BuguRTOS

4.1.0

Generated by Doxygen 1.8.17

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:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

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 hpmap
- 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 hpmap `bgrt_map_t bgrt_priv_kblock_t::hpmap`

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_C↩ ST)
- #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), __ATO↩ MIC_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

map_ptr	A pointer to atomic map.
msk	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

map_ptr	A pointer to atomic map.
msk	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

map_ptr	A pointer to atomic map.
msk	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

map_ptr	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

v	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

map_ptr	A pointer to atomic map.
msk	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

map_ptr	A pointer to atomic map.
msk	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

map_ptr	A pointer to atomic map.
msk	A mask.

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

Atomic map initialization.

Parameters

map_ptr	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 itteration 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 itteration 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

sstart	a process stack bottom.
pmain	a pointer to a function to call.
arg	an argument to a function to call.
return_address	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

core_id	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

lock	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

lock	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

lock	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

prio	a priority of a process for which we want to compute a load.
stat	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

proc	a pointer to a process.
stat	a pointer to a bgrt_ls_t 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

proc	a pointer to a process.
stat	a pointer to a bgrt_ls_t 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

stat	a pointer to a bgrt_ls_t 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

src_stat	a pointer to a signal statistic structure.
dst_stat	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

num	a number of a system call (what is going to be done).
arg	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

core	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(S↩ YNC_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

pid_ptr	- A process ID read buffer pointer.

Returns

- A system call result.

5.6.2.3 BGRT_PROC_GET_PRIO

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

Get a priority of a process.

Parameters

pid	- A process ID.
pri_ptr	- A priority read buffer pointer

Returns

- A system call result.

5.6.2.4 BGRT_PROC_LOCK

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

Set BGRT_PROC_FLG_LOCK for caller process.

5.6.2.5 BGRT_PROC_RESET_WATCHDOG

```
#define BGRT_PROC_RESET_WATCHDOG( ) BGRT_SYSCALL_N(P↩
ROC_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 BGRT_PROC_RESTART

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

A process restart routine.

This function reinitializes a process and schedules it if possible.

Parameters

pid	- 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

pid	- 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

pid	- A process ID.
prio	- 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

pid	- 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

sync_ptr	- A pointer to the object of interest.
pid_ptr	- 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

sync_ptr	A pointer to the object of interest.
touch	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

pid	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

sync_ptr	A pointer to the object of interest.
pid	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

sync_ptr	A pointer to the object of interest.
touch_ptr	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

sync_ptr	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

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

sync_ptr	A bgrt_sync_t object pointer.
pid	A process ID, that is supposed to wake up. If BGRT_PID_NOTHING, then try to wake up wait list head.
chown	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

map	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

a	An bgrt_item_t 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

item	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

item	An bgrt_item_t 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

item	A pointer to an item.
head	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

kblock	A bgrt_kblock_t object pointer.

5.10.4.2 bgrt_kblock_init() `void bgrt_kblock_init (`

`bgrt_kblock_t * kblock )`

A bgrt_kblock_t object initialization.

Parameters

kblock	A bgrt_kblock_t object pointer.

5.10.4.3 bgrt_kblock_main() `void bgrt_kblock_main (`

`bgrt_kblock_t * kblock )`

A kernel thread main function.

Parameters

kblock	A bgrt_kblock_t 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

a	Current counter value.
b	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

val	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

val	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

a	Current counter value.
b	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

pcounter	A bgrt_pcounter_t pointer.
prio	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

pcounter	A bgrt_pcounter_t pointer.
prio	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

pcounter	A bgrt_pcounter_t 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

pcounter	A bgrt_pcounter_t pointer.
prio	A priority.
count	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

pcounter	A bgrt_pcounter_t pointer.
prio	A priority.
count	A number of increment steps.

5.12 bugurtos/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 BGRT_PITEM_T_INIT

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

A static bgrt_pitem_t object initiation.

Warning

For internal usage.

Parameters

a	A variable name.
p	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

pitem	A bgrt_pitem_t 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

pitem	A bgrt_pitem_t 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

pitem	A bgrt_pitem_t pointer.
prio	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

pitem	A bgrt_pitem_t pointer.
xlist	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

src	A bgrt_xlist_t pointer.

Returns

An ordinary doublelinked list head pointer.

5.13 bugurtos/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 sleaping 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

p	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

a	a pointer to a process.
b	a priority of a bgrt_sync_t 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

a	a pointer to a process.
b	a priority of a bgrt_sync_t 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

a	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 BGRT_PROC_RUN_TEST

```
#define BGRT_PROC_RUN_TEST(
        a ) (((a)->flags & BGRT_PROC_STATE_RUN_MASK)>= BGRT_PROC_STATE_READY)
```

Check if process is ready or running.

Warning

For internal usage.

5.13.2.16 BGRT_PROC_SET_STATE

```
#define BGRT_PROC_SET_STATE(
        a,
        b ) ((a)->flags &= BGRT_PROC_STATE_CLEAR_MASK, (a)->flags |= b)
```

Sets a process state.

Warning

For internal usage.

5.13.2.17 BGRT_PROC_STATE_CLEAR_MASK

```
#define BGRT_PROC_STATE_CLEAR_MASK ((bgrt_flag_↩
t)0xF0)
```

An execution state clear mask.

Used clear execution state bits in proc->flags.

5.13.2.18 BGRT_PROC_STATE_CLEAR_RUN_MASK

```
#define BGRT_PROC_STATE_CLEAR_RUN_MASK ((bgrt↩
_flag_t)0xFC)
```

An execution state clear mask.

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

5.13.2.19 BGRT_PROC_STATE_DEAD

```
#define BGRT_PROC_STATE_DEAD ((bgrt_flag_t)0x5)
```

Abnormal termination, terminated with waiting ipc transactions.

5.13.2.20 BGRT_PROC_STATE_END

```
#define BGRT_PROC_STATE_END ((bgrt_flag_t)0x1)
```

Normal process termination.

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

An execution state mask.

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

A process has been run during priority change.

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

A process is waiting for priority change.

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

Is ready to run.

5.13.2.25 BGRT_PROC_STATE_PI_RUNNING `#define BGRT_PROC_STATE_PI_RUNNING ((bgrt_flag_↩ t)0xF)`

Is running.

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

Is ready to run.

5.13.2.27 BGRT_PROC_STATE_RESTART_MASK `#define BGRT_PROC_STATE_RESTART_MASK ((bgrt_↩ flag_t)0x8)`

A process execution state check mask.

Used by bgrt_priv_proc_restart to check for restart possibility.

5.13.2.28 BGRT_PROC_STATE_RUN_MASK `#define BGRT_PROC_STATE_RUN_MASK ((bgrt_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 sleaping 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 BGRT_PROC_STATE_WAIT_MASK `#define BGRT_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 BGRT_PROC_STATE_WD_STOPED `#define BGRT_PROC_STATE_WD_STOPED ((bgrt_flag_t)0x4)`

Watchdog termination.

5.13.2.39 BGRT_PROC_TO_PID `#define BGRT_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

p	A bgrt_proc_t pointer.

Returns

The process ID.

5.13.2.40 BGRT_USPD_INIT `#define BGRT_USPD_INIT( proc )`

Value:

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

Initialization.

5.13.2.41 BGRT_USPD_PROC_T `#define BGRT_USPD_PROC_T struct bgrt_priv_uspd_t`

User space process data.

5.13.2.42 BGRT_USPD_T `#define BGRT_USPD_T BGRT_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

proc	- 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

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

proc	- A pointer to a process.
prio	- 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

proc	- A pointer to a process.
prio	- 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

proc	- 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

proc	- 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

proc	- A pointer to a process.
prio	- 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

proc	- 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

proc	- A pointer to a process.
state	- 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

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

proc	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

stat	A pointer to a bgrt_ls_t 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

sched	- 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

proc	A pointer to a process that we want to place on a process list.
stat	A pointer to a bgrt_ls_t array, that controls corespondent 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

is_periodic	- 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

sync	A sync pointer.
prio	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

sync	A sync pointer.
prio	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

  Suystem 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

sc_name	A system call name.
arg	A system call argument pointer.

5.16.2.6 BGRT_SYSCALL_NVAR

```
#define BGRT_SYSCALL_NVAR(
        sc_name,
...  ) bgrt_syscall_var(BGRT_SC_ID(sc_name), __VA_ARGS__)
```

A system call macro, see bgrt_syscall_var.

Warning

For internal usage.

Parameters

sc_name	A system call name.

5.16.3 Typedef Documentation

5.16.3.1 bgrt_sc_enum

```
typedef enum _bgrt_sc_enum bgrt_sc_enum
```

Suystem call IDs.

5.16.3.2 bgrt_scsr_t

```
typedef bgrt_st_t(* bgrt_scsr_t) (void *)
```

System call service routine pointer.

5.16.3.3 bgrt_va_wr_t

```
typedef struct _bgrt_va_wr_t bgrt_va_wr_t
```

va_list wrapper.

5.16.4 Enumeration Type Documentation

5.16.4.1 _bgrt_sc_enum

```
enum _bgrt_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

syscall_num	System call number.
syscall_arg	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

num	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

t	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

t	A timer variable name.
s	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

t	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

tmr	A timer variable name.
time	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

t	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

t	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

tmr	A pointer to a timer.
time	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

time	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

vic	A pointer to a bgrt_vic_t 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

vic	A pointer to a bgrt_vic_t 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

vic	A pointer to a bgrt_vic_t 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

vint	A bgrt_vint_t pointer.
prio	A priority.
func	An ISR pointer.
arg	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

vint	A bgrt_vint_t pointer.
vic	A pointer to destination bgrt_vic_t 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

vint	A bgrt_vint_t pointer.
vic	A pointer to destination bgrt_vic_t object.

5.19 bugurtos/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

xlist	An bgrt_xlist_t 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

xlist	An bgrt_xlist_t 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

xlist	An bgrt_xlist_t pointer.
prio	A priority to switch.

Index