPTY ((bgrt_st_t)3)`

The process list is empty.

**5.4.2.15 BGRT\_ST\_ENULL** `#define BGRT_ST_ENULL ((bgrt_st_t)1)`

Null pointer argument.

**5.4.2.16 BGRT\_ST\_EOWN** `#define BGRT_ST_EOWN ((bgrt_st_t)2)`

Ownership error.

**5.4.2.17 BGRT\_ST\_ESTAT** `#define BGRT_ST_ESTAT ((bgrt_st_t)6)`

Process state error.

**5.4.2.18 BGRT\_ST\_ESYNC** `#define BGRT_ST_ESYNC ((bgrt_st_t)4)`

Wrong `bgrt_sync_t` object.

**5.4.2.19 BGRT\_ST\_ETIMEOUT** `#define BGRT_ST_ETIMEOUT ((bgrt_st_t)5)`

Timeout expired.

**5.4.2.20 BGRT\_ST\_IDLE** `#define BGRT_ST_IDLE ((bgrt_st_t)10)`

The system is IDLE.

**5.4.2.21 BGRT\_ST\_OK** `#define BGRT_ST_OK ((bgrt_st_t)0)`

Success.

**5.4.2.22 BGRT\_ST\_ROLL** `#define BGRT_ST_ROLL ((bgrt_st_t)9)`

Next iteration needed.

**5.4.2.23 BGRT\_ST\_SCALL** `#define BGRT_ST_SCALL ((bgrt_st_t)8)`

Wrong system call.

### 5.4.3 Typedef Documentation

**5.4.3.1 bgrt\_code\_t** `typedef void(* bgrt_code_t) (void *)`

Executable code.

A pointer to a void function, that takes void pointer as argument.

### 5.4.4 Function Documentation

**5.4.4.1 bgrt\_curr\_cpu()** `bgrt_cpuid_t bgrt_curr_cpu (void )`

Returns processor core id.

This function returns an id of a processor core on which it is run.

#### Warning

Internal usage function.

**5.4.4.2 bgrt\_curr\_proc()** `bgrt_proc_t* bgrt_curr_proc (void )`

Current process.

#### Warning

Internal usage function.

#### Returns

a pointer to a current process on a local processor core.



**5.4.4.3 bgrt\_init()** `void bgrt_init (`  
`void )`

The Kernel initiation.

Initiates the Kernel before the OS start.

**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 )`

A process stack initialization.

This function prepares a process stack for running a process. It treats a process stack in such a way that `pmain(arg)` is called when a process context is restored from a process stack.

#### Warning

Internal usage function.

#### Parameters

<i>sstart</i>	a process stack bottom.
<i>pmain</i>	a pointer to a function to call.
<i>arg</i>	an argument to a function to call.
<i>return_address</i>	an address to return from pmain.

#### Returns

a pointer to a prepared process stack top.

**5.4.4.5 bgrt\_resched()** `void bgrt_resched (`  
`bgrt_cpuid_t core_id )`

Rescheduling.

Launches a reschedule sequence on one of the processor cores of the system.

#### Warning

Internal usage function.

#### Parameters

<i>core</i> <i>_id</i>	a processor core id.
---------------------------	----------------------

**5.4.4.6 bgrt\_spin\_free()** `void bgrt_spin_free ( bgrt_lock_t * lock )`

Unlock spin-lock on MP system.

**Warning**

Internal usage function.

**Parameters**

<i>lock</i>	a pointer to a spin-lock
-------------	--------------------------

**5.4.4.7 bgrt\_spin\_init()** `void bgrt_spin_init ( bgrt_lock_t * lock )`

Spin-lock initialization for MP system.

**Warning**

Internal usage function.

**Parameters**

<i>lock</i>	a pointer to a spin-lock
-------------	--------------------------

**5.4.4.8 bgrt\_spin\_lock()** `void bgrt_spin_lock ( bgrt_lock_t * lock )`

Lock spin-lock on MP system.

**Warning**

Internal usage function.

**Parameters**

<i>lock</i>	a pointer to a spin-lock
-------------	--------------------------

**5.4.4.9 bgrt\_start()** `void bgrt_start (`  
`void )`

The OS start.

The OS start. It is not necessary to write any code after call of this function, because such a code won't be run normally.

**5.4.4.10 bgrt\_stat\_calc\_load()** `bgrt_load_t bgrt_stat_calc_load (`  
`bgrt_prio_t prio,`  
`bgrt_ls_t * stat )`

Load calculation.

Processor core load calculation.

#### Warning

Internal usage function.

#### Parameters

<i>prio</i>	a priority of a process for which we want to compute a load.
<i>stat</i>	a pointer to a Kernel statistic structure.

#### Returns

current estimation of the core load.

**5.4.4.11 bgrt\_stat\_dec()** `void bgrt_stat_dec (`  
`bgrt_proc_t * proc,`  
`bgrt_ls_t * stat )`

Statistic update on load decrease.

Statistic update on a process stop or a process cut from a signal wait list.

#### Warning

Internal usage function.

#### Parameters

<i>proc</i>	a pointer to a process.
<i>stat</i>	a pointer to a <code>bgrt_ls_t</code> structure.

**5.4.4.12 bgrt\_stat\_inc()** `void bgrt_stat_inc (`  
    `bgrt_proc_t * proc,`  
    `bgrt_ls_t * stat )`

Statistic update on load increase.

Statistic increase on a process run or a process insert to a signal wait list.

**Warning**

Internal usage function.

**Parameters**

<i>proc</i>	a pointer to a process.
<i>stat</i>	a pointer to a <code>bgrt_ls_t</code> structure.

**5.4.4.13 bgrt\_stat\_init()** `void bgrt_stat_init (`  
    `bgrt_ls_t * stat )`

Statistic initialization.

Initiates a `bgrt_ls_t` structure, in which processor core load information is stored.

**Warning**

Internal usage function.

**Parameters**

<i>stat</i>	a pointer to a <code>bgrt_ls_t</code> structure.
-------------	--

**5.4.4.14 bgrt\_stat\_merge()** `void bgrt_stat_merge (`  
    `bgrt_ls_t * src_stat,`  
    `bgrt_ls_t * dst_stat )`

Statistic merge.

Updates Kernel and a signal statistic when signal wait list is merged with scheduler ready process list.

**Warning**

Internal usage function.

## Parameters

<i>src_stat</i>	a pointer to a signal statistic structure.
<i>dst_stat</i>	a pointer to a Kernel statistic structure.

**5.4.4.15 bgrt\_switch\_to\_proc()** `void bgrt_switch_to_proc ( void )`

Kernel to process context switch.

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

A system call.

This function switches a processor core from a process context to the kernel context. The kernel code is always run in the kernel context. This is done to save memory in process stacks. A system calls are done on every operations with processes, mutexes, semaphores and signals. The Kernel does all of this job.

## Warning

Internal usage function.

## Parameters

<i>num</i>	a number of a system call (what is going to be done).
<i>arg</i>	a system call argument (a pointer to an object to be processed).

## 5.5 bugurtos/kernel/crit\_sec.h File Reference

A critical section header.

### Macros

- `#define BGRT_CRIT_SEC_ENTER()`  
*A wrapper macro.*
- `#define BGRT_CRIT_SEC_EXIT() bgrt_priv_crit_sec_exit(current_core)`  
*A wrapper macro.*

## Functions

- `bgrt_cpuid_t bgrt_priv_crit_sec_enter` (void)  
*A critical section start on a multicore system.*
- `void bgrt_priv_crit_sec_exit` (`bgrt_cpuid_t` core)  
*A critical section end on a multicore system.*

### 5.5.1 Detailed Description

A critical section header.

A critical section is a part of a code where interrupts are disabled. Critical sections are used when a common data are used for a short time. Critical sections may be nested, in this case interrupts get enabled on exit from all critical sections.

### 5.5.2 Macro Definition Documentation

#### 5.5.2.1 BGRT\_CRIT\_SEC\_ENTER `#define BGRT_CRIT_SEC_ENTER( )`

**Value:**

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

A wrapper macro.

A critical section start.

**Warning**

Must be used on a start of a code block!

All local variables must be declared before `BGRT_CRIT_SEC_ENTER`, and all executable code must be below it.

#### 5.5.2.2 BGRT\_CRIT\_SEC\_EXIT `#define BGRT_CRIT_SEC_EXIT( ) bgrt_priv_crit_sec_exit(current_↵ core)`

A wrapper macro.

A critical section end.

**Warning**

Must be used at the end of a code block.

### 5.5.3 Function Documentation

**5.5.3.1 bgrt\_priv\_crit\_sec\_enter()** `bgrt_cpuid_t bgrt_priv_crit_sec_enter ( void )`

A critical section start on a multicore system.

#### Returns

An ID of the local CPU core.

**5.5.3.2 bgrt\_priv\_crit\_sec\_exit()** `void bgrt_priv_crit_sec_exit ( bgrt_cpuid_t core )`

A critical section end on a multicore system.

#### Parameters

<i>core</i>	An id of a local CPU core.
-------------	----------------------------

#### Warning

If core param don't match the local CPU core the behavior is undefined.

## 5.6 bugurtos/kernel/default/syscall\_api.h File Reference

System call header.

#### Macros

- #define `BGRT_PROC_RUN(pid) BGRT_SYSCALL_N(PROC_RUN, (void *) (pid))`  
*A process launch routine.*
- #define `BGRT_PROC_RESTART(pid) BGRT_SYSCALL_N(PROC_RESTART, (void *) (pid))`  
*A process restart routine.*
- #define `BGRT_PROC_STOP(pid) BGRT_SYSCALL_N(PROC_STOP, (void *) (pid))`  
*A process stop routine.*
- #define `BGRT_PROC_SELF_STOP()` `BGRT_SYSCALL_N(PROC_SELF_STOP, (void *) 0)`  
*A process self stop routine.*
- #define `BGRT_PROC_LOCK()` `BGRT_SYSCALL_N(PROC_LOCK, (void *) 0)`  
*Set BGRT\_PROC\_FLG\_LOCK for caller process.*
- #define `BGRT_PROC_FREE()` `BGRT_SYSCALL_N(PROC_FREE, (void *) 0)`  
*A BGRT\_PROC\_FLG\_PRE\_STOP flag processing routine.*
- #define `BGRT_PROC_RESET_WATCHDOG()` `BGRT_SYSCALL_N(PROC_RESET_WATCHDOG, (void *) 0)`  
*A watchdog reset routine for real time processes.*
- #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)))`  
*Get a priority of a process.*

- #define `BGRT_PROC_SET_PRIO(pid, prio) BGRT_SYSCALL_NVAR(PROC_SET_PRIO, (void *) (pid), (int) (prio))`  
*Set a priority of a process.*
- #define `BGRT_PROC_GET_ID(pid_ptr) (*(pid_ptr) = BGRT_PID_NOTHING, BGRT_SYSCALL_N(PROC_GET_ID, (void *) (pid_ptr)))`  
*Get a current process ID.*
- #define `BGRT_SYNC_SET_OWNER(sync_ptr, pid) BGRT_SYSCALL_NVAR(SYNC_SET_OWNER, (void *) (sync_ptr), (void *) (pid))`  
*Set bgrt\_sync\_t object 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)))`  
*Get current bgrt\_sync\_t object owner.*
- #define `BGRT_SYNC_OWN(sync_ptr, touch) BGRT_SYSCALL_NVAR(SYNC_OWN, (void *) (sync_ptr), (int) (touch))`  
*Own bgrt\_sync\_t object.*
- #define `BGRT_SYNC_TOUCH(sync_ptr) BGRT_SYSCALL_N(SYNC_TOUCH, (void *) (sync_ptr))`  
*Touch bgrt\_sync\_t object.*
- #define `BGRT_SYNC_SLEEP(sync_ptr, touch_ptr) BGRT_SYSCALL_NVAR(SYNC_SLEEP, (void *) (sync_ptr), (void *) (touch_ptr))`  
*Sleep to wait for synchronization.*
- #define `BGRT_SYNC_WAKE(sync_ptr, pid, chown) BGRT_SYSCALL_NVAR(SYNC_WAKE, (void *) (sync_ptr), (void *) (pid), (int) (chown))`  
*Sleep to wait for synchronization.*
- #define `BGRT_SYNC_WAIT(sync_ptr, pid_ptr, block) BGRT_SYSCALL_NVAR(SYNC_WAIT, (void *) (sync_ptr), (void *) (pid_ptr), (int) (block))`  
*Sleep to wait for synchronization.*
- #define `BGRT_SYNC_PROC_TIMEOUT(pid) BGRT_SYSCALL_N(SYNC_PROC_TIMEOUT, (void *) (pid))`  
*Wake a process on timeout.*

### 5.6.1 Detailed Description

System call header.

#### Warning

This file content is internal usage!

### 5.6.2 Macro Definition Documentation

#### 5.6.2.1 BGRT\_PROC\_FREE #define BGRT\_PROC\_FREE( ) BGRT\_SYSCALL\_N(PROC\_FREE, (void \*) 0)

A BGRT\_PROC\_FLG\_PRE\_STOP flag processing routine.

#### 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)))

Get a current process ID.



**Parameters**

<i>pid_ptr</i>	- A process ID read buffer pointer.
----------------	-------------------------------------

**Returns**

- A system call result.

```
5.6.2.3 BGR_T_PROC_GET_PRI_O #define BGR_T_PROC_GET_PRI_O(
    pid,
    pri_ptr ) (*(pri_ptr) = BGR_T_PRI_O_LOWEST+1, BGR_T_SYSCALL_N(PRI_O_GET_PRI_O,
(void *) (pri_ptr), (void *) (pid)))
```

Get a priority of a process.

**Parameters**

<i>pid</i>	- A process ID.
<i>pri_ptr</i>	- A priority read buffer pointer

**Returns**

- A system call result.

```
5.6.2.4 BGR_T_PROC_LOCK #define BGR_T_PROC_LOCK( ) BGR_T_SYSCALL_N(PRI_O_LOCK, (void *)0)
```

Set BGR\_T\_PROC\_FLG\_LOCK for caller process.

```
5.6.2.5 BGR_T_PROC_RESET_WATCHDOG #define BGR_T_PROC_RESET_WATCHDOG( ) BGR_T_SYSCALL_N(PRI_O_RESET_WATCHDOG, (void *)0)
```

A watchdog reset routine for real time processes.

If a caller process is real time, then this function resets its timer. If a real time process failed to reset its watchdog, then the scheduler stops such process and wakes up next ready process.

```
5.6.2.6 BGR_T_PROC_RESTART #define BGR_T_PROC_RESTART(
    pid ) BGR_T_SYSCALL_N(PRI_O_RESTART, (void *) (pid))
```

A process restart routine.

This function reinitializes a process and schedules it if possible.

**Parameters**

<i>pid</i>	- A process ID.
------------	-----------------

**Returns**

BGRT\_ST\_OK - if a process has been scheduled, error code in other cases.

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

A process launch routine.

This function schedules a process if possible.

**Parameters**

<i>pid</i>	- A process ID.
------------	-----------------

**Returns**

BGRT\_ST\_OK - if a process has been scheduled, error code in other cases.

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

A process self stop routine.

This function stops caller process.

```
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))
```

Set a priority of a process.

It sets a process priority. A process current state doesn't matter.

**Parameters**

<i>pid</i>	- A process ID.
<i>prio</i>	- New process priority value.

**Returns**

- A system call result.

**5.6.2.10 BGRT\_PROC\_STOP** #define BGRT\_PROC\_STOP(  
*pid*) BGRT\_SYSCALL\_N(PROC\_STOP, (void \*) (pid))

A process stop routine.

This function stops a process if possible.

**Parameters**

<i>pid</i>	- A process ID.
------------	-----------------

**Returns**

BGRT\_ST\_OK - if a process has been stopped, error code in other cases.

**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)))

Get current bgrt\_sync\_t object owner.

**Parameters**

<i>sync_ptr</i>	- A pointer to the object of interest.
<i>pid_ptr</i>	- A process ID read buffer pointer.

**Returns**

BGRT\_ST\_OK on success, or error code.

**5.6.2.12 BGRT\_SYNC\_OWN** #define BGRT\_SYNC\_OWN(  
*sync\_ptr*,  
*touch*) BGRT\_SYSCALL\_NVAR(SYNC\_OWN, (void \*) (sync\_ptr), (int) (touch))

Own bgrt\_sync\_t object.

**Parameters**

<i>sync_ptr</i>	A pointer to the object of interest.
<i>touch</i>	If not 0 then mark sync as dirty on fail.

**Returns**

BGRT\_ST\_OK if on success, or error code.

**5.6.2.13 BGRT\_SYNC\_PROC\_TIMEOUT** #define BGRT\_SYNC\_PROC\_TIMEOUT(  
*pid*) BGRT\_SYSCALL\_N(SYNC\_PROC\_TIMEOUT, (void \*)*pid*)

Wake a process on timeout.

**Parameters**

<i>pid</i>	A pointer to a process, that is supposed to wake up.
------------	--

**Returns**

BGRT\_ST\_OK if target process has been woken up, BGRT\_ST\_EAGAIN if caller must do next iteration, or error code.

**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*))

Set bgrt\_sync\_t object owner.

**Parameters**

<i>sync_ptr</i>	A pointer to the object of interest.
<i>pid</i>	A unique ID of new bgrt_sync_t object owner.

**Returns**

BGRT\_ST\_OK on success, or error code.

**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*))

Sleep to wait for synchronization.

Blocks caller process.

## Parameters

<i>sync_ptr</i>	A pointer to the object of interest.
<i>touch_ptr</i>	A touch flag buffer pointer. The buffer value must be 1 if we've touched a sync before call.

## Returns

BGRT\_ST\_OK on success, or error number.

**5.6.2.16 BGRT\_SYNC\_TOUCH** `#define BGRT_SYNC_TOUCH(  
     sync_ptr ) BGRT_SYSCALL_N(SYNC_TOUCH, (void *) (sync_ptr))`

Touch `bgrt_sync_t` object.

## Parameters

<i>sync_ptr</i>	A pointer to the object of interest.
-----------------	--------------------------------------

## Returns

BGRT\_ST\_OK if on success, or error code.

**5.6.2.17 BGRT\_SYNC\_WAIT** `#define BGRT_SYNC_WAIT(  
     sync_ptr,  
     pid_ptr,  
     block ) BGRT_SYSCALL_NVAR(SYNC_WAIT, (void *) (sync_ptr), (void *) (pid_ptr), (int) (block))`

Sleep to wait for synchronization.

Wait until target process is blocked on target `bgrt_sync_t` object.

## Parameters

<i>sync_ptr</i>	A <code>bgrt_sync_t</code> object pointer.
<i>pid_ptr</i>	A pointer to an ID of a process, that is supposed to block. If <code>*pid</code> is <code>BGRT_PID_NOTHING</code> , then caller may wait for first process to block on <code>bgrt_sync_t</code> object.
<i>block</i>	Block flag. If non 0 and caller process must wait, then caller is blocked until target process is blocked on <code>bgrt_sync_t</code> object.

## Returns

BGRT\_ST\_OK if target process has blocked on target `bgrt_sync_t` object, or error code.

```

5.6.2.18 BGRT_SYNC_WAKE #define BGRT_SYNC_WAKE(
    sync_ptr,
    pid,
    chown ) BGRT_SYSCALL_NVAR(SYNC_WAKE, (void *) (sync_ptr), (void *) (pid), (int) (chown))

```

Sleep to wait for synchronization.

Unblock some waiting process. A process should be blocked on target `bgrt_sync_t` object.

#### Parameters

<code>sync_ptr</code>	A <code>bgrt_sync_t</code> object pointer.
<code>pid</code>	A process ID, that is supposed to wake up. If <code>BGRT_PID_NOTHING</code> , then try to wake up wait list head.
<code>chown</code>	A change owner flag. If non 0, then ownership is given to wake up process.

#### Returns

`BGRT_ST_OK` on process wakeup, or error code.

## 5.7 bugurtos/kernel/default/syscall\_routines.h File Reference

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

#### Typedefs

- typedef `bgrt_st_t(* bgrt_user_func_t) (bgrt_va_wr_t *)`

#### Functions

- `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 Typedef Documentation

**5.7.1.1 bgrt\_user\_func\_t** typedef bgrt\_st\_t(\* bgrt\_user\_func\_t) (bgrt\_va\_wr\_t \*)

## 5.7.2 Function Documentation

**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 bugurtos/kernel/index.h File Reference

An map search header.

### Functions

- bgrt\_prio\_t [bgrt\\_map\\_search](#) (bgrt\_map\_t map)

### 5.8.1 Detailed Description

An map search header.

### 5.8.2 Function Documentation

**5.8.2.1 bgrt\_map\_search()** bgrt\_prio\_t bgrt\_map\_search (   
 bgrt\_map\_t map )

An map search.

#### Warning

Internal usage function.

#### Parameters

<i>map</i>	An map.
------------	---------

#### Returns

Highest priority of an map (with minimal value).

## 5.9 bugurtos/kernel/item.h File Reference

A list item header.

#### Data Structures

- struct [bgrt\\_priv\\_item\\_t](#)  
*A list item.*

#### Macros

- #define [BGRT\\_ITEM\\_T\\_INIT](#)(a) { ([bgrt\\_item\\_t](#) \*)&a, ([bgrt\\_item\\_t](#) \*)&a }

#### Typedefs

- typedef struct [bgrt\\_priv\\_item\\_t](#) [bgrt\\_item\\_t](#)

#### Functions

- void [bgrt\\_item\\_init](#) ([bgrt\\_item\\_t](#) \*item)  
*An [bgrt\\_item\\_t](#) object initiation.*
- void [bgrt\\_item\\_insert](#) ([bgrt\\_item\\_t](#) \*item, [bgrt\\_item\\_t](#) \*head)  
*Insert an item to a list.*
- void [bgrt\\_item\\_cut](#) ([bgrt\\_item\\_t](#) \*item)  
*Cut an item from a list.*

### 5.9.1 Detailed Description

A list item header.

### 5.9.2 Macro Definition Documentation

**5.9.2.1 BGRT\_ITEM\_T\_INIT** #define BGRT\_ITEM\_T\_INIT(  
a ) { ([bgrt\\_item\\_t](#) \*)&a, ([bgrt\\_item\\_t](#) \*)&a }

Static item initiation.

#### Warning

For internal usage.

## Parameters

<i>a</i>	An <code>bgrt_item_t</code> variable name.
----------	--

### 5.9.3 Typedef Documentation

#### 5.9.3.1 `bgrt_item_t` typedef struct `bgrt_priv_item_t` `bgrt_item_t`

See `bgrt_priv_item_t`;

### 5.9.4 Function Documentation

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

Cut an item from a list.

##### Warning

Internal usage function.

## Parameters

<i>item</i>	A pointer to an item to cut.
-------------	------------------------------

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

An `bgrt_item_t` object initiation.

##### Warning

Internal usage function.

## Parameters

<i>item</i>	An <code>bgrt_item_t</code> pointer.
-------------	--------------------------------------

**5.9.4.3 bgrt\_item\_insert()** `void bgrt_item_insert (`  
    `bgrt_item_t * item,`  
    `bgrt_item_t * head )`

Insert an item to a list.

#### Warning

Internal usage function.

#### Parameters

<i>item</i>	A pointer to an item.
<i>head</i>	A pointer to a destination list head.

## 5.10 bugurtos/kernel/kernel.h File Reference

A kernel header.

### Data Structures

- struct [bgrt\\_priv\\_kblock\\_t](#)  
    *A BuguRTOS kernel block structure.*
- struct [bgrt\\_priv\\_kernel\\_t](#)  
    *A BuguRTOS kernel structure.*

### Macros

- #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)

### Typedefs

- typedef struct [bgrt\\_priv\\_kblock\\_t](#) [bgrt\\_kblock\\_t](#)
- typedef struct [bgrt\\_priv\\_kernel\\_t](#) [bgrt\\_kernel\\_t](#)

### Functions

- void [bgrt\\_kblock\\_init](#) ([bgrt\\_kblock\\_t](#) \*kblock)  
    *A bgrt\_kblock\_t object initialization.*
- void [bgrt\\_kblock\\_do\\_work](#) ([bgrt\\_kblock\\_t](#) \*kblock)  
    *Software interrupt processing.*
- void [bgrt\\_kblock\\_main](#) ([bgrt\\_kblock\\_t](#) \*kblock)  
    *A kernel thread main function.*
- void [bgrt\\_kernel\\_init](#) (void)  
    *The kernel initiation.*

## Variables

- [bgrt\\_kernel\\_t bgrt\\_kernel](#)  
The *BuguRTOS* kernel.

### 5.10.1 Detailed Description

A kernel header.

### 5.10.2 Macro Definition Documentation

**5.10.2.1 BGRT\_KBLOCK\_PWRSV** `#define BGRT_KBLOCK_PWRSV ((bgrt_map_t)0x8)`

A power save mode vector.

**5.10.2.2 BGRT\_KBLOCK\_VRESCH** `#define BGRT_KBLOCK_VRESCH ((bgrt_map_t)0x4)`

A CPU reschedule vector.

**5.10.2.3 BGRT\_KBLOCK\_VSCALL** `#define BGRT_KBLOCK_VSCALL ((bgrt_map_t)0x1)`

A system call vector.

**5.10.2.4 BGRT\_KBLOCK\_VSCHMSK** `#define BGRT_KBLOCK_VSCHMSK ((bgrt_map_t)0x6)`

A chrduler vector mask.

**5.10.2.5 BGRT\_KBLOCK\_VTMR** `#define BGRT_KBLOCK_VTMR ((bgrt_map_t)0x2)`

A system timer vector.

### 5.10.3 Typedef Documentation

**5.10.3.1 bgrt\_kblock\_t** `typedef struct bgrt_priv_kblock_t bgrt_kblock_t`

See `bgrt_priv_kblock_t`;

**5.10.3.2 bgrt\_kernel\_t** `typedef struct bgrt_priv_kernel_t bgrt_kernel_t`

See `bgrt_priv_kernel_t`;

### 5.10.4 Function Documentation

**5.10.4.1 bgrt\_kblock\_do\_work()** `void bgrt_kblock_do_work (  
    bgrt_kblock_t * kblock )`

Software interrupt processing.

**Parameters**

<i>kblock</i>	A <code>bgrt_kblock_t</code> object pointer.
---------------	--

**5.10.4.2 `bgrt_kblock_init()`** `void bgrt_kblock_init (`  
`bgrt_kblock_t * kblock )`

A `bgrt_kblock_t` object initialization.

**Parameters**

<i>kblock</i>	A <code>bgrt_kblock_t</code> object pointer.
---------------	--

**5.10.4.3 `bgrt_kblock_main()`** `void bgrt_kblock_main (`  
`bgrt_kblock_t * kblock )`

A kernel thread main function.

**Parameters**

<i>kblock</i>	A <code>bgrt_kblock_t</code> object pointer.
---------------	--

**5.10.4.4 `bgrt_kernel_init()`** `void bgrt_kernel_init (`  
`void )`

The kernel initiation.

This function prepares the kernel to work.

**Warning**

Internal usage function.

**5.10.5 Variable Documentation**

**5.10.5.1 `bgrt_kernel`** `bgrt_kernel_t bgrt_kernel`

The BuguRTOS kernel.

It's the one for the entire system!

## 5.11 bugurtos/kernel/pcounter.h File Reference

A locked resource counter header.

### Data Structures

- struct [bgrt\\_priv\\_pcounter\\_t](#)  
*A locked resource counter.*

### Macros

- #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))

### Typedefs

- typedef struct [bgrt\\_priv\\_pcounter\\_t](#) [bgrt\\_pcounter\\_t](#)

### Functions

- [bgrt\\_cnt\\_t bgrt\\_cnt\\_inc](#) ([bgrt\\_cnt\\_t](#) val)  
*Increment counter.*
- [bgrt\\_cnt\\_t bgrt\\_cnt\\_dec](#) ([bgrt\\_cnt\\_t](#) val)  
*Decrement counter.*
- [bgrt\\_cnt\\_t bgrt\\_cnt\\_add](#) ([bgrt\\_cnt\\_t](#) a, [bgrt\\_cnt\\_t](#) b)  
*Addition to counter value.*
- [bgrt\\_cnt\\_t bgrt\\_cnt\\_sub](#) ([bgrt\\_cnt\\_t](#) a, [bgrt\\_cnt\\_t](#) b)  
*Subtraction from counter value.*
- void [bgrt\\_pcounter\\_init](#) ([bgrt\\_pcounter\\_t](#) \*pcounter)  
*A bgrt\_pcounter\_t object initiation.*
- void [bgrt\\_pcounter\\_inc](#) ([bgrt\\_pcounter\\_t](#) \*pcounter, [bgrt\\_prio\\_t](#) prio)  
*Increment counter.*
- [bgrt\\_map\\_t bgrt\\_pcounter\\_dec](#) ([bgrt\\_pcounter\\_t](#) \*pcounter, [bgrt\\_prio\\_t](#) prio)  
*Decrement counter.*
- void [bgrt\\_pcounter\\_plus](#) ([bgrt\\_pcounter\\_t](#) \*pcounter, [bgrt\\_prio\\_t](#) prio, [bgrt\\_cnt\\_t](#) count)  
*Increase counter by a number of steps.*
- [bgrt\\_map\\_t bgrt\\_pcounter\\_minus](#) ([bgrt\\_pcounter\\_t](#) \*pcounter, [bgrt\\_prio\\_t](#) prio, [bgrt\\_cnt\\_t](#) count)  
*Decrease counter by a number of steps;.*

#### 5.11.1 Detailed Description

A locked resource counter header.

## 5.11.2 Macro Definition Documentation

**5.11.2.1 BGRT\_CNT\_ADD** `#define BGRT_CNT_ADD(  
    cnt,  
    delta ) (cnt = bgrt\_cnt\_add(cnt, delta))`

A wrappet for `bgrt_cnt_add`;

**5.11.2.2 BGRT\_CNT\_DEC** `#define BGRT_CNT_DEC(  
    cnt ) (cnt = bgrt\_cnt\_dec(cnt))`

A wrappet for `bgrt_cnt_dec`;

**5.11.2.3 BGRT\_CNT\_INC** `#define BGRT_CNT_INC(  
    cnt ) (cnt = bgrt\_cnt\_inc(cnt))`

A wrappet for `bgrt_cnt_inc`;

**5.11.2.4 BGRT\_CNT\_SUB** `#define BGRT_CNT_SUB(  
    cnt,  
    delta ) (cnt = bgrt\_cnt\_sub(cnt, delta))`

A wrappet for `bgrt_cnt_sub`;

## 5.11.3 Typedef Documentation

**5.11.3.1 bgrt\_pcounter\_t** `typedef struct bgrt\_priv\_pcounter\_t bgrt\_pcounter\_t`

See `bgrt_priv_pcounter_t`;

## 5.11.4 Function Documentation

**5.11.4.1 bgrt\_cnt\_add()** `bgrt\_cnt\_t bgrt\_cnt\_add (  
    bgrt\_cnt\_t a,  
    bgrt\_cnt\_t b )`

Addition to counter value.

### Warning

Internal usage function.



**Parameters**

<i>a</i>	Current counter value.
<i>b</i>	Change of value.

**Returns**

New counter value.

**5.11.4.2 bgrt\_cnt\_dec()** `bgrt_cnt_t bgrt_cnt_dec ( bgrt_cnt_t val )`

Decrement counter.

**Warning**

Internal usage function.

**Parameters**

<i>val</i>	Current counter value.
------------	------------------------

**Returns**

New counter value.

**5.11.4.3 bgrt\_cnt\_inc()** `bgrt_cnt_t bgrt_cnt_inc ( bgrt_cnt_t val )`

Increment counter.

**Warning**

Internal usage function.

**Parameters**

<i>val</i>	Current counter value.
------------	------------------------

**Returns**

New counter value.

**5.11.4.4 bgrt\_cnt\_sub()** `bgrt_cnt_t bgrt_cnt_sub (`  
    `bgrt_cnt_t a,`  
    `bgrt_cnt_t b )`

Subtraction from counter value.

**Warning**

Internal usage function.

**Parameters**

<i>a</i>	Current counter value.
<i>b</i>	Change of value.

**Returns**

New counter value.

**5.11.4.5 bgrt\_pcounter\_dec()** `bgrt_map_t bgrt_pcounter_dec (`  
    `bgrt_pcounter_t * pcounter,`  
    `bgrt_prio_t prio )`

Decrement counter.

**Warning**

Internal usage function.

**Parameters**

<i>pcounter</i>	A <code>bgrt_pcounter_t</code> pointer.
<i>prio</i>	A priority.

**5.11.4.6 bgrt\_pcounter\_inc()** `void bgrt_pcounter_inc (`  
    `bgrt_pcounter_t * pcounter,`  
    `bgrt_prio_t prio )`

Increment counter.

**Warning**

Internal usage function.

## Parameters

<i>pcounter</i>	A <code>bgrt_pcounter_t</code> pointer.
<i>prio</i>	A priority.

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

A `bgrt_pcounter_t` object initiation.

## Warning

Internal usage function.

## Parameters

<i>pcounter</i>	A <code>bgrt_pcounter_t</code> pointer.
-----------------	---

**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 )`

Decrease counter by a number of steps;

## Warning

Internal usage function.

## Parameters

<i>pcounter</i>	A <code>bgrt_pcounter_t</code> pointer.
<i>prio</i>	A priority.
<i>count</i>	A number of decrement steps.

## Returns

0 if correspondent counter is nulled, not 0 else.

**5.11.4.9 `bgrt_pcounter_plus()`** `void bgrt_pcounter_plus (`  
`bgrt_pcounter_t * pcounter,`

```

    bgrt_prio_t prio,
    bgrt_cnt_t count )

```

Increase counter by a number of steps.

#### Warning

Internal usage function.

#### Parameters

<i>pcounter</i>	A <code>bgrt_pcounter_t</code> pointer.
<i>prio</i>	A priority.
<i>count</i>	A number of increment steps.

## 5.12 bugrtos/kernel/pitem.h File Reference

A prioritized list item header.

#### Data Structures

- struct `bgrt_priv_pitem_t`  
*A prioritized list item.*

#### Macros

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

#### Typedefs

- typedef struct `bgrt_priv_pitem_t` `bgrt_pitem_t`

#### Functions

- void `bgrt_pitem_init` (`bgrt_pitem_t *pitem`, `bgrt_prio_t prio`)  
*A `bgrt_pitem_t` object initiation.*
- void `bgrt_pitem_insert` (`bgrt_pitem_t *pitem`, `bgrt_xlist_t *xlist`)  
*Insert `bgrt_pitem_t` object to `bgrt_xlist_t` container.*
- void `bgrt_pitem_fast_cut` (`bgrt_pitem_t *pitem`)  
*Fast cut `bgrt_pitem_t` object from `bgrt_xlist_t` container.*
- void `bgrt_pitem_cut` (`bgrt_pitem_t *pitem`)  
*Cut `bgrt_pitem_t` object from `bgrt_xlist_t` container.*
- `bgrt_pitem_t * bgrt_pitem_xlist_chain` (`bgrt_xlist_t *src`)  
*"Chain" `bgrt_pitem_t` objects from `bgrt_xlist_t` container.*

### 5.12.1 Detailed Description

A prioritized list item header.

### 5.12.2 Macro Definition Documentation

**5.12.2.1 BGR\_T\_PITEM\_T\_INIT** `#define BGR_T_PITEM_T_INIT(  
     a,  
     p ) { BGR_T_ITEM_T_INIT(a), (bgrt_xlist_t *)0, (bgrt_prio_t)p }`

A static `bgrt_pitem_t` object initiation.

#### Warning

For internal usage.

#### Parameters

<i>a</i>	A variable name.
<i>p</i>	A priority.

### 5.12.3 Typedef Documentation

**5.12.3.1 bgrt\_pitem\_t** `typedef struct bgrt_priv_pitem_t bgrt_pitem_t`

### 5.12.4 Function Documentation

**5.12.4.1 bgrt\_pitem\_cut()** `void bgrt_pitem_cut (  
     bgrt_pitem_t * pitem )`

Cut `bgrt_pitem_t` object from `bgrt_xlist_t` container.

This function calls `bgrt_pitem_fast_cut` and then nulls `pitem->list` field.

#### Warning

For internal usage.

**Parameters**

<i>pitem</i>	A <code>bgrt_pitem_t</code> pointer.
--------------	--------------------------------------

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

Fast cut `bgrt_pitem_t` object from `bgrt_xlist_t` container.

This function cuts `bgrt_pitem_t` object from `bgrt_xlist_t` container without `pitem->list` field.

**Warning**

For internal usage.

**Parameters**

<i>pitem</i>	A <code>bgrt_pitem_t</code> pointer.
--------------	--------------------------------------

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

A `bgrt_pitem_t` object initiation.

**Warning**

For internal usage.

**Parameters**

<i>pitem</i>	A <code>bgrt_pitem_t</code> pointer.
<i>prio</i>	A priority.

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

Insert `bgrt_pitem_t` object to `bgrt_xlist_t` container.

**Warning**

For internal usage.

## Parameters

<i>pitem</i>	A <code>bgrt_pitem_t</code> pointer.
<i>xlist</i>	A pointer to destination list.

#### 5.12.4.5 `bgrt_pitem_xlist_chain()` `bgrt_pitem_t*` `bgrt_pitem_xlist_chain ( bgrt_xlist_t * src )`

"Chain" `bgrt_pitem_t` objects from `bgrt_xlist_t` container.

Cut all `bgrt_pitem_t` objects from `bgrt_xlist_t` container and form an ordinary list from them.

## Warning

For internal usage.

## Parameters

<i>src</i>	A <code>bgrt_xlist_t</code> pointer.
------------	--------------------------------------

## Returns

An ordinary doublelinked list head pointer.

## 5.13 bugrtos/kernel/proc.h File Reference

A process header.

## Data Structures

- struct `bgrt_priv_uspd_t`
- struct `bgrt_priv_proc_t`

*A process.*

## Macros

- `#define BGRT_PROC_LRES_INIT(a) bgrt_pcounter_init(&((a)->lres))`  
*Wrapper macro.*
- `#define BGRT_PROC_LRES_INC(a, b) bgrt_pcounter_inc(&((a)->lres), (bgrt_prio_t)b)`  
*Wrapper macro.*
- `#define BGRT_PROC_LRES_DEC(a, b) bgrt_pcounter_dec(&((a)->lres), (bgrt_prio_t)b)`  
*Wrapper macro.*
- `#define BGRT_PID_T bgrt_proc_t *`  
*A unique process ID.*
- `#define BGRT_PID_TO_PROC(p) (p)`

- Lookup the bgrt\_proc\_t object for a given process ID.*

  - #define `BGRT_PROC_TO_PID(p)` (`p`)
- Lookup the ID for a specified bgrt\_proc\_t object.*

  - #define `BGRT_PID_NOTHING` `((BGRT_PID_T)0)`
- An empty process ID.*

  - #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)`
- A real time flag.*

  - #define `BGRT_PROC_FLG_RR` `((bgrt_flag_t)0x40)`
- A round-robin flag.*

  - #define `BGRT_PROC_FLG_LOCK` `((bgrt_flag_t)0x20)`
- A process stop lock flag.*

  - #define `BGRT_PROC_FLG_PRE_STOP` `((bgrt_flag_t)0x10)`
- A process stop preparation flag.*

  - #define `BGRT_PROC_FLG_LOCK_MASK` `((bgrt_flag_t)(BGRT_PROC_FLG_LOCK))`
- A BGRT\_PROC\_FLG\_LOCK.*

  - #define `BGRT_PROC_STATE_CLEAR_MASK` `((bgrt_flag_t)0xF0)`
- An execution state clear mask.*

  - #define `BGRT_PROC_STATE_CLEAR_RUN_MASK` `((bgrt_flag_t)0xFC)`
- An execution state clear mask.*

  - #define `BGRT_PROC_STATE_MASK` `((bgrt_flag_t)0x0F)`
- An execution state mask.*

  - #define `BGRT_PROC_STATE_RESTART_MASK` `((bgrt_flag_t)0x8)`
- A process execution state check mask.*

  - #define `BGRT_PROC_STATE_RUN_MASK` `((bgrt_flag_t)0x3)`
- A process execution state check mask.*

  - #define `BGRT_PROC_STATE_WAIT_MASK` `((bgrt_flag_t)0x8)`
- A process execution state check mask.*

  - #define `BGRT_PROC_STATE_STOPED` `((bgrt_flag_t)0x0)`
- Initial state, stopped.*

  - #define `BGRT_PROC_STATE_END` `((bgrt_flag_t)0x1)`
- Normal process termination.*

  - #define `BGRT_PROC_STATE_READY` `((bgrt_flag_t)0x2)`
- Is ready to run.*

  - #define `BGRT_PROC_STATE_RUNNING` `((bgrt_flag_t)0x3)`
- Is running.*

  - #define `BGRT_PROC_STATE_WD_STOPED` `((bgrt_flag_t)0x4)`
- Watchdog termination.*

  - #define `BGRT_PROC_STATE_DEAD` `((bgrt_flag_t)0x5)`
- Abnormal termination, terminated with waiting ipc transactions.*

  - #define `BGRT_PROC_STATE_TO_READY` `((bgrt_flag_t)0x6)`
- Is ready to run.*

  - #define `BGRT_PROC_STATE_TO_RUNNING` `((bgrt_flag_t)0x7)`
- Is running.*

  - #define `BGRT_PROC_STATE_SYNC_WAIT` `((bgrt_flag_t)0x8)`
- Is waiting for sleeping processes.*

  - #define `BGRT_PROC_STATE_SYNC_SLEEP` `((bgrt_flag_t)0x9)`
- Is waiting for wakeup.*



- #define `BGRT_PROC_STATE_SYNC_READY` ((bgrt\_flag\_t)0xA)  
*Is ready to run.*
- #define `BGRT_PROC_STATE_SYNC_RUNNING` ((bgrt\_flag\_t)0xB)  
*Is running.*
- #define `BGRT_PROC_STATE_PI_PEND` ((bgrt\_flag\_t)0xC)  
*A process is waiting for priority change.*
- #define `BGRT_PROC_STATE_PI_DONE` ((bgrt\_flag\_t)0xD)  
*A process has been run during priority change.*
- #define `BGRT_PROC_STATE_PI_READY` ((bgrt\_flag\_t)0xE)  
*Is ready to run.*
- #define `BGRT_PROC_STATE_PI_RUNNING` ((bgrt\_flag\_t)0xF)  
*Is running.*
- #define `BGRT_PROC_PRE_STOP_TEST`(a) (((a)->flags) & `BGRT_PROC_FLG_PRE_STOP`) && (!(((a)->flags) & `BGRT_PROC_FLG_LOCK_MASK`)))  
*A BGRT\_PROC\_FLG\_PRE\_STOP condition test macro.*
- #define `BGRT_PROC_RUN_TEST`(a) (((a)->flags & `BGRT_PROC_STATE_RUN_MASK`) >= `BGRT_PROC_STATE_READY`)  
*Check if process is ready or running.*
- #define `BGRT_PROC_GET_STATE`(a) ((a)->flags & `BGRT_PROC_STATE_MASK`)  
*Reads a process state.*
- #define `BGRT_PROC_SET_STATE`(a, b) ((a)->flags &= `BGRT_PROC_STATE_CLEAR_MASK`, (a)->flags |= b)  
*Sets a process state.*

## Typedefs

- typedef struct `bgrt_priv_proc_t` `bgrt_proc_t`

## Functions

- void `bgrt_priv_proc_lres_inc` (`bgrt_proc_t` \*proc, `bgrt_prio_t` prio)  
*Process priority control.*
- void `bgrt_priv_proc_lres_dec` (`bgrt_proc_t` \*proc, `bgrt_prio_t` prio)  
*Process priority control.*
- void `bgrt_priv_proc_stop_ensure` (`bgrt_proc_t` \*proc, `bgrt_flag_t` state)  
*Stops a process.*
- `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)  
*A process initialization. Must be used in critical sections and interrupt service routines.*
- `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)  
*A process initialization.*
- void `bgrt_proc_terminate` (void)  
*A process termination routine called after proc->pmain return. Internal usage function.*
- void `bgrt_priv_proc_terminate` (void)  
*A process termination routine called after proc->pmain return. Internal usage function.*
- `bgrt_st_t` `bgrt_priv_proc_run` (`bgrt_proc_t` \*proc)  
*A process launch routine for usage in interrupt service routines and critical sections.*
- `bgrt_st_t` `bgrt_priv_proc_restart` (`bgrt_proc_t` \*proc)  
*A process restart routine for usage in interrupt service routines and critical sections.*

- `bgrt_st_t bgrt_priv_proc_stop (bgrt_proc_t *proc)`  
*A process stop routine for usage in interrupts service routines and critical sections.*
- `void bgrt_priv_proc_self_stop (void)`  
*A process self stop routine.*
- `void bgrt_priv_proc_reset_watchdog (void)`  
*A watchdog reset routine for real time processes.*
- `void bgrt_priv_proc_lock (void)`  
*Set BGRT\_PROC\_FLG\_LOCK for caller process.*
- `void bgrt_priv_proc_free (void)`  
*A BGRT\_PROC\_FLG\_PRE\_STOP flag processing routine.*
- `void bgrt_priv_proc_set_prio (bgrt_proc_t *proc, bgrt_prio_t prio)`  
*Set a priority of a process.*
- `bgrt_prio_t bgrt_priv_proc_get_prio (bgrt_proc_t *proc)`  
*Get a priority of a process.*

### 5.13.1 Detailed Description

A process header.

### 5.13.2 Macro Definition Documentation

**5.13.2.1 BGRT\_GET\_USPD** `#define BGRT_GET_USPD( ) (&(bgrt_curr_proc())->udata)`

Get current process userspace data pointer.

**5.13.2.2 BGRT\_PID\_NOTHING** `#define BGRT_PID_NOTHING ((BGRT_PID_T)0)`

An empty process ID.

**5.13.2.3 BGRT\_PID\_T** `#define BGRT_PID_T bgrt_proc_t *`

A unique process ID.

**5.13.2.4 BGRT\_PID\_TO\_PROC** `#define BGRT_PID_TO_PROC(  
     p ) (p)`

Lookup the `bgrt_proc_t` object for a given process ID.

#### Note

This macro may check the process ID for validity.

## Parameters

<i>p</i>	A process ID.
----------	---------------

## Returns

The `bgrt_proc_t` pointer or a zero pointer.

**5.13.2.5 BGRT\_PROC\_FLG\_LOCK** `#define BGRT_PROC_FLG_LOCK ((bgrt_flag_t)0x20)`

A process stop lock flag.

A process can not be stopped at this moment.

**5.13.2.6 BGRT\_PROC\_FLG\_LOCK\_MASK** `#define BGRT_PROC_FLG_LOCK_MASK ((bgrt_flag_t)(BGRT_PROC_FLG_LOCK))`

A `BGRT_PROC_FLG_LOCK`.

Used to test if a process has locked some resources.

**5.13.2.7 BGRT\_PROC\_FLG\_PRE\_STOP** `#define BGRT_PROC_FLG_PRE_STOP ((bgrt_flag_t)0x10)`

A process stop preparation flag.

A process must be stopped, but it can't be stopped now. It'll be stopped when possible.

**5.13.2.8 BGRT\_PROC\_FLG\_RR** `#define BGRT_PROC_FLG_RR ((bgrt_flag_t)0x40)`

A round-robin flag.

If this flag is set, then round-robin scheduling is used, else FIFO a process is scheduled in fifo manner.

**5.13.2.9 BGRT\_PROC\_FLG\_RT** `#define BGRT_PROC_FLG_RT ((bgrt_flag_t)0x80)`

A real time flag.

This flag enables real time process scheduling policy.

**5.13.2.10 BGRT\_PROC\_GET\_STATE** `#define BGRT_PROC_GET_STATE(  
a) ((a)->flags & BGRT_PROC_STATE_MASK)`

Reads a process state.

## Warning

For internal usage.

**5.13.2.11 BGRT\_PROC\_LRES\_DEC** `#define BGRT_PROC_LRES_DEC(  
a,  
b) bgrt_pcounter_dec(&((a)->lres), (bgrt_prio_t)b)`

Wrapper macro.

A decrement of `proc->lres`.

**Parameters**

<i>a</i>	a pointer to a process.
<i>b</i>	a priority of a <code>bgrt_sync_t</code> object.

**5.13.2.12 BGRT\_PROC\_LRES\_INC** `#define BGRT_PROC_LRES_INC(  
    a,  
    b ) bgrt_pcounter_inc(&((a)->lres), (bgrt_prio_t)b`

Wrapper macro.

An increment of `proc->lres`.

**Parameters**

<i>a</i>	a pointer to a process.
<i>b</i>	a priority of a <code>bgrt_sync_t</code> object.

**5.13.2.13 BGRT\_PROC\_LRES\_INIT** `#define BGRT_PROC_LRES_INIT(  
    a ) bgrt_pcounter_init(&((a)->lres))`

Wrapper macro.

Initiates `proc->lres` field of a process.

**Parameters**

<i>a</i>	a pointer to a process.
----------	-------------------------

**5.13.2.14 BGRT\_PROC\_PRE\_STOP\_TEST** `#define BGRT_PROC_PRE_STOP_TEST(  
    a ) (((a)->flags) & BGRT_PROC_FLG_PRE_STOP) && (!(((a)->flags) & BGRT_PROC_FLG_LOCK_MASK))`

A `BGRT_PROC_FLG_PRE_STOP` condition test macro.

Used to test if a process can be stopped on `BGRT_PROC_FLG_PRE_STOP` flag. A process should not have locked resources at a moment of a flag stop.

**Warning**

For internal usage.

```
5.13.2.15 BGR_T_PROC_RUN_TEST #define BGR_T_PROC_RUN_TEST(  
    a ) ((a)->flags & BGR_T_PROC_STATE_RUN_MASK) >= BGR_T_PROC_STATE_READY)
```

Check if process is ready or running.

#### Warning

For internal usage.

```
5.13.2.16 BGR_T_PROC_SET_STATE #define BGR_T_PROC_SET_STATE(  
    a,  
    b ) ((a)->flags &= BGR_T_PROC_STATE_CLEAR_MASK, (a)->flags |= b)
```

Sets a process state.

#### Warning

For internal usage.

```
5.13.2.17 BGR_T_PROC_STATE_CLEAR_MASK #define BGR_T_PROC_STATE_CLEAR_MASK ((bgr_t_flag_t) 0xF0)
```

An execution state clear mask.

Used clear execution state bits in proc->flags.

```
5.13.2.18 BGR_T_PROC_STATE_CLEAR_RUN_MASK #define BGR_T_PROC_STATE_CLEAR_RUN_MASK ((bgr_t_flag_t) 0xFC)
```

An execution state clear mask.

Used clear execution three LSBs state bits in proc->flags.

```
5.13.2.19 BGR_T_PROC_STATE_DEAD #define BGR_T_PROC_STATE_DEAD ((bgr_t_flag_t) 0x5)
```

Abnormal termination, terminated with waiting ipc transactions.

```
5.13.2.20 BGR_T_PROC_STATE_END #define BGR_T_PROC_STATE_END ((bgr_t_flag_t) 0x1)
```

Normal process termination.

**5.13.2.21 BGR\_T\_PROC\_STATE\_MASK** #define BGR\_T\_PROC\_STATE\_MASK ((bgr\_t\_flag\_t)0x0F)

An execution state mask.

**5.13.2.22 BGR\_T\_PROC\_STATE\_PI\_DONE** #define BGR\_T\_PROC\_STATE\_PI\_DONE ((bgr\_t\_flag\_t)0xD)

A process has been run during priority change.

**5.13.2.23 BGR\_T\_PROC\_STATE\_PI\_PEND** #define BGR\_T\_PROC\_STATE\_PI\_PEND ((bgr\_t\_flag\_t)0xC)

A process is waiting for priority change.

**5.13.2.24 BGR\_T\_PROC\_STATE\_PI\_READY** #define BGR\_T\_PROC\_STATE\_PI\_READY ((bgr\_t\_flag\_t)0xE)

Is ready to run.

**5.13.2.25 BGR\_T\_PROC\_STATE\_PI\_RUNNING** #define BGR\_T\_PROC\_STATE\_PI\_RUNNING ((bgr\_t\_flag\_t)0xF)

Is running.

**5.13.2.26 BGR\_T\_PROC\_STATE\_READY** #define BGR\_T\_PROC\_STATE\_READY ((bgr\_t\_flag\_t)0x2)

Is ready to run.

**5.13.2.27 BGR\_T\_PROC\_STATE\_RESTART\_MASK** #define BGR\_T\_PROC\_STATE\_RESTART\_MASK ((bgr\_t\_flag\_t)0x8)

A process execution state check mask.

Used by bgr\_t\_priv\_proc\_restart to check for restart possibility.

**5.13.2.28 BGR\_T\_PROC\_STATE\_RUN\_MASK** #define BGR\_T\_PROC\_STATE\_RUN\_MASK ((bgr\_t\_flag\_t)0x3)

A process execution state check mask.

Used to check if the process has been run.

**5.13.2.29 BGRT\_PROC\_STATE\_RUNNING** `#define BGRT_PROC_STATE_RUNNING ((bgrt_flag_t)0x3)`

Is running.

**5.13.2.30 BGRT\_PROC\_STATE\_STOPED** `#define BGRT_PROC_STATE_STOPED ((bgrt_flag_t)0x0)`

Initial state, stopped.

**5.13.2.31 BGRT\_PROC\_STATE\_SYNC\_READY** `#define BGRT_PROC_STATE_SYNC_READY ((bgrt_flag_t)0xA)`

Is ready to run.

**5.13.2.32 BGRT\_PROC\_STATE\_SYNC\_RUNNING** `#define BGRT_PROC_STATE_SYNC_RUNNING ((bgrt_flag_t)0xB)`

Is running.

**5.13.2.33 BGRT\_PROC\_STATE\_SYNC\_SLEEP** `#define BGRT_PROC_STATE_SYNC_SLEEP ((bgrt_flag_t)0x9)`

Is waiting for wakeup.

**5.13.2.34 BGRT\_PROC\_STATE\_SYNC\_WAIT** `#define BGRT_PROC_STATE_SYNC_WAIT ((bgrt_flag_t)0x8)`

Is waiting for sleeping processes.

**5.13.2.35 BGRT\_PROC\_STATE\_TO\_READY** `#define BGRT_PROC_STATE_TO_READY ((bgrt_flag_t)0x6)`

Is ready to run.

**5.13.2.36 BGRT\_PROC\_STATE\_TO\_RUNNING** `#define BGRT_PROC_STATE_TO_RUNNING ((bgrt_flag_t)0x7)`

Is running.

**5.13.2.37 BGR\_T\_PROC\_STATE\_WAIT\_MASK** `#define BGR_T_PROC_STATE_WAIT_MASK ((bgrt_flag_t)0x8)`

A process execution state check mask.

Used to check if the process is waiting for synchronization.

**5.13.2.38 BGR\_T\_PROC\_STATE\_WD\_STOPED** `#define BGR_T_PROC_STATE_WD_STOPED ((bgrt_flag_t)0x4)`

Watchdog termination.

**5.13.2.39 BGR\_T\_PROC\_TO\_PID** `#define BGR_T_PROC_TO_PID(  
    p ) (p)`

Lookup the ID for a specified `bgrt_proc_t` object.

#### Note

This macro may check the `bgrt_proc_t` pointer for validity.

#### Parameters

<i>p</i>	A <code>bgrt_proc_t</code> pointer.
----------	-------------------------------------

#### Returns

The process ID.

**5.13.2.40 BGR\_T\_USPD\_INIT** `#define BGR_T_USPD_INIT(  
    proc )`

#### Value:

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

Initialization.

**5.13.2.41 BGR\_T\_USPD\_PROC\_T** `#define BGR_T_USPD_PROC_T struct bgrt\_priv\_uspd\_t`

User space process data.

**5.13.2.42 BGR\_T\_USPD\_T** `#define BGR_T_USPD_T BGR\_T\_USPD\_PROC\_T *`

User space process data.



### 5.13.3 Typedef Documentation

**5.13.3.1 `bgrt_proc_t`** typedef struct `bgrt_priv_proc_t` `bgrt_proc_t`

See `bgrt_priv_proc_t`;

### 5.13.4 Function Documentation

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

A `BGRT_PROC_FLG_PRE_STOP` flag processing routine.

#### Warning

For internal usage.

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

Get a priority of a process.

#### Warning

For internal usage.

#### Parameters

<code>proc</code>	- A process pointer.
-------------------	----------------------

#### Returns

- A process priority value.

**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 )

```

A process initialization. Must be used in critical sections and interrupt service routines.

#### Parameters

<i>proc</i>	A pointer to a initialized process.
<i>pmain</i>	A pointer to a process "main" routine.
<i>sv_hook</i>	A context save hook pointer.
<i>rs_hook</i>	A context save hook pointer.
<i>arg</i>	An argument pointer.
<i>sstart</i>	A process stack bottom pointer.
<i>prio</i>	A process priority.
<i>time_quant</i>	A process time slice.
<i>is_rt</i>	A real time flag. If true, then a process is scheduled in a real time manner.
<i>affinity</i>	A process affinity.

**5.13.4.4 bgrt\_priv\_proc\_lock()** void bgrt\_priv\_proc\_lock ( void )

Set BGRT\_PROC\_FLG\_LOCK for caller process.

#### Warning

For internal usage.

**5.13.4.5 bgrt\_priv\_proc\_lres\_dec()** void bgrt\_priv\_proc\_lres\_dec ( bgrt\_proc\_t \* proc, bgrt\_prio\_t prio )

Process priority control.

Decrements proc->lres counter, clears BGRT\_PROC\_FLG\_LOCK flag if needed.

#### Warning

For internal usage.

## Parameters

<i>proc</i>	- A pointer to a process.
<i>prio</i>	- New process priority value.

**5.13.4.6 bgrt\_priv\_proc\_lres\_inc()** `void bgrt_priv_proc_lres_inc (`  
`bgrt_proc_t * proc,`  
`bgrt_prio_t prio )`

Process priority control.

Increments `proc->lres` counter, sets `BGRT_PROC_FLG_LOCK` flag.

## Warning

For internal usage.

## Parameters

<i>proc</i>	- A pointer to a process.
<i>prio</i>	- New process priority value.

**5.13.4.7 bgrt\_priv\_proc\_reset\_watchdog()** `void bgrt_priv_proc_reset_watchdog (`  
`void )`

A watchdog reset routine for real time processes.

If a caller process is real time, then this function resets its timer. If a real time process failed to reset its watchdog, then the scheduler stops such process and wakes up next ready process.

## Warning

For internal usage.

**5.13.4.8 bgrt\_priv\_proc\_restart()** `bgrt_st_t bgrt_priv_proc_restart (`  
`bgrt_proc_t * proc )`

A process restart routine for usage in interrupt service routines and critical sections.

This function reinitializes a process and schedules it if possible.

**Parameters**

<i>proc</i>	- A pointer to a process to launch.
-------------	-------------------------------------

**Returns**

BGRT\_ST\_OK - if a process has been scheduled, error code in other cases.

**5.13.4.9 bgrt\_priv\_proc\_run()** `bgrt_st_t bgrt_priv_proc_run (`  
`bgrt_proc_t * proc )`

A process launch routine for usage in interrupt service routines and critical sections.

This function schedules a process if possible.

**Parameters**

<i>proc</i>	- A pointer to a process to launch.
-------------	-------------------------------------

**Returns**

BGRT\_ST\_OK - if a process has been scheduled, error code in other cases.

**5.13.4.10 bgrt\_priv\_proc\_self\_stop()** `void bgrt_priv_proc_self_stop (`  
`void )`

A process self stop routine.

This function stops caller process.

**Warning**

For internal usage.

**5.13.4.11 bgrt\_priv\_proc\_set\_prio()** `void bgrt_priv_proc_set_prio (`  
`bgrt_proc_t * proc,`  
`bgrt_prio_t prio )`

Set a priority of a process.

It sets a process priority. A process current state doesn't matter.

**Warning**

For internal usage.

## Parameters

<i>proc</i>	- A pointer to a process.
<i>prio</i>	- New process priority value.

**5.13.4.12 bgrt\_priv\_proc\_stop()** `bgrt_st_t bgrt_priv_proc_stop ( bgrt_proc_t * proc )`

A process stop routine for usage in interrupts service routines and critical sections.

This function stops a process if possible.

## Parameters

<i>proc</i>	- A pointer to a process to stop.
-------------	-----------------------------------

## Returns

BGRT\_ST\_OK - if a process has been stopped, error code in other cases.

**5.13.4.13 bgrt\_priv\_proc\_stop\_ensure()** `void bgrt_priv_proc_stop_ensure ( bgrt_proc_t * proc, bgrt_flag_t state )`

Stops a process.

Stops a process for sure.

## Warning

For internal usage.

## Parameters

<i>proc</i>	- A pointer to a process.
<i>state</i>	- A new process state.

**5.13.4.14 bgrt\_priv\_proc\_terminate()** `void bgrt_priv_proc_terminate ( void )`

A process termination routine called after `proc->pmain` return. Internal usage function.

**Warning**

For internal usage.

```
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 )
```

A process initialization.

**Parameters**

<i>proc</i>	A pointer to a initialized process.
<i>pmain</i>	A pointer to a process "main" routine.
<i>sv_hook</i>	A context save hook pointer.
<i>rs_hook</i>	A context save hook pointer.
<i>arg</i>	An argument pointer.
<i>sstart</i>	A process stack bottom pointer.
<i>prio</i>	A process priority.
<i>time_quant</i>	A process time slice.
<i>is_rt</i>	A real time flag. If true, then a process is scheduled in a real time manner.
<i>affinity</i>	A process affinity.

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

A process termination routine called after `proc->pmain` return. Internal usage function.

**5.14 bugurtos/kernel/sched.h File Reference**

A scheduler header.

**Data Structures**

- struct [bgrt\\_priv\\_sched\\_t](#)  
A scheduler.
- struct [bgrt\\_priv\\_kstat\\_t](#)

## Macros

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

## Typedefs

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

## Functions

- void `bgrt_sched_init (bgrt_sched_t *sched)`  
*A scheduler initiation routine.*
- `bgrt_st_t bgrt_sched_run (bgrt_bool_t is_periodic)`  
*A scheduler function.*
- void `bgrt_sched_proc_run (bgrt_proc_t *proc, bgrt_flag_t state)`  
*A low level process run routine. For internal usage.*
- void `bgrt_sched_proc_stop (bgrt_proc_t *proc, bgrt_flag_t state)`  
*A low level process stop routine. For internal usage.*
- `bgrt_bool_t bgrt_priv_sched_proc_yield (void)`  
*Pass control to next ready process (for internal usage only!).*
- `bgrt_bool_t bgrt_sched_proc_yield (void)`  
*Pass control to next ready process.*
- `bgrt_cpuid_t bgrt_sched_load_balancer (bgrt_proc_t *proc, bgrt_ls_t *stat)`  
*A load balancer routine.*
- `bgrt_cpuid_t bgrt_sched_highest_load_core (bgrt_ls_t *stat)`  
*Find most loaded core.*
- void `bgrt_priv_sched_proc_set_core (bgrt_proc_t *proc)`  
*Assign a core.*
- void `bgrt_sched_lazy_local_load_balancer (void)`  
*A lazy local load balancer routine.*

### 5.14.1 Detailed Description

A scheduler header.

#### Warning

All functions in this file are internal usage functions!!!

### 5.14.2 Macro Definition Documentation

**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 Typedef Documentation

**5.14.3.1 bgrt\_kstat\_t** typedef struct `bgrt_priv_kstat_t` `bgrt_kstat_t`

A statistic for load balancing, CPU hotplug is not supported.

**5.14.3.2 bgrt\_sched\_t** typedef struct `bgrt_priv_sched_t` `bgrt_sched_t`

See `bgrt_priv_sched_t`;

### 5.14.4 Function Documentation

**5.14.4.1 bgrt\_priv\_sched\_proc\_set\_core()** void `bgrt_priv_sched_proc_set_core` (  
`bgrt_proc_t * proc` )

Assign a core.

This assigns a core when a process starts execution.

#### Warning

For internal usage.

#### Parameters

<code>proc</code>	A pointer to a process.
-------------------	-------------------------

**5.14.4.2 bgrt\_priv\_sched\_proc\_yield()** `bgrt_bool_t` `bgrt_priv_sched_proc_yield` (  
void )

Pass control to next ready process (for internal usage only!).

If there is another running process, this function passes control to it.

#### Warning

For internal usage.

#### Returns

One if power saving mode can be used, zero in other cases.



**5.14.4.3 bgrt\_sched\_highest\_load\_core()** `bgrt_cpuid_t bgrt_sched_highest_load_core ( bgrt_ls_t * stat )`

Find most loaded core.

This function is used in Kernel load balancing and in sig\_signal function.

#### Warning

For internal usage.

#### Parameters

<i>stat</i>	A pointer to a <code>bgrt_ls_t</code> array of the kernel or of a signal.
-------------	---

#### Returns

An ID of the most loaded process list.

**5.14.4.4 bgrt\_sched\_init()** `void bgrt_sched_init ( bgrt_sched_t * sched )`

A scheduler initiation routine.

This function prepares a scheduler object for work.

#### Warning

For internal usage.

#### Parameters

<i>sched</i>	- A scheduler pointer.
--------------	------------------------

**5.14.4.5 bgrt\_sched\_lazy\_local\_load\_balancer()** `void bgrt_sched_lazy_local_load_balancer ( void )`

A lazy local load balancer routine.

Transfers one process from a current CPU core to the least loaded CPU core on the system.

**5.14.4.6 bgrt\_sched\_load\_balancer()** `bgrt_cpuid_t bgrt_sched_load_balancer ( bgrt_proc_t * proc, bgrt_ls_t * stat )`

A load balancer routine.

This function is used for load balancing of the kernel and of signals.

**Warning**

For internal usage.

**Parameters**

<i>proc</i>	A pointer to a process that we want to place on a process list.
<i>stat</i>	A pointer to a <code>bgrt_ls_t</code> array, that controls correspondent process list.

**Returns**

An ID of the least loaded process list.

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

A low level process run routine. For internal usage.

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

A low level process stop routine. For internal usage.

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

Pass control to next ready process.

If there is another running process, this function passes control to it.

**Returns**

One if power saving mode can be used, zero in other cases.

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

A scheduler function.

**Warning**

For internal usage.

## Parameters

<code>is_periodic</code>	- A periodic interrupt flag.
--------------------------	------------------------------

## Returns

BGRT\_ST\_OK is new ready process scheduled, BGRT\_ST\_EEMPTY if there were no ready processes.

## 5.15 bugurtos/kernel/sync.h File Reference

A sync header.

## Data Structures

- struct `bgrt_priv_sync_t`  
*Basic synchronization primitive.*

## Macros

- #define `BGRT_SYNC_PRIO(s) bgrt_priv_sync_prio(s)`  
*Calculates a `bgrt_sync_t` object priority.*
- #define `BGRT_SYNC_INIT(s, p) bgrt_sync_init((bgrt_sync_t *)s, (bgrt_prio_t)p)`  
*Watch `bgrt_sync_init`.*
- #define `BGRT_PRIV_SYNC_INIT(s, p) bgrt_priv_sync_init((bgrt_sync_t *)s, (bgrt_prio_t)p)`  
*Watch `bgrt_priv_sync_init`.*

## Typedefs

- typedef struct `bgrt_priv_sync_t` `bgrt_sync_t`

## Functions

- `bgrt_prio_t bgrt_priv_sync_prio (bgrt_sync_t *sync)`  
*Returns current `bgrt_sync_t` object priority.*
- `bgrt_st_t bgrt_sync_init (bgrt_sync_t *sync, bgrt_prio_t prio)`  
*A basic synchronization primitive initiation.*
- `bgrt_st_t bgrt_priv_sync_init (bgrt_sync_t *sync, bgrt_prio_t prio)`  
*A sync initiation for usage in ISRs or in critical sections.*
- `bgrt_st_t bgrt_priv_sync_set_owner (bgrt_sync_t *sync, bgrt_proc_t *proc)`  
*Watch `BGRT_SYNC_SET_OWNER`.*
- `bgrt_proc_t * bgrt_priv_sync_get_owner (bgrt_sync_t *sync)`  
*See `BGRT_SYNC_GET_OWNER`.*
- `bgrt_st_t bgrt_priv_sync_own (bgrt_sync_t *sync, bgrt_flag_t touch)`  
*Watch `BGRT_SYNC_OWN`.*
- `bgrt_st_t bgrt_priv_sync_touch (bgrt_sync_t *sync)`  
*Watch `BGRT_SYNC_TOUCH`.*
- `bgrt_st_t bgrt_priv_sync_wake (bgrt_sync_t *sync, bgrt_proc_t *proc, bgrt_flag_t chown)`  
*Watch `BGRT_SYNC_WAKE`.*
- `bgrt_st_t bgrt_priv_sync_sleep (bgrt_sync_t *sync, bgrt_flag_t *touch)`  
*Watch `BGRT_SYNC_SLEEP`.*
- `bgrt_st_t bgrt_priv_sync_wait (bgrt_sync_t *sync, bgrt_proc_t **proc, bgrt_flag_t block)`  
*Watch `BGRT_SYNC_WAIT`.*
- `bgrt_st_t bgrt_priv_sync_proc_timeout (bgrt_proc_t *proc)`  
*Watch `BGRT_SYNC_PROC_TIMEOUT`.*

### 5.15.1 Detailed Description

A sync header.

### 5.15.2 Macro Definition Documentation

**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)`

Watch `bgrt_priv_sync_init`.

**5.15.2.2 BGRT\_SYNC\_INIT** `#define BGRT_SYNC_INIT(  
 s,  
 p ) bgrt_sync_init((bgrt_sync_t *)s, (bgrt_prio_t)p)`

Watch `bgrt_sync_init`.

**5.15.2.3 BGRT\_SYNC\_PRIO** `#define BGRT_SYNC_PRIO(  
 s ) bgrt_priv_sync_prio(s)`

Calculates a `bgrt_sync_t` object priority.

### 5.15.3 Typedef Documentation

**5.15.3.1 bgrt\_sync\_t** `typedef struct bgrt_priv_sync_t bgrt_sync_t`

See `bgrt_priv_sync_t`;

### 5.15.4 Function Documentation

**5.15.4.1 bgrt\_priv\_sync\_get\_owner()** `bgrt_proc_t* bgrt_priv_sync_get_owner (  
 bgrt_sync_t * sync )`

See `BGRT_SYNC_GET_OWNER`.

#### Warning

For internal usage.

**5.15.4.2 bgrt\_priv\_sync\_init()** `bgrt_st_t bgrt_priv_sync_init (  
 bgrt_sync_t * sync,  
 bgrt_prio_t prio )`

A sync initiation for usage in ISRs or in critical sections.

## Parameters

<i>sync</i>	A sync pointer.
<i>prio</i>	A priority.

**5.15.4.3 bgrt\_priv\_sync\_own()** `bgrt_st_t bgrt_priv_sync_own (`  
`bgrt_sync_t * sync,`  
`bgrt_flag_t touch )`

Watch BGRT\_SYNC\_OWN.

## Warning

For internal usage.

**5.15.4.4 bgrt\_priv\_sync\_prio()** `bgrt_prio_t bgrt_priv_sync_prio (`  
`bgrt_sync_t * sync )`

Returns current `bgrt_sync_t` object priority.

## Warning

For internal usage.

**5.15.4.5 bgrt\_priv\_sync\_proc\_timeout()** `bgrt_st_t bgrt_priv_sync_proc_timeout (`  
`bgrt_proc_t * proc )`

Watch BGRT\_SYNC\_PROC\_TIMEOUT.

## Warning

For internal usage.

**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 )`

Watch BGRT\_SYNC\_SET\_OWNER.

## Warning

For internal usage.

**5.15.4.7 bgrt\_priv\_sync\_sleep()** `bgrt_st_t bgrt_priv_sync_sleep (`  
    `bgrt_sync_t * sync,`  
    `bgrt_flag_t * touch )`

Watch BGRT\_SYNC\_SLEEP.

**Warning**

For internal usage.

**5.15.4.8 bgrt\_priv\_sync\_touch()** `bgrt_st_t bgrt_priv_sync_touch (`  
    `bgrt_sync_t * sync )`

Watch BGRT\_SYNC\_TOUCH.

**Warning**

For internal usage.

**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 )`

Watch BGRT\_SYNC\_WAIT.

**Warning**

For internal usage.

**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 )`

Watch BGRT\_SYNC\_WAKE.

**Warning**

For internal usage.

**5.15.4.11 bgrt\_sync\_init()** `bgrt_st_t bgrt_sync_init (`  
    `bgrt_sync_t * sync,`  
    `bgrt_prio_t prio )`

A basic synchronization primitive initiation.

## Parameters

<i>sync</i>	A sync pointer.
<i>prio</i>	A priority.

## 5.16 bugurtos/kernel/syscall.h File Reference

System call header.

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

## Data Structures

- struct [\\_bgrt\\_va\\_wr\\_t](#)

## Macros

- #define [BGRT\\_SC\\_ID](#)(syscall) [BGRT\\_CONCAT](#)([BGRT\\_SC\\_ENUM\\_](#), syscall)  
*Get system call id.*
- #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](#)))  
*System call service routine name.*
- #define [BGRT\\_SC\\_SR](#)(syscall, arg) [bgrt\\_st\\_t](#) [BGRT\\_SC\\_SR\\_NAME](#)(syscall)(arg)  
*System call service routine.*
- #define [BGRT\\_SYSCALL\\_N](#)(sc\_name, arg) [bgrt\\_syscall](#)([BGRT\\_SC\\_ID](#)(sc\_name), arg)  
*A system call macro, see [bgrt\\_syscall](#).*
- #define [BGRT\\_SYSCALL\\_NVAR](#)(sc\_name, ...) [bgrt\\_syscall\\_var](#)([BGRT\\_SC\\_ID](#)(sc\_name), [\\_\\_VA\\_ARGS\\_](#)↵  
↵)  
*A system call macro, see [bgrt\\_syscall\\_var](#).*

## Typedefs

- typedef enum [\\_bgrt\\_sc\\_enum](#) [bgrt\\_sc\\_enum](#)  
*System call IDs.*
- typedef [bgrt\\_st\\_t](#)(\* [bgrt\\_scsr\\_t](#)) (void \*)  
*System call service routine pointer.*
- typedef struct [\\_bgrt\\_va\\_wr\\_t](#) [bgrt\\_va\\_wr\\_t](#)  
*va\_list wrapper.*

## Enumerations

- enum [\\_bgrt\\_sc\\_enum](#) { [BGRT\\_SC\\_ENUM\\_END](#) }

## Functions

- `bgrt_st_t bgrt_priv_do_syscall` (`bgrt_syscall_t syscall_num, void *syscall_arg`)  
*System call processing routine.*
- `bgrt_st_t bgrt_syscall_var` (`bgrt_syscall_t num,...`)  
*A system call.*

### 5.16.1 Detailed Description

System call header.

#### Warning

This file content is internal usage!

### 5.16.2 Macro Definition Documentation

**5.16.2.1 BGRT\_SC\_ID** `#define BGRT_SC_ID(  
 syscall ) BGRT_CONCAT(BGRT_SC_ENUM_, syscall)`

Get system call id.

**5.16.2.2 BGRT\_SC\_SR** `#define BGRT_SC_SR(  
 syscall,  
 arg ) bgrt_st_t BGRT_SC_SR_NAME(syscall) (arg)`

System call service routine.

**5.16.2.3 BGRT\_SC\_SR\_NAME** `#define BGRT_SC_SR_NAME(  
 syscall ) BGRT_CONCAT2(BGRT_SC_, BGRT_CONCAT(syscall, _SR))`

System call service routine name.

**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)`

A system call macro, see `bgrt_syscall`.

#### Warning

For internal usage.



## Parameters

<i>sc_name</i>	A system call name.
<i>arg</i>	A system call argument pointer.

**5.16.2.6 BGRТ\_SYSCALL\_NVAR** `#define BGRТ_SYSCALL_NVAR(  
     sc_name,  
     ... ) bgrт_syscall_var(BGRТ_SC_ID(sc_name), __VA_ARGS__)`

A system call macro, see `bgrт_syscall_var`.

## Warning

For internal usage.

## Parameters

<i>sc_name</i>	A system call name.
----------------	---------------------

## 5.16.3 Typedef Documentation

**5.16.3.1 bgrт\_sc\_enum** `typedef enum _bgrт_sc_enum bgrт_sc_enum`

System call IDs.

**5.16.3.2 bgrт\_scsr\_t** `typedef bgrт_st_t(* bgrт_scsr_t) (void *)`

System call service routine pointer.

**5.16.3.3 bgrт\_va\_wr\_t** `typedef struct _bgrт_va_wr_t bgrт_va_wr_t`

va\_list wrapper.

## 5.16.4 Enumeration Type Documentation

**5.16.4.1 \_bgrт\_sc\_enum** `enum _bgrт_sc_enum`

**Enumerator**

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

**5.16.5 Function Documentation**

**5.16.5.1 bgrt\_priv\_do\_syscall()** `bgrt_st_t bgrt_priv_do_syscall (`  
    `bgrt_syscall_t syscall_num,`  
    `void * syscall_arg )`

System call processing routine.

This function calls system call handlers and passes arguments to them.

**Parameters**

<i>syscall_num</i>	System call number.
<i>syscall_arg</i>	System call argument.

**Returns**

System call execution status.

**5.16.5.2 bgrt\_syscall\_var()** `bgrt_st_t bgrt_syscall_var (`  
    `bgrt_syscall_t num,`  
    `... )`

A system call.

This function switches a processor core from a process context to the kernel context. The kernel code is always run in the kernel context. This is done to save memory in process stacks. A system calls are done on every operations with processes, mutexes, semaphores and signals. The Kernel does all of this job.

**Warning**

Internal usage function.

**Parameters**

<i>num</i>	a number of a system call (what is going to be done).
------------	---

## 5.17 bugurtos/kernel/timer.h File Reference

A software timer headers.

### Data Structures

- struct [bgrt\\_priv\\_ktimer\\_t](#)

### Macros

- #define [BGRT\\_CLEAR\\_TIMER](#)(t) [bgrt\\_priv\\_clear\\_timer](#)((bgrt\_tmr\_t \*)&t)  
*Reset software timer.*
- #define [BGRT\\_SET\\_TIMER](#)(t, s) (t += s)  
*Set software timer.*
- #define [BGRT\\_TIMER](#)(t) (bgrt\_tmr\_t)bgrt\_priv\_timer((bgrt\_tmr\_t)t)  
*Get software timer value.*
- #define [BGRT\\_WAIT\\_INTERVAL](#)(tmr, time) ([bgrt\\_priv\\_wait\\_interval](#)(&tmr, time))  
*The caller waits until system time becomes greater or equal than tmr + time.*

### Typedefs

- typedef struct [bgrt\\_priv\\_ktimer\\_t](#) [bgrt\\_ktimer\\_t](#)

### Functions

- void [bgrt\\_wait\\_time](#) (bgrt\_tmr\_t time)  
*Wait for certain time.*
- void [bgrt\\_priv\\_clear\\_timer](#) (bgrt\_tmr\_t \*t)  
*Clear software timer.*
- bgrt\_tmr\_t [bgrt\\_priv\\_timer](#) (bgrt\_tmr\_t t)  
*Get software timer value.*
- void [bgrt\\_priv\\_wait\\_interval](#) (bgrt\_tmr\_t \*tmr, bgrt\_tmr\_t time)  
*The caller waits until system time becomes greater or equal than \*tmr + time.*

#### 5.17.1 Detailed Description

A software timer headers.

Software timers used for time-process synchronization.

#### Warning

Software timers can not be used for precision time interval measurement!

#### 5.17.2 Macro Definition Documentation

**5.17.2.1 BGRT\_CLEAR\_TIMER** #define BGRT\_CLEAR\_TIMER(  
t ) [bgrt\\_priv\\_clear\\_timer](#)((bgrt\_tmr\_t \*)&t)

Reset software timer.

**Parameters**

<i>t</i>	A timer variable name.
----------	------------------------

**5.17.2.2 BGRT\_SET\_TIMER** #define BGRT\_SET\_TIMER(  

```
    t,  
    s ) ( t += s )
```

Set software timer.

May be used instead of BGRT\_CLEAR\_TIMER for periodic code execution. Software timers will have deterministic behavior. Software timers with the same period will work synchronously.

**Parameters**

<i>t</i>	A timer variable name.
<i>s</i>	A timer step.

**5.17.2.3 BGRT\_TIMER** #define BGRT\_TIMER(  

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

Get software timer value.

**Parameters**

<i>t</i>	Software timer value.
----------	-----------------------

**5.17.2.4 BGRT\_WAIT\_INTERVAL** #define BGRT\_WAIT\_INTERVAL(  

```
    tmr,  
    time ) ( bgrt_priv_wait_interval( &tmr, time ) )
```

The caller waits until system time becomes greater or equal than tmr + time.

**Parameters**

<i>tmr</i>	A timer variable name.
<i>time</i>	Wait time interval.

**5.17.3 Typedef Documentation**

**5.17.3.1 bgrt\_ktimer\_t** typedef struct bgrt\_priv\_ktimer\_t bgrt\_ktimer\_t

The system timer (used by the kernel to count ticks).

## 5.17.4 Function Documentation

**5.17.4.1 bgrt\_priv\_clear\_timer()** void bgrt\_priv\_clear\_timer (  
bgrt\_tmr\_t \* t )

Clear software timer.

### Warning

For internal usage.

### Parameters

<i>t</i>	A pointer to a timer.
----------	-----------------------

**5.17.4.2 bgrt\_priv\_timer()** bgrt\_tmr\_t bgrt\_priv\_timer (  
bgrt\_tmr\_t t )

Get software timer value.

### Warning

For internal usage.

### Parameters

<i>t</i>	A timer value.
----------	----------------

**5.17.4.3 bgrt\_priv\_wait\_interval()** void bgrt\_priv\_wait\_interval (  
bgrt\_tmr\_t \* tmr,  
bgrt\_tmr\_t time )

The caller waits until system time becomes greater or equal than \*tmr + time.

### Warning

For internal usage.

**Parameters**

<i>tmr</i>	A pointer to a timer.
<i>time</i>	Wait time interval.

**5.17.4.4 bgrt\_wait\_time()** `void bgrt_wait_time ( bgrt_tmr_t time )`

Wait for certain time.

Caller process spins in a loop for a time.

**Parameters**

<i>time</i>	Wait time.
-------------	------------

## 5.18 bugurtos/kernel/vint.h File Reference

A virtual interrupt header.

**Data Structures**

- struct [bgrt\\_priv\\_vint\\_t](#)  
*A virtual interrupt.*
- struct [bgrt\\_priv\\_vic\\_t](#)  
*A virtual interrupt controller.*

**Macros**

- #define [BGRT\\_VINT\\_CS\\_START\(\)](#) [BGRT\\_INT\\_LOCK\(\)](#)
- #define [BGRT\\_VINT\\_CS\\_END\(\)](#) [BGRT\\_INT\\_FREE\(\)](#)

**Typedefs**

- typedef struct [bgrt\\_priv\\_vint\\_t](#) [bgrt\\_vint\\_t](#)
- typedef struct [bgrt\\_priv\\_vic\\_t](#) [bgrt\\_vic\\_t](#)

## Functions

- void `bgrt_vint_init` (`bgrt_vint_t` \*vint, `bgrt_prio_t` prio, `bgrt_code_t` func, void \*arg)  
*A `bgrt_vint_t` object initiation.*
- `bgrt_st_t` `bgrt_vint_push_isr` (`bgrt_vint_t` \*vint, `bgrt_vic_t` \*vic)  
*Insert `bgrt_vint_t` object to `bgrt_vic_t` container for processing (for ISR usage).*
- `bgrt_st_t` `bgrt_vint_push` (`bgrt_vint_t` \*vint, `bgrt_vic_t` \*vic)  
*Insert `bgrt_vint_t` object to `bgrt_xlist_t` container for processing.*
- void `bgrt_vic_init` (`bgrt_vic_t` \*vic)  
*Virtual interrupt controller initialization.*
- `bgrt_st_t` `bgrt_vic_iterator` (`bgrt_vic_t` \*vic)  
*Virtual interrupt processing.*
- void `bgrt_vic_do_work` (`bgrt_vic_t` \*vic)  
*Virtual interrupt processing.*

### 5.18.1 Detailed Description

A virtual interrupt header.

### 5.18.2 Macro Definition Documentation

**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 Typedef Documentation

**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 Function Documentation

**5.18.4.1 bgrt\_vic\_do\_work()** `void bgrt_vic_do_work (`  
`bgrt_vic_t * vic )`

Virtual interrupt processing.

#### Warning

For internal usage.

**Parameters**

<i>vic</i>	A pointer to a <code>bgrt_vic_t</code> object.
------------	--

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

Virtual interrupt controller initialization.

**Warning**

For internal usage.

**Parameters**

<i>vic</i>	A pointer to a <code>bgrt_vic_t</code> object.
------------	--

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

Virtual interrupt processing.

**Warning**

For internal usage.

**Parameters**

<i>vic</i>	A pointer to a <code>bgrt_vic_t</code> object.
------------	--

**Returns**

BGRT\_ST\_ROLL if next iteration needed, BGRT\_ST\_OK if all done.

**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 )`

A `bgrt_vint_t` object initiation.



**Warning**

For internal usage.

**Parameters**

<i>vint</i>	A <code>bgrt_vint_t</code> pointer.
<i>prio</i>	A priority.
<i>func</i>	An ISR pointer.
<i>arg</i>	An ISR arg pointer.

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

Insert `bgrt_vint_t` object to `bgrt_xlist_t` container for processing.

**Warning**

For internal usage.

**Parameters**

<i>vint</i>	A <code>bgrt_vint_t</code> pointer.
<i>vic</i>	A pointer to destination <code>bgrt_vic_t</code> object.

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

Insert `bgrt_vint_t` object to `bgrt_vic_t` container for processing (for ISR usage).

**Warning**

For internal usage.

**Parameters**

<i>vint</i>	A <code>bgrt_vint_t</code> pointer.
<i>vic</i>	A pointer to destination <code>bgrt_vic_t</code> object.

## 5.19 bgrtos/kernel/xlist.h File Reference

A prioritized list header.

### Data Structures

- struct [bgrt\\_priv\\_xlist\\_t](#)  
*A prioritized list.*

### Typedefs

- typedef struct [bgrt\\_priv\\_xlist\\_t](#) [bgrt\\_xlist\\_t](#)

### Functions

- void [bgrt\\_xlist\\_init](#) ([bgrt\\_xlist\\_t](#) \*xlist)  
*An [bgrt\\_xlist\\_t](#) object initiation.*
- [bgrt\\_item\\_t](#) \* [bgrt\\_xlist\\_head](#) ([bgrt\\_xlist\\_t](#) \*xlist)  
*List head search.*
- void [bgrt\\_xlist\\_switch](#) ([bgrt\\_xlist\\_t](#) \*xlist, [bgrt\\_prio\\_t](#) prio)  
*Switch a head pointer.*

#### 5.19.1 Detailed Description

A prioritized list header.

#### 5.19.2 Typedef Documentation

**5.19.2.1 [bgrt\\_xlist\\_t](#)** typedef struct [bgrt\\_priv\\_xlist\\_t](#) [bgrt\\_xlist\\_t](#)

See [bgrt\\_priv\\_xlist\\_t](#);

#### 5.19.3 Function Documentation

**5.19.3.1 [bgrt\\_xlist\\_head\(\)](#)** [bgrt\\_item\\_t](#)\* [bgrt\\_xlist\\_head](#) (  
[bgrt\\_xlist\\_t](#) \* *xlist* )

List head search.

#### Warning

For internal usage.

## Parameters

<i>xlist</i>	An <code>bgrt_xlist_t</code> pointer.
--------------	---------------------------------------

## Returns

The head pointer, which is the most prioritized pointer in the list head pointer array.

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

An `bgrt_xlist_t` object initiation.

## Warning

For internal usage.

## Parameters

<i>xlist</i>	An <code>bgrt_xlist_t</code> pointer.
--------------	---------------------------------------

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

Switch a head pointer.

Does `xlist->item[prio] = xlist->item[prio]->next`.

## Warning

For internal usage.

## Parameters

<i>xlist</i>	An <code>bgrt_xlist_t</code> pointer.
<i>prio</i>	A priority to switch.



## Index

- [\\_bgrt\\_sc\\_enum](#)
    - [syscall.h, 85](#)
  - [\\_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, 17](#)
  - [BGRT\\_ATM\\_BGET\\_ISR, 17](#)
  - [BGRT\\_ATM\\_BSET\\_ISR, 17](#)
  - [BGRT\\_ATM\\_INIT\\_ISR, 17](#)
  - [BGRT\\_VINT\\_PUSH\\_ISR, 17](#)
- [atm\\_gen\\_1.h](#)
  - [BGRT\\_ATM\\_BCLR\\_ISR, 18](#)
  - [BGRT\\_ATM\\_BGET\\_ISR, 18](#)
  - [BGRT\\_ATM\\_BSET\\_ISR, 18](#)
  - [BGRT\\_ATM\\_INIT\\_ISR, 18](#)
  - [BGRT\\_VINT\\_PUSH\\_ISR, 18](#)
- [base\\_prio](#)
  - [bgrt\\_priv\\_proc\\_t, 10](#)
- [BGRT\\_ASSERT](#)
  - [bugurt.h, 25](#)
- [bgrt\\_atm\\_bclr](#)
  - [bugurt\\_port.h, 21](#)
- [BGRT\\_ATM\\_BCLR\\_ISR](#)
  - [atm\\_cortex\\_m34\\_1.h, 17](#)
  - [atm\\_gen\\_1.h, 18](#)
  - [bugurt\\_port.h, 19](#)
- [bgrt\\_atm\\_bget](#)
  - [bugurt\\_port.h, 22](#)
- [BGRT\\_ATM\\_BGET\\_ISR](#)
  - [atm\\_cortex\\_m34\\_1.h, 17](#)
  - [atm\\_gen\\_1.h, 18](#)
  - [bugurt\\_port.h, 20](#)
- [bgrt\\_atm\\_bset](#)
  - [bugurt\\_port.h, 22](#)
- [BGRT\\_ATM\\_BSET\\_ISR](#)
  - [atm\\_cortex\\_m34\\_1.h, 17](#)
  - [atm\\_gen\\_1.h, 18](#)
  - [bugurt\\_port.h, 20](#)
- [bgrt\\_atm\\_init](#)
  - [bugurt\\_port.h, 23](#)
- [BGRT\\_ATM\\_INIT\\_ISR](#)
  - [atm\\_cortex\\_m34\\_1.h, 17](#)
  - [atm\\_gen\\_1.h, 18](#)
  - [bugurt\\_port.h, 20](#)
- [BGRT\\_CDECL\\_BEGIN](#)
  - [bugurt.h, 25](#)
- [BGRT\\_CDECL\\_END](#)
  - [bugurt.h, 25](#)
- [BGRT\\_CLEAR\\_TIMER](#)
  - [timer.h, 87](#)
- [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, 28](#)
- [BGRT\\_CONCAT](#)
  - [bugurt.h, 25](#)
- [BGRT\\_CONCAT2](#)
  - [bugurt.h, 25](#)
- [BGRT\\_CONCAT3](#)
  - [bugurt.h, 25](#)
- [BGRT\\_CRIT\\_SEC\\_ENTER](#)
  - [crit\\_sec.h, 34](#)
- [BGRT\\_CRIT\\_SEC\\_EXIT](#)
  - [crit\\_sec.h, 34](#)
- [bgrt\\_curr\\_cpu](#)
  - [bugurt.h, 28](#)
- [BGRT\\_CURR\\_PROC](#)
  - [bugurt\\_port.h, 21](#)
- [bgrt\\_curr\\_proc](#)
  - [bugurt.h, 28](#)
- [BGRT\\_GET\\_USPD](#)
  - [proc.h, 62](#)
- [bgrt\\_init](#)
  - [bugurt.h, 28](#)
- [BGRT\\_INT\\_FREE](#)
  - [bugurt\\_port.h, 21](#)
- [BGRT\\_INT\\_LOCK](#)
  - [bugurt\\_port.h, 21](#)
- [BGRT\\_ISR](#)
  - [bugurt\\_port.h, 21](#)
- [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, 21  
 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, 26  
 bgrt\_kernel\_t  
     kernel.h, 49  
 bgrt\_kstat\_t  
     sched.h, 76  
 bgrt\_ktimer\_t  
     timer.h, 88  
 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, 26  
 bgrt\_priv\_clear\_timer  
     timer.h, 89  
 bgrt\_priv\_crit\_sec\_enter  
     crit\_sec.h, 34  
 bgrt\_priv\_crit\_sec\_exit  
     crit\_sec.h, 35  
 bgrt\_priv\_do\_syscall  
     syscall.h, 86  
 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, 6  
     lock, 6  
     val, 6  
 bgrt\_priv\_ktimer\_t, 6  
     lock, 7  
     tick, 7  
     val, 7  
 bgrt\_priv\_pcounter\_t, 7  
     counter, 7  
     map, 7  
 bgrt\_priv\_pitem\_t, 8  
     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, 69  
 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, 9
  - affinity, 9
  - arg, 9
  - base\_prio, 10
  - cnt\_lock, 10
  - core\_id, 10
  - flags, 10
  - lock, 10
  - lres, 10
  - parent, 10
  - pmain, 10
  - rs\_hook, 10
  - spointer, 10
  - sstart, 10
  - sv\_hook, 11
  - sync, 11
  - time\_quant, 11
  - timer, 11
  - udata, 11
- bgrt\_priv\_proc\_terminate
  - proc.h, 73
- bgrt\_priv\_sched\_proc\_set\_core
  - sched.h, 76
- bgrt\_priv\_sched\_proc\_yield
  - sched.h, 76
- bgrt\_priv\_sched\_t, 11
  - current\_proc, 12
  - expired, 12
  - lock, 12
  - nested\_crit\_sec, 12
  - plst, 12
  - ready, 12
- bgrt\_priv\_sync\_get\_owner
  - sync.h, 80
- BGRT\_PRIV\_SYNC\_INIT
  - sync.h, 80
- bgrt\_priv\_sync\_init
  - sync.h, 80
- bgrt\_priv\_sync\_own
  - sync.h, 81
- bgrt\_priv\_sync\_prio
  - sync.h, 81
- bgrt\_priv\_sync\_proc\_timeout
  - sync.h, 81
- bgrt\_priv\_sync\_set\_owner
  - sync.h, 81
- bgrt\_priv\_sync\_sleep
  - sync.h, 81
- bgrt\_priv\_sync\_t, 12
  - dirty, 13
  - lock, 13
  - owner, 13
  - prio, 13
  - pwake, 13
  - sleep, 13
  - snum, 13
- bgrt\_priv\_sync\_touch
  - sync.h, 82
- bgrt\_priv\_sync\_wait
  - sync.h, 82
- bgrt\_priv\_sync\_wake
  - sync.h, 82
- bgrt\_priv\_timer
  - timer.h, 89
- bgrt\_priv\_uspd\_t, 14
  - scarg, 14
  - scnum, 14
  - scret, 14
- bgrt\_priv\_vic\_t, 14
  - list, 15
  - prio, 15
- bgrt\_priv\_vint\_t, 15
  - arg, 15
  - func, 15
  - parent, 15
- bgrt\_priv\_wait\_interval
  - timer.h, 89
- bgrt\_priv\_xlist\_t, 16
  - 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, 37
- 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, 38  
 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, 29  
 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, 67  
 BGRT\_PROC\_STATE\_WD\_STOPPED  
   proc.h, 68  
 BGRT\_PROC\_STOP  
   syscall\_api.h, 39  
 bgrt\_proc\_t  
   proc.h, 69  
 bgrt\_proc\_terminate  
   proc.h, 74  
 BGRT\_PROC\_TO\_PID  
   proc.h, 68  
 bgrt\_resched  
   bugurt.h, 29  
 BGRT\_RESCHED\_PROC  
   bugurt.h, 26  
 bgrt\_sc\_enum  
   syscall.h, 85  
 BGRT\_SC\_ENUM\_END  
   syscall.h, 86  
 BGRT\_SC\_ID  
   syscall.h, 84  
 BGRT\_SC\_SR  
   syscall.h, 84  
   syscall\_routines.h, 43–45  
 BGRT\_SC\_SR\_NAME  
   syscall.h, 84  
 BGRT\_SC\_TBL\_ENTRY  
   syscall.h, 84  
 bgrt\_sched\_highest\_load\_core  
   sched.h, 76  
 bgrt\_sched\_init  
   sched.h, 77  
 bgrt\_sched\_lazy\_local\_load\_balancer  
   sched.h, 77  
 bgrt\_sched\_load\_balancer  
   sched.h, 77  
 bgrt\_sched\_proc\_run  
   sched.h, 78  
 BGRT\_SCHED\_PROC\_SET\_CORE  
   sched.h, 75  
 bgrt\_sched\_proc\_stop  
   sched.h, 78  
 bgrt\_sched\_proc\_yield  
   sched.h, 78  
 bgrt\_sched\_run  
   sched.h, 78  
 bgrt\_sched\_t  
   sched.h, 76  
 bgrt\_scsr\_t  
   syscall.h, 85  
 BGRT\_SET\_TIMER  
   timer.h, 88  
 BGRT\_SPIN\_FREE  
   bugurt.h, 26



bgrt\_spin\_free  
bugurt.h, 30

BGRT\_SPIN\_INIT  
bugurt.h, 26

bgrt\_spin\_init  
bugurt.h, 30

BGRT\_SPIN\_LOCK  
bugurt.h, 26

bgrt\_spin\_lock  
bugurt.h, 30

BGRT\_ST\_EAGAIN  
bugurt.h, 26

BGRT\_ST\_EEMPTY  
bugurt.h, 26

BGRT\_ST\_ENULL  
bugurt.h, 27

BGRT\_ST\_EOWN  
bugurt.h, 27

BGRT\_ST\_ESTAT  
bugurt.h, 27

BGRT\_ST\_ESYNC  
bugurt.h, 27

BGRT\_ST\_ETIMEOUT  
bugurt.h, 27

BGRT\_ST\_IDLE  
bugurt.h, 27

BGRT\_ST\_OK  
bugurt.h, 27

BGRT\_ST\_ROLL  
bugurt.h, 27

BGRT\_ST\_SCALL  
bugurt.h, 28

bgrt\_start  
bugurt.h, 30

bgrt\_stat\_calc\_load  
bugurt.h, 31

bgrt\_stat\_dec  
bugurt.h, 31

bgrt\_stat\_inc  
bugurt.h, 31

bgrt\_stat\_init  
bugurt.h, 32

bgrt\_stat\_merge  
bugurt.h, 32

bgrt\_switch\_to\_proc  
bugurt.h, 33

BGRT\_SYNC\_GET\_OWNER  
syscall\_api.h, 39

BGRT\_SYNC\_INIT  
sync.h, 80

bgrt\_sync\_init  
sync.h, 82

BGRT\_SYNC\_OWN  
syscall\_api.h, 39

BGRT\_SYNC\_PRIO  
sync.h, 80

BGRT\_SYNC\_PROC\_TIMEOUT  
syscall\_api.h, 40

BGRT\_SYNC\_SET\_OWNER  
syscall\_api.h, 40

BGRT\_SYNC\_SLEEP  
syscall\_api.h, 40

bgrt\_sync\_t  
sync.h, 80

BGRT\_SYNC\_TOUCH  
syscall\_api.h, 41

BGRT\_SYNC\_WAIT  
syscall\_api.h, 41

BGRT\_SYNC\_WAKE  
syscall\_api.h, 41

bgrt\_syscall  
bugurt.h, 33

BGRT\_SYSCALL\_N  
syscall.h, 84

BGRT\_SYSCALL\_NVAR  
syscall.h, 85

bgrt\_syscall\_var  
syscall.h, 86

BGRT\_TIMER  
timer.h, 88

bgrt\_user\_func\_t  
syscall\_routines.h, 43

BGRT\_USPD\_INIT  
proc.h, 68

BGRT\_USPD\_PROC\_T  
proc.h, 68

BGRT\_USPD\_T  
proc.h, 68

bgrt\_va\_wr\_t  
syscall.h, 85

bgrt\_vic\_do\_work  
vint.h, 91

bgrt\_vic\_init  
vint.h, 92

bgrt\_vic\_iterator  
vint.h, 92

bgrt\_vic\_t  
vint.h, 91

BGRT\_VINT\_CS\_END  
vint.h, 91

BGRT\_VINT\_CS\_START  
vint.h, 91

bgrt\_vint\_init  
vint.h, 92

bgrt\_vint\_push  
vint.h, 93

BGRT\_VINT\_PUSH\_ISR  
atm\_cortex\_m34\_1.h, 17  
atm\_gen\_1.h, 18

bgrt\_vint\_push\_isr  
vint.h, 93

bgrt\_vint\_t  
vint.h, 91

BGRT\_WAIT\_INTERVAL  
timer.h, 88

bgrt\_wait\_time

- timer.h, 90
- bgrt\_xlist\_head
  - xlist.h, 94
- bgrt\_xlist\_init
  - xlist.h, 95
- bgrt\_xlist\_switch
  - xlist.h, 95
- bgrt\_xlist\_t
  - xlist.h, 94
- bugurt.h
  - BGRT\_ASSERT, 25
  - BGRT\_CDECL\_BEGIN, 25
  - BGRT\_CDECL\_END, 25
  - bgrt\_code\_t, 28
  - BGRT\_CONCAT, 25
  - BGRT\_CONCAT2, 25
  - BGRT\_CONCAT3, 25
  - bgrt\_curr\_cpu, 28
  - bgrt\_curr\_proc, 28
  - bgrt\_init, 28
  - BGRT\_KERNEL\_PREEMPT, 26
  - BGRT\_PRIO\_LOWEST, 26
  - bgrt\_proc\_stack\_init, 29
  - bgrt\_resched, 29
  - BGRT\_RESCHED\_PROC, 26
  - BGRT\_SPIN\_FREE, 26
  - bgrt\_spin\_free, 30
  - BGRT\_SPIN\_INIT, 26
  - bgrt\_spin\_init, 30
  - BGRT\_SPIN\_LOCK, 26
  - bgrt\_spin\_lock, 30
  - BGRT\_ST\_EAGAIN, 26
  - BGRT\_ST\_EEMPTY, 26
  - BGRT\_ST\_ENULL, 27
  - BGRT\_ST\_EOWN, 27
  - BGRT\_ST\_ESTAT, 27
  - BGRT\_ST\_ESYNC, 27
  - BGRT\_ST\_ETIMEOUT, 27
  - BGRT\_ST\_IDLE, 27
  - BGRT\_ST\_OK, 27
  - BGRT\_ST\_ROLL, 27
  - BGRT\_ST\_SCALL, 28
  - bgrt\_start, 30
  - bgrt\_stat\_calc\_load, 31
  - bgrt\_stat\_dec, 31
  - bgrt\_stat\_inc, 31
  - bgrt\_stat\_init, 32
  - bgrt\_stat\_merge, 32
  - bgrt\_switch\_to\_proc, 33
  - bgrt\_syscall, 33
- bugurt\_port.h
  - bgrt\_atm\_bclr, 21
  - BGRT\_ATM\_BCLR\_ISR, 19
  - bgrt\_atm\_bget, 22
  - BGRT\_ATM\_BGET\_ISR, 20
  - bgrt\_atm\_bset, 22
  - BGRT\_ATM\_BSET\_ISR, 20
  - bgrt\_atm\_init, 23
  - BGRT\_ATM\_INIT\_ISR, 20
  - BGRT\_CURR\_PROC, 21
  - BGRT\_INT\_FREE, 21
  - BGRT\_INT\_LOCK, 21
  - BGRT\_ISR, 21
  - BGRT\_KBLOCK, 21
- bugurtos/arch/common/atm\_cortex\_m34\_1.h, 17
- bugurtos/arch/common/atm\_gen\_1.h, 18
- bugurtos/doc/doxygen/bugurt\_port.h, 19
- bugurtos/kernel/bugurt.h, 23
- bugurtos/kernel/crit\_sec.h, 33
- bugurtos/kernel/default/syscall\_api.h, 35
- 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, 74
- bugurtos/kernel/sync.h, 79
- bugurtos/kernel/syscall.h, 83
- bugurtos/kernel/timer.h, 87
- bugurtos/kernel/vint.h, 90
- bugurtos/kernel/xlist.h, 94
- cnt\_lock
  - bgrt\_priv\_proc\_t, 10
- core\_id
  - bgrt\_priv\_proc\_t, 10
- counter
  - bgrt\_priv\_pcounter\_t, 7
- crit\_sec.h
  - BGRT\_CRIT\_SEC\_ENTER, 34
  - BGRT\_CRIT\_SEC\_EXIT, 34
  - bgrt\_priv\_crit\_sec\_enter, 34
  - bgrt\_priv\_crit\_sec\_exit, 35
- current\_proc
  - bgrt\_priv\_sched\_t, 12
- dirty
  - bgrt\_priv\_sync\_t, 13
- expired
  - bgrt\_priv\_sched\_t, 12
- flags
  - bgrt\_priv\_proc\_t, 10
- 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](#), [15](#)
- lock
  - [bgrt\\_priv\\_kstat\\_t](#), [6](#)
  - [bgrt\\_priv\\_ktimer\\_t](#), [7](#)
  - [bgrt\\_priv\\_proc\\_t](#), [10](#)
  - [bgrt\\_priv\\_sched\\_t](#), [12](#)
  - [bgrt\\_priv\\_sync\\_t](#), [13](#)
- lpmmap
  - [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](#), [12](#)
- 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](#), [15](#)
- 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](#), [69](#)
  - [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](#), [73](#)
  - [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, 67
    - 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, 68
  - pwake
    - bgrt\_priv\_sync\_t, 13
  - ready
    - bgrt\_priv\_sched\_t, 12
  - rs\_hook
    - bgrt\_priv\_proc\_t, 10
  - scarg
    - bgrt\_priv\_uspd\_t, 14
  - sched
    - bgrt\_priv\_kernel\_t, 5
  - sched.h
    - bgrt\_kstat\_t, 76
    - bgrt\_priv\_sched\_proc\_set\_core, 76
    - bgrt\_priv\_sched\_proc\_yield, 76
    - bgrt\_sched\_highest\_load\_core, 76
    - bgrt\_sched\_init, 77
    - bgrt\_sched\_lazy\_local\_load\_balancer, 77
    - bgrt\_sched\_load\_balancer, 77
    - bgrt\_sched\_proc\_run, 78
    - BGRT\_SCHED\_PROC\_SET\_CORE, 75
    - bgrt\_sched\_proc\_stop, 78
    - bgrt\_sched\_proc\_yield, 78
    - bgrt\_sched\_run, 78
    - 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, 11
  - sync
    - bgrt\_priv\_proc\_t, 11
  - sync.h
    - bgrt\_priv\_sync\_get\_owner, 80
    - BGRT\_PRIV\_SYNC\_INIT, 80
    - bgrt\_priv\_sync\_init, 80
    - bgrt\_priv\_sync\_own, 81
    - bgrt\_priv\_sync\_prio, 81
    - bgrt\_priv\_sync\_proc\_timeout, 81
    - bgrt\_priv\_sync\_set\_owner, 81
    - bgrt\_priv\_sync\_sleep, 81
    - bgrt\_priv\_sync\_touch, 82
    - bgrt\_priv\_sync\_wait, 82
    - bgrt\_priv\_sync\_wake, 82
    - BGRT\_SYNC\_INIT, 80
    - bgrt\_sync\_init, 82
    - BGRT\_SYNC\_PRIO, 80
    - bgrt\_sync\_t, 80
  - syscall.h
    - \_bgrt\_sc\_enum, 85
    - bgrt\_priv\_do\_syscall, 86
    - bgrt\_sc\_enum, 85
    - BGRT\_SC\_ENUM\_END, 86
    - BGRT\_SC\_ID, 84
    - BGRT\_SC\_SR, 84
    - BGRT\_SC\_SR\_NAME, 84
    - BGRT\_SC\_TBL\_ENTRY, 84
    - bgrt\_scsr\_t, 85
    - BGRT\_SYSCALL\_N, 84
    - BGRT\_SYSCALL\_NVAR, 85
    - bgrt\_syscall\_var, 86
    - bgrt\_va\_wr\_t, 85
  - syscall\_api.h
    - BGRT\_PROC\_FREE, 36
    - BGRT\_PROC\_GET\_ID, 36
    - BGRT\_PROC\_GET\_PRIO, 37
    - BGRT\_PROC\_LOCK, 37
    - BGRT\_PROC\_RESET\_WATCHDOG, 37
    - BGRT\_PROC\_RESTART, 37
    - BGRT\_PROC\_RUN, 38
    - BGRT\_PROC\_SELF\_STOP, 38
    - BGRT\_PROC\_SET\_PRIO, 38
    - BGRT\_PROC\_STOP, 39
    - BGRT\_SYNC\_GET\_OWNER, 39
    - BGRT\_SYNC\_OWN, 39
    - BGRT\_SYNC\_PROC\_TIMEOUT, 40
    - BGRT\_SYNC\_SET\_OWNER, 40
    - BGRT\_SYNC\_SLEEP, 40

- BGRT\_SYNC\_TOUCH, [41](#)
- BGRT\_SYNC\_WAIT, [41](#)
- BGRT\_SYNC\_WAKE, [41](#)
- syscall\_routines.h
  - BGRT\_SC\_SR, [43–45](#)
  - bgrt\_user\_func\_t, [43](#)
- tick
  - bgrt\_priv\_ktimer\_t, [7](#)
- time\_quant
  - bgrt\_priv\_proc\_t, [11](#)
- timer
  - bgrt\_priv\_kernel\_t, [5](#)
  - bgrt\_priv\_proc\_t, [11](#)
- timer.h
  - BGRT\_CLEAR\_TIMER, [87](#)
  - bgrt\_ktimer\_t, [88](#)
  - bgrt\_priv\_clear\_timer, [89](#)
  - bgrt\_priv\_timer, [89](#)
  - bgrt\_priv\_wait\_interval, [89](#)
  - BGRT\_SET\_TIMER, [88](#)
  - BGRT\_TIMER, [88](#)
  - BGRT\_WAIT\_INTERVAL, [88](#)
  - bgrt\_wait\_time, [90](#)
- udata
  - bgrt\_priv\_proc\_t, [11](#)
- val
  - bgrt\_priv\_kstat\_t, [6](#)
  - bgrt\_priv\_ktimer\_t, [7](#)
- vint.h
  - bgrt\_vic\_do\_work, [91](#)
  - bgrt\_vic\_init, [92](#)
  - bgrt\_vic\_iterator, [92](#)
  - bgrt\_vic\_t, [91](#)
  - BGRT\_VINT\_CS\_END, [91](#)
  - BGRT\_VINT\_CS\_START, [91](#)
  - bgrt\_vint\_init, [92](#)
  - bgrt\_vint\_push, [93](#)
  - bgrt\_vint\_push\_isr, [93](#)
  - bgrt\_vint\_t, [91](#)
- xlist.h
  - bgrt\_xlist\_head, [94](#)
  - bgrt\_xlist\_init, [95](#)
  - bgrt\_xlist\_switch, [95](#)
  - bgrt\_xlist\_t, [94](#)