barelog

0.1

Generated by Doxygen 1.8.9.1

Wed Nov 25 2015 15:31:26

# **Contents**

1	Mair	n Page		1
2	Data	Struct	ure Index	5
	2.1	Data S	tructures	5
3	File	Index		7
	3.1	File Lis	st	7
4	Data	Struct	ure Documentation	9
	4.1	barelo	g_device_mem_manager_t Struct Reference	9
		4.1.1	Detailed Description	10
		4.1.2	Field Documentation	10
			4.1.2.1 read	10
			4.1.2.2 write	10
	4.2	barelo	g_event_buffer_t Struct Reference	10
		4.2.1	Detailed Description	11
		4.2.2	Field Documentation	11
			4.2.2.1 buffer	11
			4.2.2.2 empty	11
			4.2.2.3 full	11
			4.2.2.4 head	11
			4.2.2.5 tail	11
	4.3	barelo	g_host_mem_manager_t Struct Reference	11
		4.3.1	Detailed Description	12
		4.3.2	Field Documentation	12
			4.3.2.1 finalize	12
			4.3.2.2 init	12
			4.3.2.3 read	12
			4.3.2.4 write	13
	4.4	barelo	g_logger_t Struct Reference	14
		4.4.1	Detailed Description	14

iv CONTENTS

			4.4.2.1	get_clock	. 14
			4.4.2.2	init_clock	. 14
			4.4.2.3	start_clock	. 15
	4.5	barelo	g_platform	_t Struct Reference	. 15
		4.5.1	Detailed	Description	. 15
		4.5.2	Field Doo	cumentation	. 15
			4.5.2.1	mem_space	. 15
			4.5.2.2	name	. 15
	4.6	barelo	g_result_bu	uffer_t Struct Reference	. 15
		4.6.1	Detailed	Description	. 16
		4.6.2	Field Doo	cumentation	. 16
			4.6.2.1	buffer	. 16
			4.6.2.2	buffer_length	. 16
			4.6.2.3	sub_buffer_length	. 16
	4.7	barelo	g_shared_i	mem_buffer_t Struct Reference	. 16
		4.7.1	Detailed	Description	. 16
		4.7.2	Field Doo	cumentation	. 17
			4.7.2.1	events	. 17
			4.7.2.2	imax	. 17
			4.7.2.3	index	. 17
5	File	Docum	entation		19
5	<b>File</b> 5.1		entation	/barelog_buffer.h File Reference	19
5			entation on/include/	/barelog_buffer.h File Reference	<b>19</b> . 19
5		commo	entation on/include/ Detailed	Description	19 . 19 . 20
5	5.1	commo	entation on/include/ Detailed on/include/		19 . 19 . 20 . 21
5	5.1	5.1.1 commo	entation on/include/ Detailed on/include/ Detailed	Description	19 . 19 . 20 . 21
5	5.1	5.1.1 commo	entation on/include/ Detailed on/include/ Detailed	Description /barelog_config.h File Reference  Description	19 . 19 . 20 . 21 . 22
5	5.1	5.1.1 commo	entation on/include/ Detailed   on/include/ Detailed   Macro De	Description  /barelog_config.h File Reference  Description  efinition Documentation	19 . 19 . 20 . 21 . 22 . 22
5	5.1	5.1.1 commo	entation on/include/ Detailed on/include/ Detailed Macro Detailed 5.2.2.1	Description  /barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE	19 . 19 . 20 . 21 . 22 . 22 . 22
5	5.1	5.1.1 commo	pentation on/include/ Detailed on/include/ Detailed Macro Detailed 5.2.2.1	Description  barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE	19 . 19 . 20 . 21 . 22 . 22 . 22
5	5.1	5.1.1 commo	pentation on/include/ Detailed on/include/ Detailed on/include/ Macro Detailed of 5.2.2.1 5.2.2.2 5.2.2.3	Description  /barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22
5	5.1	5.1.1 commo	pentation on/include/ Detailed   Detailed   Macro Detailed   5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4	Description  /barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22
5	5.1	5.1.1 commo	pentation on/include/ Detailed on/include/ Detailed on/include/ Macro Detailed on/include/ 5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4 5.2.2.5	Description  /barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE  BARELOG_LOCAL_MEM_PER_CORE	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22 . 22
5	5.1	5.1.1 commo 5.2.1 5.2.2	pentation on/include/ Detailed on/include/ Detailed on/include/ Detailed on/include/ 5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4 5.2.2.5 5.2.2.6 5.2.2.7	Description  barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE  BARELOG_LOCAL_MEM_PER_CORE  BARELOG_NB_CORES	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22 . 22 . 22
5	5.1	5.1.1 commo 5.2.1 5.2.2	pentation on/include/ Detailed on/include/ Detailed on/include/ Macro Detailed on/include/ 5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4 5.2.2.5 5.2.2.6 5.2.2.7 on/include/	Description  /barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE  BARELOG_LOCAL_MEM_PER_CORE  BARELOG_NB_CORES  BARELOG_PLATFORM_NAME_LENGTH	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22 . 22 . 22
5	5.1	commo 5.1.1 commo 5.2.1 5.2.2	entation on/include/ Detailed   On/include/ Detailed   Macro Detailed   5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4 5.2.2.5 5.2.2.6 5.2.2.7 on/include/ Detailed	Description  barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE  BARELOG_LOCAL_MEM_PER_CORE  BARELOG_NB_CORES  BARELOG_PLATFORM_NAME_LENGTH  barelog_event.h File Reference	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22 . 22 . 23 . 24
5	5.1	commo 5.1.1 commo 5.2.1 5.2.2	entation on/include/ Detailed   On/include/ Detailed   Macro Detailed   5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4 5.2.2.5 5.2.2.6 5.2.2.7 on/include/ Detailed	Description  barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE  BARELOG_LOCAL_MEM_PER_CORE  BARELOG_NB_CORES  BARELOG_PLATFORM_NAME_LENGTH  barelog_event.h File Reference  Description	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22 . 24 . 24
5	5.1	commo 5.1.1 commo 5.2.1 5.2.2	entation on/include/ Detailed   on/include/ Detailed   Macro Detailed   5.2.2.1 5.2.2.2 5.2.2.3 5.2.2.4 5.2.2.5 5.2.2.6 5.2.2.7 on/include/ Detailed   Macro Detailed   Macro Detailed   Macro Detailed   5.3.2.1	Description  barelog_config.h File Reference  Description  efinition Documentation  BARELOG_CHECK_MODE  BARELOG_EVENT_MAX_SIZE  BARELOG_EVENT_SHARED_MEM_MAX  BARELOG_LOCAL_MEM_ATTRIBUTE  BARELOG_LOCAL_MEM_PER_CORE  BARELOG_NB_CORES  BARELOG_PLATFORM_NAME_LENGTH  barelog_event.h File Reference  Description  efinition Documentation	19 . 19 . 20 . 21 . 22 . 22 . 22 . 22 . 22 . 24 . 24

CONTENTS

		5.3.3.2 bar	elog_event_to_string	24
		5.3.3.3 bar	elog_events_to_strings	25
	5.3.4	Variable Docu	mentation	25
		5.3.4.1 BA	RELOG_EVENT_INITIALIZER	25
5.4	commo	n/include/bare	log_internal.h File Reference	25
	5.4.1	Detailed Desc	cription	27
	5.4.2	Macro Definit	ion Documentation	27
		5.4.2.1 BA	RELOG_BUF_MAX_SIZE	27
		5.4.2.2 BA	RELOG_DEBUG_MEM_SIZE	27
		5.4.2.3 BA	RELOG_DEBUG_MODE_I	27
		5.4.2.4 BA	RELOG_DEBUG_OFF	27
		5.4.2.5 BA	RELOG_ERR	28
		5.4.2.6 BA	RELOG_EVENT_CONVERSION_ERR	28
		5.4.2.7 BA	RELOG_EVENT_PER_CORE_MAX	28
		5.4.2.8 BA	RELOG_EVENT_PER_CORE_SHR_MEM_MAX	28
		5.4.2.9 BA	RELOG_HOST_NB_MEM_SPACE	28
		5.4.2.10 BA	RELOG_INCONSISTENT_PARAM_ERR	28
		5.4.2.11 BA	RELOG_INIT_ERR	28
		5.4.2.12 BA	RELOG_MUTEX_TRY_MAX	28
		5.4.2.13 BA	RELOG_NB_MUTEX_BYTES	28
		5.4.2.14 BA	RELOG_SAFE_MEM_SIZE	29
		5.4.2.15 BA	RELOG_SAFE_MODE_I	29
		5.4.2.16 BA	RELOG_SHARED_MEM_DATA_OFFSET	29
		5.4.2.17 BA	RELOG_SHARED_MEM_MAX	29
		5.4.2.18 BA	RELOG_SHARED_MEM_PER_CORE_MAX	29
		5.4.2.19 bar	elog_shrmem_mutex_t	29
		5.4.2.20 BA	RELOG_SHRMEM_READ_ERR	29
		5.4.2.21 BA	RELOG_SHRMEM_WRITE_ERR	29
		5.4.2.22 BA	RELOG_SUCCESS	29
		5.4.2.23 BA	RELOG_TIMEOUT_ERR	30
		5.4.2.24 BA	RELOG_UNINITIALIZED_PARAM_ERR	30
5.5	commo	on/include/bare	log_mem_space.h File Reference	30
	5.5.1	Detailed Desc	cription	31
	5.5.2	Function Doc	umentation	31
		5.5.2.1a	attribute	31
	5.5.3	Variable Docu	mentation	31
		5.5.3.1 ME	M_SPACE_INITIALIZER	31
5.6	commo	on/include/bare	log_platform.h File Reference	32
	5.6.1	Detailed Desc	cription	33
5.7	commo	n/include/bare	log_policy.h File Reference	33

vi CONTENTS

	5.7.1	Detailed	Description	34
	5.7.2	Enumera	tion Type Documentation	34
		5.7.2.1	barelog_policy_t	34
5.8	host/ind	clude/bare	log_host.h File Reference	34
	5.8.1	Detailed	Description	35
	5.8.2	Macro De	efinition Documentation	35
		5.8.2.1	barelog_host_finalize	35
		5.8.2.2	barelog_host_init	35
		5.8.2.3	barelog_read_debug	36
		5.8.2.4	barelog_read_log	36
5.9	host/ind	clude/bare	log_host_mem_manager.h File Reference	36
	5.9.1	Detailed	Description	37
	5.9.2	Function	Documentation	38
		5.9.2.1	host_mem_manager_finalize	38
		5.9.2.2	host_mem_manager_init	38
		5.9.2.3	host_mem_manager_read_debug	38
		5.9.2.4	host_mem_manager_read_mem_space	38
5.10	platforn	ns/barelog	parallella.h File Reference	39
	5.10.1	Detailed	Description	39
5.11	target/i	nclude/bar	relog_device_mem_manager.h File Reference	40
	5.11.1	Detailed	Description	41
	5.11.2	Function	Documentation	42
		5.11.2.1	barelog_debug_log	42
		5.11.2.2	device_mem_manager_clean	42
		5.11.2.3	device_mem_manager_clean_buffer	42
		5.11.2.4	device_mem_manager_clean_memory	42
		5.11.2.5	device_mem_manager_flush	43
		5.11.2.6	device_mem_manager_flush_buffer	44
		5.11.2.7	device_mem_manager_init	44
		5.11.2.8	device_mem_manager_is_buffer_full	44
		5.11.2.9	device_mem_manager_write_buffer	44
5.12	target/i	nclude/bar	relog_logger.h File Reference	45
	5.12.1	Detailed	Description	47
	5.12.2	Macro De	efinition Documentation	47
		5.12.2.1	barelog_clean	47
		5.12.2.2	barelog_clean_buffer	47
		5.12.2.3	barelog_clean_memory	47
		5.12.2.4	barelog_flush	47
		5.12.2.5	barelog_flush_buffer	47
		5.12.2.6	barelog_is_buffer_full	48

	5.12.3	Enumera	tion Type	Document	ation	 		 		 			 	 	48
		5.12.3.1	barelog_	_lvl_t		 		 		 			 	 	48
	5.12.4	Function	Documen	tation		 		 		 				 	48
		5.12.4.1	barelog_	immediate	_log	 		 		 				 	48
		5.12.4.2	barelog_	init_logger	r	 		 		 			 	 	48
		5.12.4.3	barelog_	log		 		 		 				 	48
		5.12.4.4	barelog_	start		 		 		 				 	49
landar.															-4
Index															51

vii

**CONTENTS** 

## Chapter 1

## Main Page

**barelog** is a set of C99 modules that can be used to do some logging on many-core systems. The primary targets of barelog are the embedded heterogeneous many-core platforms (such as the Parallella platform) or any core that is too small to run any Linux based OS, thus forbidding the use of traditional tools.

The main use-case would be the logging of some calculus-specific cores that don't have any kernel but **can still access a shared memory space** to interact with a more "traditional" host (that is to say another CPU able to run a Linux kernel).

Please note that due to it's current limitations, barelog is not meant to be used for serious, efficient logging/tracing. For a more sophisticated tool that provides very efficient tracing, please see also barectf.

Note: in the following document, the terms "host" and "target" refer respectively to a system running a Linux kernel and able to initialize the shared memory and to the specific core that doesn't run any kernel.

## Key features:

- Entirely configurable: you have full control over the functions used by the modules to interact with the shared memory as well as the total amount of memory used by barelog (inside each core as well as in the shared section).
- Easy to use: a simple call of the **barelog\_log()** function (after proper initialization of the modules) allows you to log events without any further complications.
- Provides several "functioning modes": you can enable/disable some parts of the code to suit your needs. For
  example, to gain some performance, you might want to disable the "DEBUG\_MODE" that only offers some
  internal debugging functions.
- Flush events whenever you want: a round-buffer allows you to store the events in the local memory of the logged core as long as you want before actually flushing them into the shared memory. You have full control over which stored event to actually put into the shared space.
- Format the events data as you want: since the logging module use a modified version of "snprintf" you can store any type of data (represented as a string) in a event.

## **Current limitations:**

- Pretty heavy impact on the performances: since the logging module use a modified version of "snprintf", it's quite demanding in terms of clock cycles to produce an event.
- The size of the actual event's data is statically fixed: that means that if the events data are not full, there will be waste of both local memory (of the logged core) and shared memory.
- The data of an event is represented by a string: which means that you can't directly access to all the data logged into that event since they are wrapped in a string.

2 Main Page

#### Using

#### Compiling the modules

 You first need to edit the common/include/config.h file to ensure that barelog is configured to suit your needs. Note that you can directly include a custom configuration header by placing it inside the platforms directory and then including it.

- 2. Once it's done, you may want need to edit the Makefile to properly set the compiler used to compile the target module code. You can also set the TARGET\_CC flag during the 'make'.
- 3. Then simply compile the modules using the provided Makefile. You can specify whether or not to use a cross-compiling toolchain by setting the CROSS\_COMPILE flag. You can also decide if you rather want the resulting libraries to be static (.a) or shared (.so) by setting the HLIBTYPE and/or TLIBTYPE flags (where 'H' stands for Host and 'T' for Target). The default behavior is to produce static libraries.

make

Or

make HLIBTYPE=so TLIBTYP=a

If everything went well, two libraries should have been produced in the libs folder:

- libbarelog host: targets the host program.
- libbarelog\_logger: targets the target program.

Instrumenting and compiling your code

Instrumenting your code

Once you have compiled the modules, you just need to instrument your code to get started!

To do that, you have to follow those steps:

- Initialize the host: you will have to create the <u>barelog\_platform\_t</u> along with some memory management functions and to register them to the logger on the host by calling the <u>barelog\_host\_init()</u> function. This will allocate all the needed chunks of shared memory according to the "config" file and initialize the all host module.
- 2. Initialize the target: this basically involve the same steps as above but with everything specific to the target.
- 3. Instrument the target code: by using a combination of the barelog\_log() barelog\_flush() and barelog\_clean(), you should be able to produce and manage the events inside the logged core.
- 4. Retrieve the events on the host: the host API offers some functions to extract and display the logged events (please see the given example).
- 5. Finalize the logger: once you're done logging around, use the \*\*barelog\_finalize() function to ensure every resource is correctly deallocated.

Please refer to the documentation and/or the given example for more informations.

**WARNING**: if you use barelog, some part of the shared memory (beginning at the given platform's mem\_space) will be used by it. To avoid every hazardous behavior, consider using the **BARELOG\_SHARED\_MEM\_MAX** macro (which give the size (in bytes) of the memory taken by barelog) when allocating new chunks of memory for your personal needs.

## Compiling your code

Now that we have everything ready, we just need to compile our programs (one running on the host and the other on the target).

First of all, make sure that the previously generated barelog's libraries can be found by the compiler/linker. Assuming that your using gcc, you just need to specify the -L option :

```
gcc -L path/to/libraries/
```

You will then have to build the host program using the **libbarelog\_host** library and the target program with the **libbarelog\_logger** library:

```
gcc -L path/to/libraries/ target_main.c -lbarelog_logger
gcc -L path/to/libraries/ host_main.c -lbarelog_host
```

Of course, this need to be adapted in case you need to use another compiler.

#### Create your own configuration file

To create you own configuration file, you can simply follow the "template" given by **common/include/config.h**. You don't have to fulfill every fields since there already is some default values (please refer to config.h).

Once it's done, you just have to put it in the **platforms** directory, thus guaranteeing that you could later include it in the config header.

#### Warnings

- The core numbering on the target must begin at 0.
- The barelog device mem manager module should be placed in the local memory of each logged core.
- The "SAFE" mode, providing shared memory synchronization mechanism is still to be tested, thus implying that no guarantee can be provided.

## Configuring new behaviors/functionalities

You might want to add some functionalities that need some data stored into the shared memory space of barelog. Since this space is strictly ordered, you will have to follow those steps to ensure the good global behavior of the modules:

- 1. Define the size taken by those data inside the **barelog\_internal.h** file: you can use the following naming convention: 'BARELOG\_FUNCNAME\_MEM\_SIZE'
- 2. Edit the BARELOG\_SHARED\_MEM\_DATA\_OFFSET macro to take the new data in account while computing the offsets of each barelog's data inside the shared memory.
- 3. Reserve a new mem space for your data by adding '1' to the BARELOG HOST NB MEM SPACE macro.
- 4. Define the new index inside the host's mem\_space table of the new data: you can use the following naming convention: 'BARELOG\_FUNCNAME\_I'. Please be careful with the index since some may already have been taken and the BARELOG\_NB\_CORES first refer to the actual events reserved memory spaces. You can follow what has been done with BARELOG\_DEBUG\_MODE and BARELOG\_SAFE\_MODE to get the global picture of how to do it.
- 1. Modify the behavior of the "host\_mem\_manager\_init()" and "host\_mem\_manager\_finalize()" functions to respectively init and finalize the newly reserved mem\_space.

Main Page

# **Chapter 2**

# **Data Structure Index**

## 2.1 Data Structures

Here are the data structures with brief descriptions:

barelog_device_mem_manager_t	٠
barelog_event_buffer_t	10
barelog_host_mem_manager_t	1
barelog_logger_t	14
barelog_platform_t	15
barelog_result_buffer_t	15
barelog shared mem buffer t	16

6 Data Structure Index

# **Chapter 3**

# File Index

## 3.1 File List

Here is a list of all documented files with brief descriptions:

common/barelog_event.c	??
common/barelog_mem_space.c	??
common/include/barelog_buffer.h	• •
Module defining the different buffers used by barelog's internals	19
common/include/barelog config.h	10
Module defining the configurations used by barelog	21
common/include/barelog_event.h	21
Module defining the events and their related functions	23
common/include/barelog internal.h	20
Module defining the internal configurations of barelog	25
common/include/barelog mem space.h	20
<del>-</del> ·	30
Module defining mem_space structure	30
common/include/barelog_platform.h	32
Module defining a platform to use barelog against	32
common/include/barelog_policy.h	00
Module defining the different policies that can be used when an events buffer is full	33
host/barelog_host.c	?? ??
host/barelog_host_mem_manager.c	"
host/include/barelog_host.h	0.4
Module providing some nice wrapping for the host_mem_manager	34
host/include/barelog_host_mem_manager.h	00
Module defining all functions offered by barelog for the host program	36
platforms/barelog_parallella.h	-00
Module defining the configurations used by barelog specifically for the Parallella platform	39
target/barelog_device_mem_manager.c	??
target/barelog_logger.c	??
target/barelog_snprintf.c	??
target/include/barelog_device_mem_manager.h	
Module defining all functions offered by barelog for the host program	40
target/include/barelog_logger.h	
Module providing some nice wrapping for the device_mem_manager	45
target/include/barelog_snprintf.h	??

8 File Index

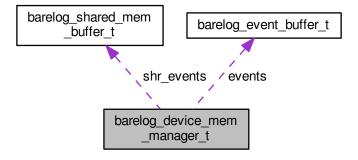
## **Chapter 4**

## **Data Structure Documentation**

## 4.1 barelog\_device\_mem\_manager\_t Struct Reference

#include <barelog\_device\_mem\_manager.h>

Collaboration diagram for barelog\_device\_mem\_manager\_t:



## **Data Fields**

- uint8\_t initialized
- uint32\_t core
- barelog\_mem\_space\_t mem\_space
- barelog\_event\_buffer\_t events
- barelog\_shared\_mem\_buffer\_t shr\_events
- barelog\_policy\_t buffer\_policy
- barelog\_policy\_t memory\_policy
- int8\_t(\* read )(const void \*address, size\_t size, void \*buffer)
- int8\_t(\* write )(void \*address, size\_t size, const void \*buffer)
- void \* debug\_address

## 4.1.1 Detailed Description

Structure used to hold all of the barelog device manager functions. We use pointers to allow the user to use the functions of their choice, depending on the logged platform.

Definition at line 52 of file barelog\_device\_mem\_manager.h.

#### 4.1.2 Field Documentation

4.1.2.1 int8\_t(\* barelog\_device\_mem\_manager\_t::read) (const void \*address, size\_t size, void \*buffer)

Function used by the target to read into the shared memory.

#### **Parameters**

address	the address to read.
size	the size of the memory to read.
buffer	the buffer in which to store the reading result.

#### Returns

BARELOG SUCCESS if all is clear, an error code otherwise.

Definition at line 73 of file barelog\_device\_mem\_manager.h.

4.1.2.2 int8\_t(\* barelog\_device\_mem\_manager\_t::write) (void \*address, size\_t size, const void \*buffer)

Function used by the target to write into the shared memory.

## Parameters

address	the address to write.
size	the size of the memory to write.
buffer	the buffer from which to write the reading result.

### Returns

BARELOG\_SUCCESS if all is clear, an error code otherwise.

Definition at line 80 of file barelog\_device\_mem\_manager.h.

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

• target/include/barelog\_device\_mem\_manager.h

## 4.2 barelog\_event\_buffer\_t Struct Reference

#include <barelog\_buffer.h>

## **Data Fields**

- barelog\_event\_t buffer [BARELOG\_EVENT\_PER\_CORE\_MAX]
- uint32 t head
- uint32\_t tail
- uint8 t full
- uint8\_t empty

## 4.2.1 Detailed Description

Queue of events, used to store the local events into a core local memory.

Definition at line 46 of file barelog\_buffer.h.

#### 4.2.2 Field Documentation

4.2.2.1 barelog\_event\_t barelog\_event\_buffer\_t::buffer[BARELOG\_EVENT\_PER\_CORE\_MAX]

buffer containing the events (queue)

Definition at line 48 of file barelog\_buffer.h.

4.2.2.2 uint8\_t barelog\_event\_buffer\_t::empty

indicates whether or not the buffer is empty

Definition at line 56 of file barelog\_buffer.h.

4.2.2.3 uint8\_t barelog\_event\_buffer\_t::full

indicates whether or not the buffer is full

Definition at line 54 of file barelog\_buffer.h.

4.2.2.4 uint32\_t barelog\_event\_buffer\_t::head

index of the next position to store an event

Definition at line 50 of file barelog\_buffer.h.

4.2.2.5 uint32\_t barelog\_event\_buffer\_t::tail

index of the first position effectively used

Definition at line 52 of file barelog\_buffer.h.

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

· common/include/barelog buffer.h

## 4.3 barelog\_host\_mem\_manager\_t Struct Reference

```
#include <barelog_host_mem_manager.h>
```

## **Data Fields**

- · uint8 t initialized
- barelog\_mem\_space\_t mem\_space [BARELOG\_HOST\_NB\_MEM\_SPACE]
- void \*(\* init )(void \*address, size\_t size, void \*data)
- int8\_t(\* read )(const void \*address, size\_t size, void \*buffer)
- int8 t(\* write )(void \*address, size t size, const void \*buffer)
- int8\_t(\* finalize )(void \*mem\_space)

## 4.3.1 Detailed Description

Structure used to hold all of the barelog host manager functions. We use pointers to allow the user to use the functions of their choice, depending on the logged platform.

Definition at line 51 of file barelog\_host\_mem\_manager.h.

## 4.3.2 Field Documentation

4.3.2.1 int8\_t(\* barelog\_host\_mem\_manager\_t::finalize) (void \*mem\_space)

Function used to finalize a previously initialized chunk of shared memory.

#### **Parameters**

mem_space	the mem_space to finalize.

#### Returns

BARELOG SUCCESS if all is clear, an error code otherwise.

Definition at line 88 of file barelog\_host\_mem\_manager.h.

4.3.2.2 void\*(\* barelog\_host\_mem\_manager\_t::init) (void \*address, size\_t size, void \*data)

Function used to initialize a chunk in the shared memory space.

#### **Parameters**

address	the beginning address of the chunk to initialize.
size	the size of the chunk to initialize.
data	(optional) parameter that may be used by the initialization function.

#### Returns

must return the virtual address corresponding to the base of the allocated memory space (if any). After the initialization, one must use this address to access the allocated memory within the host. Should return NULL in case something went wrong.

Definition at line 69 of file barelog\_host\_mem\_manager.h.

4.3.2.3 int8\_t(\* barelog\_host\_mem\_manager\_t::read) (const void \*address, size\_t size, void \*buffer)

Function used by the host to read into the shared memory.

#### **Parameters**

address	the address to read.
size	the size of the memory to read.
buffer	the buffer in which to store the reading result.

#### Returns

BARELOG\_SUCCESS if all is clear, an error code otherwise.

Definition at line 76 of file barelog\_host\_mem\_manager.h.

4.3.2.4 int8\_t(\* barelog\_host\_mem\_manager\_t::write) (void \*address, size\_t size, const void \*buffer)

Function used by the host to write into the shared memory.

#### **Parameters**

address	the address to write.
size	the size of the memory to write.
buffer	the buffer from which to write the reading result.

#### Returns

BARELOG\_SUCCESS if all is clear, an error code otherwise.

Definition at line 83 of file barelog\_host\_mem\_manager.h.

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

· host/include/barelog\_host\_mem\_manager.h

## 4.4 barelog\_logger\_t Struct Reference

```
#include <barelog_logger.h>
```

#### **Data Fields**

- barelog\_lvl\_t log\_lvl
- uint32\_t(\* get\_clock )(void)
- int8\_t(\* init\_clock )(void)
- int8\_t(\* start\_clock )(void)

## 4.4.1 Detailed Description

Structure used to hold all of the barelog logger functions. We use pointers to allow the user to use the functions of their choice, depending on the logged platform.

Definition at line 66 of file barelog\_logger.h.

## 4.4.2 Field Documentation

```
4.4.2.1 uint32_t(* barelog_logger_t::get_clock) (void)
```

Function used to retrieve the current clock of the core.

## Returns

a timestamp on 32 bits.

Definition at line 71 of file barelog\_logger.h.

4.4.2.2 int8\_t(\* barelog\_logger\_t::init\_clock) (void)

Function used to initialize (reset) the current clock of the core.

## Returns

BARELOG\_SUCCESS on success, an error code otherwise.

Definition at line 75 of file barelog\_logger.h.

4.4.2.3 int8\_t(\* barelog\_logger\_t::start\_clock) (void)

Function used to start the current clock of the core.

Returns

BARELOG\_SUCCESS on success, an error code otherwise.

Definition at line 79 of file barelog\_logger.h.

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

• target/include/barelog\_logger.h

## 4.5 barelog\_platform\_t Struct Reference

```
#include <barelog_platform.h>
```

## **Data Fields**

- char name [BARELOG\_PLATFORM\_NAME\_LENGTH]
- barelog\_mem\_space\_t mem\_space

## 4.5.1 Detailed Description

Structure of a platform to use barelog against.

Definition at line 43 of file barelog\_platform.h.

## 4.5.2 Field Documentation

4.5.2.1 barelog\_mem\_space\_t barelog\_platform\_t::mem\_space

Shared memory space to use barelog on

Definition at line 47 of file barelog\_platform.h.

#### 4.5.2.2 char barelog\_platform\_t::name[BARELOG\_PLATFORM\_NAME\_LENGTH]

Name of the platform (deprecated)

Definition at line 45 of file barelog\_platform.h.

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

• common/include/barelog\_platform.h

## 4.6 barelog\_result\_buffer\_t Struct Reference

#include <barelog\_buffer.h>

#### **Data Fields**

- char \*\* buffer
- · size\_t buffer\_length
- size\_t sub\_buffer\_length

## 4.6.1 Detailed Description

Structure used to store the events of a logged core, represented by strings and not actual events (for display or treatment purposes).

Definition at line 63 of file barelog\_buffer.h.

#### 4.6.2 Field Documentation

```
4.6.2.1 char** barelog_result_buffer_t::buffer
```

buffer of events (considered as strings)

Definition at line 65 of file barelog\_buffer.h.

```
4.6.2.2 size_t barelog_result_buffer_t::buffer_length
```

number of events to consider

Definition at line 67 of file barelog\_buffer.h.

4.6.2.3 size\_t barelog\_result\_buffer\_t::sub\_buffer\_length

length of each event

Definition at line 69 of file barelog\_buffer.h.

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

• common/include/barelog\_buffer.h

## 4.7 barelog\_shared\_mem\_buffer\_t Struct Reference

```
#include <barelog_buffer.h>
```

## **Data Fields**

- barelog\_event\_t \* events
- uint32\_t index
- uint32\_t imax

## 4.7.1 Detailed Description

Structure used to store the events in the shared memory.

Definition at line 75 of file barelog\_buffer.h.

## 4.7.2 Field Documentation

 $\textbf{4.7.2.1} \quad barelog\_event\_t*\ barelog\_shared\_mem\_buffer\_t::events$ 

events queue

Definition at line 77 of file barelog\_buffer.h.

4.7.2.2 uint32\_t barelog\_shared\_mem\_buffer\_t::imax

max index

Definition at line 81 of file barelog\_buffer.h.

4.7.2.3 uint32\_t barelog\_shared\_mem\_buffer\_t::index

current index inside the queue

Definition at line 79 of file barelog\_buffer.h.

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

• common/include/barelog\_buffer.h



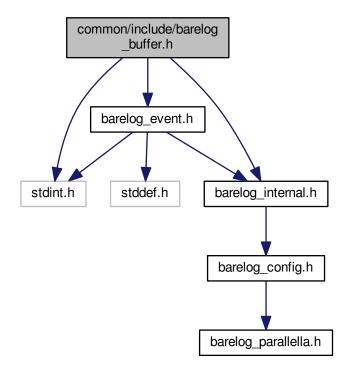
## **Chapter 5**

## **File Documentation**

## 5.1 common/include/barelog\_buffer.h File Reference

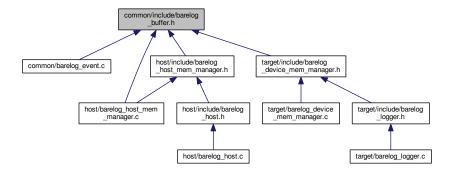
Module defining the different buffers used by barelog's internals.

```
#include <stdint.h>
#include "barelog_internal.h"
#include "barelog_event.h"
Include dependency graph for barelog_buffer.h:
```



20 File Documentation

This graph shows which files directly or indirectly include this file:



## **Data Structures**

- struct barelog\_event\_buffer\_t
- · struct barelog\_result\_buffer\_t
- struct barelog\_shared\_mem\_buffer\_t

## 5.1.1 Detailed Description

Module defining the different buffers used by barelog's internals.

This header defines the different types of buffer used by barelog.

**Author** 

Thomas Bertauld

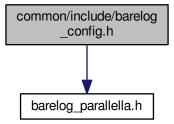
Date

17/10/2015

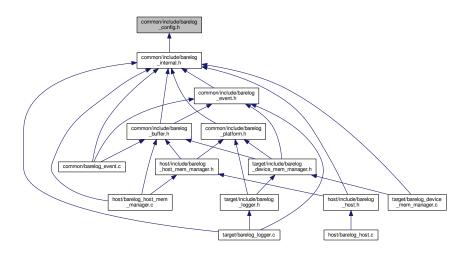
## 5.2 common/include/barelog\_config.h File Reference

Module defining the configurations used by barelog.

#include "barelog\_parallella.h"
Include dependency graph for barelog\_config.h:



This graph shows which files directly or indirectly include this file:



#### **Macros**

- #define BARELOG\_NB\_CORES 16
- #define BARELOG\_EVENT\_SHARED\_MEM\_MAX 1000000
- #define BARELOG\_PLATFORM\_NAME\_LENGTH 20
- #define BARELOG\_EVENT\_MAX\_SIZE 100
- #define BARELOG\_LOCAL\_MEM\_PER\_CORE 1000
- #define BARELOG\_LOCAL\_MEM\_ATTRIBUTE
- #define BARELOG\_CHECK\_MODE 1

22 File Documentation

## 5.2.1 Detailed Description

Module defining the configurations used by barelog.

This header is used to define every external parameters that we might use to configure the behavior of the application.

**Author** 

Thomas Bertauld

Date

17/10/2015

#### 5.2.2 Macro Definition Documentation

## 5.2.2.1 #define BARELOG\_CHECK\_MODE 1

Memory synchronization (mutexes) between host and device (/!\ not fully tested) Defines whether or not we should apply defensive strategies on code

Definition at line 84 of file barelog\_config.h.

## 5.2.2.2 #define BARELOG\_EVENT\_MAX\_SIZE 100

Maximum size (in bytes) of a barelog\_event :

Definition at line 62 of file barelog\_config.h.

## 5.2.2.3 #define BARELOG\_EVENT\_SHARED\_MEM\_MAX 1000000

Maximum size (in bytes) taken in the shared memory by barelog events :

Definition at line 52 of file barelog config.h.

### 5.2.2.4 static void \*mutex\_byte\_address BARELOG\_LOCAL\_MEM\_ATTRIBUTE

(Optional) attribute used to ensure that some parts of the code are stored in the local memory of the traced core.

Definition at line 74 of file barelog\_config.h.

## 5.2.2.5 #define BARELOG\_LOCAL\_MEM\_PER\_CORE 1000

Maximum size (in bytes) of each core's local memory reserved for barelog :

Definition at line 67 of file barelog config.h.

## 5.2.2.6 #define BARELOG\_NB\_CORES 16

Extern configuration file to load (if any). Number of cores to log on

Definition at line 47 of file barelog\_config.h.

## 5.2.2.7 #define BARELOG\_PLATFORM\_NAME\_LENGTH 20

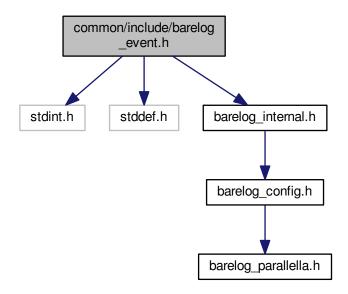
Maximum string length of the platform name (deprecated):

Definition at line 57 of file barelog\_config.h.

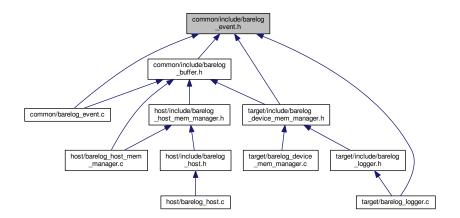
## 5.3 common/include/barelog\_event.h File Reference

Module defining the events and their related functions.

```
#include <stdint.h>
#include <stddef.h>
#include "barelog_internal.h"
Include dependency graph for barelog event.h:
```



This graph shows which files directly or indirectly include this file:



## **Macros**

#define EVENT\_TO\_STRING\_SIZE BARELOG\_EVENT\_MAX\_SIZE\*2

24 File Documentation

## **Typedefs**

typedef struct barelog result buffer t barelog result buffer t

#### **Functions**

- struct \_\_attribute\_\_ ((packed))
- int8\_t barelog\_event\_to\_string (const barelog\_event\_t event, char \*buffer)
- int8\_t barelog\_events\_to\_strings (const barelog\_event\_t \*events, size\_t n, barelog\_result\_buffer\_t \*buffer)

## **Variables**

- · barelog\_event\_t
- const barelog\_event\_t BARELOG\_EVENT\_INITIALIZER

## 5.3.1 Detailed Description

Module defining the events and their related functions.

This header defines the main structure of an event as seen by every other barelog files. It also defines some common functions to manipulate those events.

**Author** 

Thomas Bertauld

Date

17/10/2015

## 5.3.2 Macro Definition Documentation

```
5.3.2.1 #define EVENT_TO_STRING_SIZE BARELOG_EVENT_MAX_SIZE*2
```

Maximum size (in bytes) of a formatted string containing all barelog event information.

Definition at line 49 of file barelog\_event.h.

## 5.3.3 Function Documentation

```
5.3.3.1 struct __attribute__ ( (packed) )
```

Main structure of what we call an event. timestamp of the event

core on which the event occured

actual data contained by the event

Definition at line 54 of file barelog\_event.h.

5.3.3.2 int8\_t barelog\_event\_to\_string ( const barelog\_event\_t event, char \* buffer )

Converts an event structure into a single string.

#### **Parameters**

event	event to convert.
buffer	buffer to use for the conversion (should be at least EVENT_TO_STRING_SIZE bytes long).

#### Returns

the return code of snprintf().

Definition at line 39 of file barelog\_event.c.

5.3.3.3 int8\_t barelog\_events\_to\_strings ( const barelog\_event\_t \* events, size\_t n, barelog\_result\_buffer\_t \* buffer )

Converts an events queue into a buffer of strings.

## **Parameters**

events	events queue to convert.
n	size of the events queue.
buffer	result buffer.

#### Returns

the BARELOG\_SUCCESS if everything went well, an error code otherwise.

Definition at line 51 of file barelog\_event.c.

## 5.3.4 Variable Documentation

5.3.4.1 const barelog\_event\_t BARELOG\_EVENT\_INITIALIZER

Event initializer, every field is set to 0 except for data, set to "".

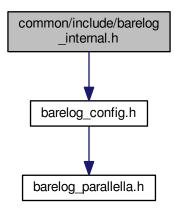
Definition at line 33 of file barelog\_event.c.

## 5.4 common/include/barelog\_internal.h File Reference

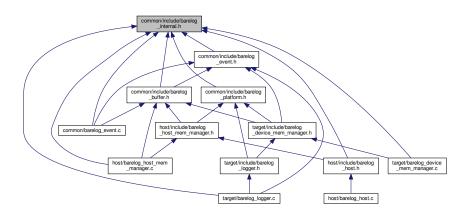
Module defining the internal configurations of barelog.

26 File Documentation

#include "barelog\_config.h"
Include dependency graph for barelog\_internal.h:



This graph shows which files directly or indirectly include this file:



## **Macros**

- #define BARELOG SUCCESS 0
- #define BARELOG\_ERR -1
- #define BARELOG\_UNINITIALIZED\_PARAM\_ERR -2
- #define BARELOG\_INCONSISTENT\_PARAM\_ERR -3
- #define BARELOG SHRMEM WRITE ERR -4
- #define BARELOG\_SHRMEM\_READ\_ERR -5
- #define BARELOG\_TIMEOUT\_ERR -6
- #define BARELOG\_EVENT\_CONVERSION\_ERR -7
- #define BARELOG\_INIT\_ERR -8
- #define barelog\_shrmem\_mutex\_t uint8\_t
- #define BARELOG\_MUTEX\_TRY\_MAX 5
- #define BARELOG\_NB\_MUTEX\_BYTES BARELOG\_NB\_CORES

- #define BARELOG\_SAFE\_MEM\_SIZE BARELOG\_NB\_MUTEX\_BYTES
- #define BARELOG\_SAFE\_MODE\_I BARELOG\_NB\_CORES
- #define BARELOG\_DEBUG\_MEM\_SIZE sizeof(barelog\_event\_t)
- #define BARELOG\_DEBUG\_MODE\_I (BARELOG\_NB\_CORES + BARELOG\_SAFE\_MODE)
- #define BARELOG\_DEBUG\_OFF BARELOG\_SAFE\_MEM\_SIZE

- #define BARELOG\_BUF\_MAX\_SIZE (BARELOG\_EVENT\_MAX\_SIZE 2\*sizeof(uint32\_t))
- #define BARELOG\_SHARED\_MEM\_PER\_CORE\_MAX (BARELOG\_EVENT\_SHARED\_MEM\_MAX/BAR ← ELOG\_NB\_CORES)
- #define BARELOG\_HOST\_NB\_MEM\_SPACE (BARELOG\_NB\_CORES + BARELOG\_SAFE\_MODE + B↔ ARELOG\_DEBUG\_MODE)

### 5.4.1 Detailed Description

Module defining the internal configurations of barelog.

This header defines every configuration's data needed internally by every barelog's file. Modify at your own risks!

**Author** 

Thomas Bertauld

Date

17/10/2015

### 5.4.2 Macro Definition Documentation

## 5.4.2.1 #define BARELOG\_BUF\_MAX\_SIZE (BARELOG\_EVENT\_MAX\_SIZE - 2\*sizeof(uint32\_t))

Maximum size (in bytes) of the string buffer inside a barelog event :

Definition at line 119 of file barelog\_internal.h.

### 5.4.2.2 #define BARELOG\_DEBUG\_MEM\_SIZE sizeof(barelog\_event\_t)

Size (in bytes) taken by all data used by the debug mode

Definition at line 99 of file barelog internal.h.

## 5.4.2.3 #define BARELOG\_DEBUG\_MODE\_I (BARELOG\_NB\_CORES + BARELOG\_SAFE\_MODE)

Index of the debug mode in the mem\_space hierarchy

Definition at line 101 of file barelog\_internal.h.

## 5.4.2.4 #define BARELOG\_DEBUG\_OFF BARELOG\_SAFE\_MEM\_SIZE

Offset in the shared memory of the beginning of the debug mode section

Definition at line 103 of file barelog\_internal.h.

28 File Documentation

5.4.2.5 #define BARELOG\_ERR -1

General error return code

Definition at line 49 of file barelog internal.h.

5.4.2.6 #define BARELOG\_EVENT\_CONVERSION\_ERR -7

Event conversion error return code

Definition at line 61 of file barelog internal.h.

5.4.2.7 #define BARELOG\_EVENT\_PER\_CORE\_MAX (BARELOG\_LOCAL\_MEM\_PER\_CORE/BARELOG\_EVENT\_← MAX\_SIZE)

Maximum number of events manageable locally per core :

Definition at line 122 of file barelog\_internal.h.

5.4.2.8 #define BARELOG\_EVENT\_PER\_CORE\_SHR\_MEM\_MAX (BARELOG\_SHARED\_MEM\_PER\_CORE\_MAX/BA← RELOG\_EVENT\_MAX\_SIZE)

Maximum number of events manageable in shared memory per core :

Definition at line 128 of file barelog\_internal.h.

5.4.2.9 #define BARELOG\_HOST\_NB\_MEM\_SPACE (BARELOG\_NB\_CORES + BARELOG\_SAFE\_MODE + BARELOG\_DEBUG\_MODE)

Number of used barelog\_mem\_space\_t in the host manager :

Definition at line 131 of file barelog\_internal.h.

5.4.2.10 #define BARELOG\_INCONSISTENT\_PARAM\_ERR -3

Inconsistent parameter error return code

Definition at line 53 of file barelog\_internal.h.

5.4.2.11 #define BARELOG\_INIT\_ERR -8

Barelog initialization error return code

Definition at line 63 of file barelog\_internal.h.

5.4.2.12 #define BARELOG\_MUTEX\_TRY\_MAX 5

Number of tries to do in order to get a mutex

Definition at line 80 of file barelog\_internal.h.

5.4.2.13 #define BARELOG\_NB\_MUTEX\_BYTES BARELOG\_NB\_CORES

Size (in bytes) taken by the mutexes in shared memory

Definition at line 85 of file barelog\_internal.h.

### 5.4.2.14 #define BARELOG\_SAFE\_MEM\_SIZE BARELOG\_NB\_MUTEX\_BYTES

Size (in bytes) taken by all data used by the safe mode

Definition at line 87 of file barelog\_internal.h.

### 5.4.2.15 #define BARELOG\_SAFE\_MODE\_I BARELOG\_NB\_CORES

Index of the safe mode in the mem\_space hierarchy

Definition at line 89 of file barelog\_internal.h.

# 5.4.2.16 #define BARELOG\_SHARED\_MEM\_DATA\_OFFSET (BARELOG\_NB\_MUTEX\_BYTES + BARELOG\_DEBUG\_MEM\_SIZE)

Defines the offset (in bytes) to use to access the events part in the shared memory. It corresponds to the reserved size at the beginning of the allowed shared memory used for barelog's settings such as synchronization flags.

Definition at line 113 of file barelog\_internal.h.

# 5.4.2.17 #define BARELOG\_SHARED\_MEM\_MAX (BARELOG\_EVENT\_SHARED\_MEM\_MAX + BARELOG\_SHARED\_MEM\_DATA\_OFFSET)

Maximum size (in bytes) taken in the shared memory by barelog data

Definition at line 116 of file barelog\_internal.h.

# 5.4.2.18 #define BARELOG\_SHARED\_MEM\_PER\_CORE\_MAX (BARELOG\_EVENT\_SHARED\_MEM\_MAX/BARELO ← G\_NB\_CORES)

Size (in bytes) of each shared memory area reserved per core :

Definition at line 125 of file barelog\_internal.h.

5.4.2.19 #define barelog\_shrmem\_mutex\_t uint8\_t

barelog mutex type

Definition at line 71 of file barelog internal.h.

## 5.4.2.20 #define BARELOG\_SHRMEM\_READ\_ERR -5

Shared memory reading error return code

Definition at line 57 of file barelog\_internal.h.

## 5.4.2.21 #define BARELOG\_SHRMEM\_WRITE\_ERR -4

Shared memory writing error return code

Definition at line 55 of file barelog\_internal.h.

## 5.4.2.22 #define BARELOG\_SUCCESS 0

Success return code

Definition at line 47 of file barelog\_internal.h.

### 5.4.2.23 #define BARELOG\_TIMEOUT\_ERR -6

Timeout expired error return code

Definition at line 59 of file barelog\_internal.h.

## 5.4.2.24 #define BARELOG\_UNINITIALIZED\_PARAM\_ERR -2

Unitialized parameter error return code

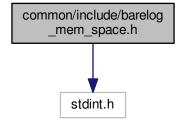
Definition at line 51 of file barelog\_internal.h.

## 5.5 common/include/barelog\_mem\_space.h File Reference

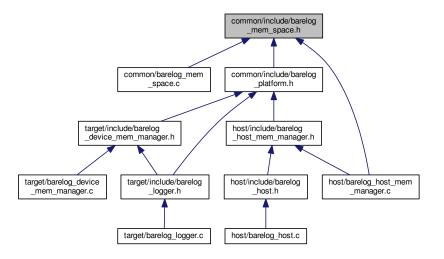
Module defining mem\_space structure.

#include <stdint.h>

Include dependency graph for barelog\_mem\_space.h:



This graph shows which files directly or indirectly include this file:



## **Macros**

- #define BARELOG\_WORD 1
- #define BARELOG\_DOUBLE\_WORD 2
- #define BARELOG HALF WORD (1/2)
- #define BARELOG BYTE 0

#### **Functions**

• struct \_\_attribute\_\_ ((packed, aligned))

## **Variables**

- · barelog mem space t
- const barelog\_mem\_space\_t MEM\_SPACE\_INITIALIZER

## 5.5.1 Detailed Description

Module defining mem\_space structure.

This header defines the structure of what will be called a mem space. It represents a chunk of the shared memory.

#### **Author**

Thomas Bertauld

Date

17/10/2015

## 5.5.2 Function Documentation

```
5.5.2.1 struct __attribute__ ( (packed, aligned) )
```

Main structure of a mem space, representing a chunk of the shared memory, physical address

(possibly) virtual address (the one used by memcpy on the target of execution)

length of the memory space

prefered alignment of data inside this memory space (reserved for future use)

size of words inside this memory space (reserved for future use)

field used to store any return value of the shared memory initialization function

Definition at line 49 of file barelog\_mem\_space.h.

## 5.5.3 Variable Documentation

5.5.3.1 const barelog\_mem\_space\_t MEM\_SPACE\_INITIALIZER

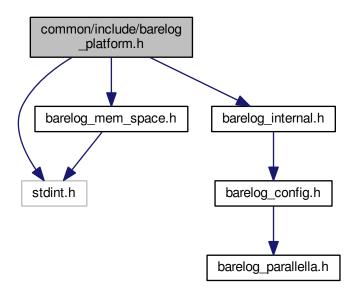
mem\_space initializer.

Definition at line 26 of file barelog\_mem\_space.c.

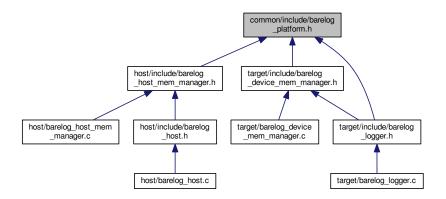
## 5.6 common/include/barelog\_platform.h File Reference

Module defining a platform to use barelog against.

```
#include <stdint.h>
#include "barelog_mem_space.h"
#include "barelog_internal.h"
Include dependency graph for barelog_platform.h:
```



This graph shows which files directly or indirectly include this file:



## **Data Structures**

struct barelog\_platform\_t

## 5.6.1 Detailed Description

Module defining a platform to use barelog against.

Author

Thomas Bertauld

Date

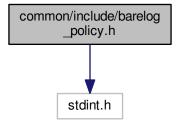
17/10/2015

## 5.7 common/include/barelog\_policy.h File Reference

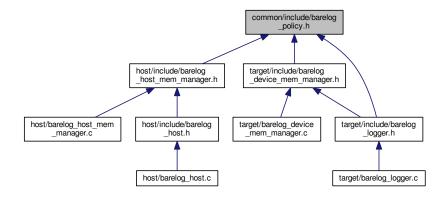
Module defining the different policies that can be used when an events buffer is full.

```
#include <stdint.h>
```

Include dependency graph for barelog\_policy.h:



This graph shows which files directly or indirectly include this file:



## **Enumerations**

enum barelog\_policy\_t { SKIP, REPLACE, FLUSH, DESTROY }

## 5.7.1 Detailed Description

Module defining the different policies that can be used when an events buffer is full.

**Author** 

Thomas Bertauld

Date

17/10/2015

## 5.7.2 Enumeration Type Documentation

## 5.7.2.1 enum barelog\_policy\_t

Enum of all the policies available.

**Enumerator** 

SKIP When buffer full, ignore new events.

**REPLACE** When buffer full, replace with new events.

**FLUSH** When buffer full, flush it to shared memory.

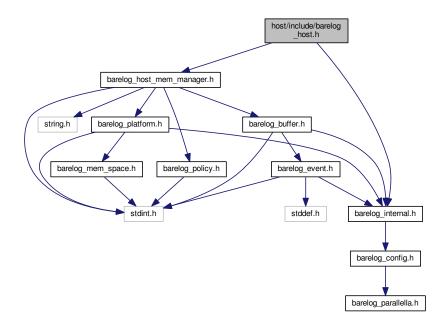
**DESTROY** Destroy buffer when full.

Definition at line 41 of file barelog policy.h.

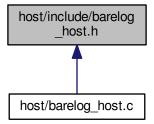
## 5.8 host/include/barelog\_host.h File Reference

Module providing some nice wrapping for the host\_mem\_manager.

```
#include "barelog_host_mem_manager.h"
#include "barelog_internal.h"
Include dependency graph for barelog host.h:
```



This graph shows which files directly or indirectly include this file:



### **Macros**

- #define barelog\_host\_init(platform, initfct, readfct, writefct, finalizefct) host\_mem\_manager\_init(platform, initfct, readfct, writefct, finalizefct)
- #define barelog\_host\_finalize() host\_mem\_manager\_finalize()
- #define barelog\_read\_log(core, res) host\_mem\_manager\_read\_mem\_space(core, res)
- #define barelog\_read\_debug() host\_mem\_manager\_read\_debug()

## 5.8.1 Detailed Description

Module providing some nice wrapping for the host\_mem\_manager.

Only this module should be used by the host program.

Author

Thomas Bertauld

Date

17/10/2015

## 5.8.2 Macro Definition Documentation

5.8.2.1 #define barelog\_host\_finalize( ) host\_mem\_manager\_finalize()

See also

host\_mem\_manager\_finalize

Definition at line 50 of file barelog\_host.h.

5.8.2.2 #define barelog\_host\_init( platform, initfct, readfct, writefct, finalizefct ) host\_mem\_manager\_init(platform, initfct, readfct, writefct, finalizefct)

See also

host mem manager init

Definition at line 44 of file barelog\_host.h.

5.8.2.3 #define barelog\_read\_debug( ) host\_mem\_manager\_read\_debug()

See also

host\_mem\_manager\_read\_debug

Definition at line 61 of file barelog\_host.h.

5.8.2.4 #define barelog\_read\_log( core, res ) host mem manager read mem space(core, res)

See also

host\_mem\_manager\_read\_mem\_space

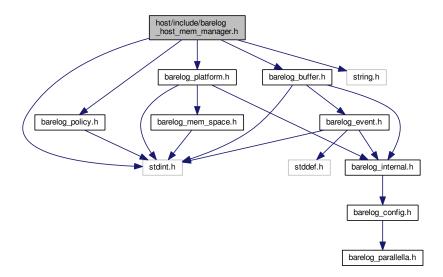
Definition at line 55 of file barelog\_host.h.

## 5.9 host/include/barelog\_host\_mem\_manager.h File Reference

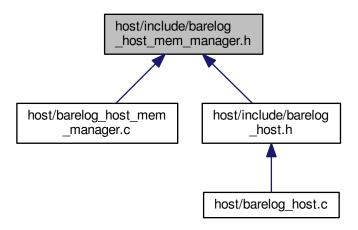
Module defining all functions offered by barelog for the host program.

```
#include <stdint.h>
#include <string.h>
#include "barelog_platform.h"
#include "barelog_buffer.h"
#include "barelog_policy.h"
```

Include dependency graph for barelog\_host\_mem\_manager.h:



This graph shows which files directly or indirectly include this file:



## **Data Structures**

struct barelog\_host\_mem\_manager\_t

## **Functions**

- int8\_t host\_mem\_manager\_init (const barelog\_platform\_t platform, void \*(\*init)(void \*address, size\_t size, void \*data), int8\_t(\*read)(const void \*address, size\_t size, void \*buffer), int8\_t(\*write)(void \*address, size\_t size, const void \*buffer), int8\_t(\*finalize)(void \*mem\_space)) \_\_attribute\_\_((cold))
- int8\_t host\_mem\_manager\_finalize (void) \_\_attribute\_\_((cold
- int32\_t host\_mem\_manager\_read\_mem\_space (uint32\_t core, barelog\_event\_t \*\*events)
- int8 t host mem manager read debug (void)

## **Variables**

• int8\_t destructor

## 5.9.1 Detailed Description

Module defining all functions offered by barelog for the host program.

This header defines the functions structures and functions used to initialize and finalize the host part of the logger and to read the events inside the shared memory once the logging session is over.

## **Author**

Thomas Bertauld

## Date

24/11/2015

### 5.9.2 Function Documentation

5.9.2.1 int8\_t host\_mem\_manager\_finalize ( void )

Finalizes the host's memory manager. Deallocate all previously allocated (shared) memory segments.

#### Returns

BARELOG\_NB\_CORES on success. Otherwise if ret > 0, it is the number of memory segments correctly deallocated and if ret < 0 it indicates an error code.

5.9.2.2 int8\_t host\_mem\_manager\_init ( const barelog\_platform\_t platform, void \*(\*)(void \*address, size\_t size, void \*data) init, int8\_t(\*)(const void \*address, size\_t size, void \*buffer) read, int8\_t(\*)(void \*address, size\_t size, const void \*buffer) write, int8\_t(\*)(void \*mem\_space) finalize )

Initializes the host's memory manager. Should be called before any subsequent call to any other function in this module.

### **Parameters**

platform	the platform to allocate the (shared) memory spaces against.
init	the function used by the host to initialize a memory section.
read	the function used by the host to read data from a memory section.
write	the function used by the host to write data into a memory section.
finalize	the function used by the host to deallocate a (shared) memory space.

### Returns

BARELOG\_NB\_CORES on success. Otherwise if ret > 0, it is the number of memory segments correctly allocated and if ret < 0 it is an error code.

Definition at line 78 of file barelog\_host\_mem\_manager.c.

5.9.2.3 int8\_t host\_mem\_manager\_read\_debug ( void )

Function used to read and display on stderr the shared memory error section (if applicable).

#### See also

barelog\_debug\_log

#### Returns

BARELOG SUCCESS if everything went well, an error code otherwise.

Definition at line 226 of file barelog\_host\_mem\_manager.c.

5.9.2.4 int32\_t host\_mem\_manager\_read\_mem\_space ( uint32\_t core, barelog\_event\_t \*\* events )

Reads the memory section dedicated to a core and returns the corresponding events buffer. WARNING: it is the responsibility of the caller to free this buffer afterwards.

#### **Parameters**

core	the core on which to read the events.
------	---------------------------------------

### Returns

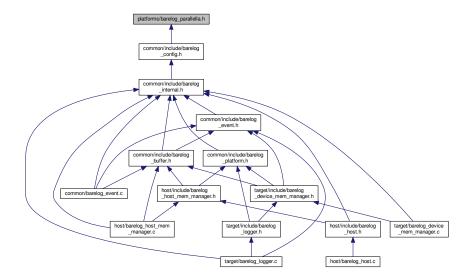
the number of events read from shared memory.

Definition at line 185 of file barelog\_host\_mem\_manager.c.

## 5.10 platforms/barelog\_parallella.h File Reference

Module defining the configurations used by barelog specifically for the Parallella platform.

This graph shows which files directly or indirectly include this file:



## **Macros**

- #define BARELOG\_NB\_CORES 16
- #define BARELOG\_EVENT\_SHARED\_MEM\_MAX 1000000
- #define BARELOG\_PLATFORM\_NAME\_LENGTH 20
- #define BARELOG\_EVENT\_MAX\_SIZE 100
- #define BARELOG\_LOCAL\_MEM\_PER\_CORE 1000
- #define BARELOG\_LOCAL\_MEM\_ATTRIBUTE \_\_attribute\_\_ ((section(".data\_bank0")))
- #define **BARELOG\_VERBOSE** 0
- #define **BARELOG\_SAFE\_MODE** 0
- #define BARELOG\_CHECK\_MODE 0

## 5.10.1 Detailed Description

Module defining the configurations used by barelog specifically for the Parallella platform.

This header is used to define every external parameters that we might use to configure the behavior of the application on the Parallella platform.

#### See also

```
https://www.parallella.org/
```

## **Author**

Thomas Bertauld

Date

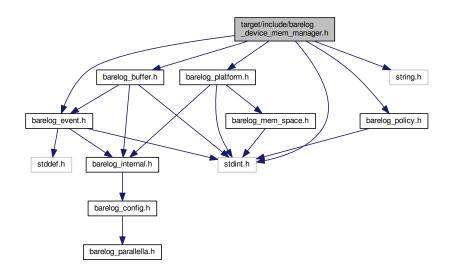
17/10/2015

## 5.11 target/include/barelog\_device\_mem\_manager.h File Reference

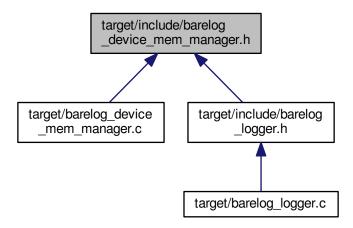
Module defining all functions offered by barelog for the host program.

```
#include <stdint.h>
#include <string.h>
#include "barelog_buffer.h"
#include "barelog_event.h"
#include "barelog_policy.h"
#include "barelog_platform.h"
```

Include dependency graph for barelog\_device\_mem\_manager.h:



This graph shows which files directly or indirectly include this file:



#### **Data Structures**

• struct barelog\_device\_mem\_manager\_t

## **Macros**

• #define BARELOG\_DEBUG(file, line, errcode, message) barelog\_debug\_log(file, line, errcode, message)

## **Functions**

- int8\_t device\_mem\_manager\_init (const uint32\_t core, const barelog\_platform\_t platform, const barelog\_← policy\_t buffer\_policy, const barelog\_policy\_t memory\_policy, int8\_t(\*read)(const void \*address, size\_t size, void \*buffer), int8\_t(\*write)(void \*address, size\_t size, const void \*buffer)) \_\_attribute\_\_((cold))
- int8\_t device\_mem\_manager\_clean\_buffer (void)
- int8\_t device\_mem\_manager\_clean (uint32\_t n)
- int8\_t device\_mem\_manager\_write\_buffer (barelog\_event\_t event) \_\_attribute\_\_((hot))
- int8\_t device\_mem\_manager\_flush\_buffer (void)
- int8\_t device\_mem\_manager\_flush (uint32\_t n)
- int8\_t device\_mem\_manager\_clean\_memory (void)
- int8\_t device\_mem\_manager\_is\_buffer\_full (void)
- void <a href="barelog\_debug\_log">barelog\_debug\_log</a> (char \*file, int line, int8\_t errcode, const char \*message)

## 5.11.1 Detailed Description

Module defining all functions offered by barelog for the host program.

This header defines the functions structures and functions used to initialize and finalize the host part of the logger and to read the events inside the shared memory once the logging session is over.

#### Author

Thomas Bertauld

Date

24/11/2015

### 5.11.2 Function Documentation

5.11.2.1 void barelog debug log ( char \* file, int line, int8 t errcode, const char \* message )

Internal function used for debugging purposes: writes the latest errcode with full description into the shared memory.

Note that for obvious debugging reasons, this functions doesn't call any other functions in the barelog's modules and use only memopy to interact with the shared memory, thus disregarding the manager.read function.

## See also

host mem manager read debug

#### **Parameters**

file	the file in which the error occurred (usually FILE).
line	the line on which the error occurred (usually <b>LINE</b> ).
errcode	the error code to return.
message	a description message to go along with the error code.

Definition at line 82 of file barelog\_device\_mem\_manager.c.

5.11.2.2 int8\_t device\_mem\_manager\_clean ( uint32\_t n )

Discards the events from the oldest one to n further events in the local buffer of the calling core.

#### **Parameters**

n number of events to discard.	

#### Returns

BARELOG\_SUCCESS on success, an error code if an error occurs.

Definition at line 224 of file barelog\_device\_mem\_manager.c.

5.11.2.3 int8\_t device\_mem\_manager\_clean\_buffer(void) [inline]

Discards all current events in the calling core's local buffer.

## Returns

BARELOG\_SUCCESS on success or an error code in case of exception.

Definition at line 215 of file barelog\_device\_mem\_manager.c.

5.11.2.4 int8\_t device\_mem\_manager\_clean\_memory ( void )

Erases all events in the shared memory buffer.

#### Returns

BARELOG\_SUCCESS on success, an error code if something went wrong.

Definition at line 378 of file barelog\_device\_mem\_manager.c.

5.11.2.5 int8\_t device\_mem\_manager\_flush ( uint32\_t n )

Flushes all event contained in the calling core's event buffer from the older one to n events further into the corresponding shared memory section.

#### **Parameters**

n	number of events to flush.
---	----------------------------

### Returns

BARELOG SUCCESS on success, an error code if an error occurs.

Definition at line 265 of file barelog\_device\_mem\_manager.c.

5.11.2.6 int8\_t device\_mem\_manager\_flush\_buffer ( void ) [inline]

Flushes the local event buffer into the shared memory section associated to the calling core.

#### Returns

BARELOG\_SUCCESS on success, an error code if an error occurs.

Definition at line 256 of file barelog\_device\_mem\_manager.c.

5.11.2.7 int8\_t device\_mem\_manager\_init ( const uint32\_t core, const barelog\_platform\_t platform, const barelog\_policy\_t buffer\_policy, const barelog\_policy\_t memory\_policy, int8\_t(\*)(const void \*address, size\_t size, void \*buffer) read, int8\_t(\*)(void \*address, size\_t size, const void \*buffer) write )

Defines and initializes the device memory manager. Should be called before any subsequent call to any other function in this module.

#### **Parameters**

core	index of the core to initialize.
platform	platform used to log (the device memory manager will be created against this platform infor-
	mation).
buffer_policy	policy to use when the events buffer is full.
memory_policy	policy to use when the shared memory buffer is full.
read	function used by device to read in shared memory.
write	function used by device to write in shared memory.

#### Returns

BARELOG\_NB\_CORES on success, an error code in case of exception.

Definition at line 100 of file barelog\_device\_mem\_manager.c.

5.11.2.8 int8\_t device\_mem\_manager\_is\_buffer\_full ( void )

Indicates whether or not the local events buffer is full (i.e we can possibly override older events, depending on the used policy).

## Returns

1 if the buffer is full, 0 otherwise.

Definition at line 388 of file barelog\_device\_mem\_manager.c.

5.11.2.9 int8\_t device\_mem\_manager\_write\_buffer ( barelog\_event\_t event )

Writes an event into the local event buffer of the calling core.

#### **Parameters**

event	the event to write.
-------	---------------------

### Returns

BARELOG\_SUCCESS on success, an error code if an error occurs.

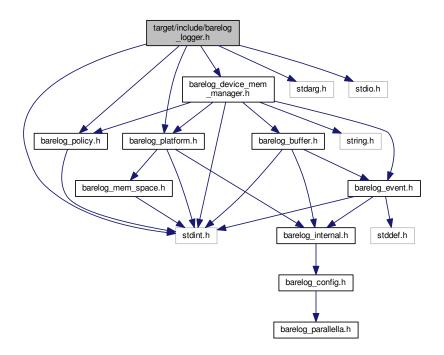
Definition at line 154 of file barelog device mem manager.c.

#### target/include/barelog\_logger.h File Reference 5.12

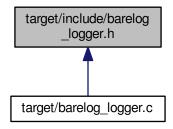
Module providing some nice wrapping for the device\_mem\_manager.

```
#include <stdint.h>
#include <stdarg.h>
#include <stdio.h>
#include "barelog_platform.h"
#include "barelog_policy.h"
#include "barelog_device_mem_manager.h"
```

Include dependency graph for barelog\_logger.h:



This graph shows which files directly or indirectly include this file:



#### **Data Structures**

· struct barelog logger t

## **Macros**

- #define BARELOG DEFAULT LOG LVL BARELOG INFO LVL
- #define barelog\_clean\_buffer() device\_mem\_manager\_clean\_buffer()
- #define barelog\_clean(n) device\_mem\_manager\_clean(n)
- #define barelog\_flush\_buffer() device\_mem\_manager\_flush\_buffer()
- #define barelog\_flush(n) device\_mem\_manager\_flush(n)
- #define barelog\_is\_buffer\_full() device\_mem\_manager\_is\_buffer\_full()
- #define barelog\_clean\_memory() device\_mem\_manager\_clean\_memory()

### **Enumerations**

```
    enum barelog_lvl_t {
        BARELOG_OFF = 0, BARELOG_CRITICAL_LVL, BARELOG_ERROR_LVL, BARELOG_WARNING_L
        VL,
        BARELOG_DEBUG_LVL, BARELOG_INFO_LVL }
```

## **Functions**

- int8\_t barelog\_init\_logger (const uint32\_t my\_core, const barelog\_platform\_t platform, const barelog\_policy
   \_t buffer\_policy, const barelog\_policy\_t memory\_policy, int8\_t(\*read)(const void \*address, size\_t size, void
   \*buffer), int8\_t(\*write)(void \*address, size\_t size, const void \*buffer), uint32\_t(\*get\_clock)(void), int8\_t(\*init
   \_clock)(void), int8\_t(\*start\_clock)(void)) \_\_attribute\_\_((cold))
- int8\_t barelog\_start (void) \_\_attribute\_\_((cold))
- int8\_t barelog\_log (barelog\_lvl\_t lvl, const char \*format,...) \_\_attribute\_\_((hot))
- int8\_t barelog\_immediate\_log (barelog\_lvl\_t lvl, const char \*format,...) \_\_attribute\_\_((hot))
- void barelog\_set\_log\_lvl (barelog\_lvl\_t lvl)
- barelog\_lvl\_t barelog\_get\_log\_lvl (void)

## 5.12.1 Detailed Description

Module providing some nice wrapping for the device\_mem\_manager.

Only this module should be used by the target program.

**Author** 

Thomas Bertauld

Date

24/11/2015

### 5.12.2 Macro Definition Documentation

```
5.12.2.1 #define barelog_clean(n) device mem manager clean(n)
```

See also

```
device_mem_manager_clean
```

Definition at line 146 of file barelog\_logger.h.

```
5.12.2.2 #define barelog_clean_buffer( ) device_mem_manager_clean_buffer()
```

See also

```
device_mem_manager_clean_buffer
```

Definition at line 141 of file barelog logger.h.

```
5.12.2.3 #define barelog_clean_memory( ) device_mem_manager_clean_memory()
```

See also

```
device_mem_manager_clean_memory
```

Definition at line 166 of file barelog\_logger.h.

```
5.12.2.4 #define barelog_flush( n ) device_mem_manager_flush(n)
```

See also

```
device_mem_manager_flush
```

Definition at line 156 of file barelog\_logger.h.

```
5.12.2.5 #define barelog_flush_buffer( ) device mem_manager_flush_buffer()
```

See also

```
device_mem_manager_flush_buffer
```

Definition at line 151 of file barelog\_logger.h.

5.12.2.6 #define barelog\_is\_buffer\_full( ) device\_mem\_manager\_is\_buffer\_full()

See also

device mem manager is buffer full

Definition at line 161 of file barelog\_logger.h.

## 5.12.3 Enumeration Type Documentation

5.12.3.1 enum barelog\_lvl\_t

Defines the different possible logging levels.

Definition at line 48 of file barelog\_logger.h.

### 5.12.4 Function Documentation

```
5.12.4.1 int8_t barelog_immediate_log ( barelog_lvl_t /v/, const char * format, ... )
```

Does the same thing as barelog\_log but flushes directly the computed event and cleans the corresponding buffer.

See also

barelog\_log

Definition at line 96 of file barelog\_logger.c.

5.12.4.2 int8\_t barelog\_init\_logger ( const uint32\_t my\_core, const barelog\_platform\_t platform, const barelog\_policy\_t buffer\_policy, const barelog\_policy\_t memory\_policy, int8\_t(\*)(const void \*address, size\_t size, void \*buffer) read, int8\_t(\*)(void \*address, size\_t size, const void \*buffer) write, uint32\_t(\*)(void) get\_clock, int8\_t(\*)(void) init\_clock, int8\_t(\*)(void) start\_clock)

Initializes the logger. Should be called before any subsequent call to any other function in this module.

## **Parameters**

my_core	index of the core to log.
platform	the platform to allocate the (shared) memory spaces against.
buffer_policy	policy to apply when the events buffer is full.
memory_policy	policy to apply when the memory buffer is full.
read	the function used by the target to read data from a memory section.
write	the function used by the target to write data into a memory section.
get_clock	the function used to retrieve timestamps.
init_clock	the function used to initialize the target's clock.
start_clock	the function used to start the target's clock.

## Returns

BARELOG\_SUCCESS on success.

Definition at line 38 of file barelog\_logger.c.

```
5.12.4.3 int8_t barelog_log ( barelog_lvl_t lvl, const char * format, ... )
```

The logging function, follows the same format than printf(). If a real and functional get\_clock() function was given upon initialization, it will be used to automatically add a timestamp to the created event containing the message.

## **Parameters**

IvI	the log-level of the event.
format	the event's data formatting string, followed, if needed, by the corresponding data values.

Definition at line 75 of file barelog\_logger.c.

5.12.4.4 int8\_t barelog\_start ( void )

Starts the logging engine. Should be called before any subsequent call to the barelog\_log() function.

Returns

BARELOG\_SUCCESS on success, or an error code if something went wrong.

Definition at line 62 of file barelog\_logger.c.

# Index

attribute	BARELOG_SHARED_MEM_DATA_OFFSET
barelog_event.h, 24	barelog_internal.h, 29
barelog_mem_space.h, 31	BARELOG_SHARED_MEM_MAX
	barelog_internal.h, 29
BARELOG_BUF_MAX_SIZE	BARELOG_SHARED_MEM_PER_CORE_MAX
barelog_internal.h, 27	barelog_internal.h, 29
BARELOG_CHECK_MODE	BARELOG_SHRMEM_READ_ERR
barelog_config.h, 22	barelog_internal.h, 29
BARELOG_DEBUG_MEM_SIZE	BARELOG_SHRMEM_WRITE_ERR
barelog_internal.h, 27	barelog_internal.h, 29
BARELOG_DEBUG_MODE_I	BARELOG_SUCCESS
barelog_internal.h, 27	barelog_internal.h, 29
BARELOG_DEBUG_OFF	BARELOG TIMEOUT ERR
barelog_internal.h, 27	barelog_internal.h, 29
BARELOG_ERR	BARELOG_UNINITIALIZED_PARAM_ERR
barelog_internal.h, 27	barelog_internal.h, 30
BARELOG_EVENT_CONVERSION_ERR	barelog_clean
barelog_internal.h, 28	barelog_logger.h, 47
BARELOG_EVENT_INITIALIZER	barelog_clean_buffer
barelog_event.h, 25	barelog_logger.h, 47
BARELOG_EVENT_MAX_SIZE	barelog_clean_memory
barelog_config.h, 22	barelog_logger.h, 47
BARELOG_EVENT_PER_CORE_MAX	barelog_logger.n, 47 barelog_config.h
barelog_internal.h, 28	BARELOG_CHECK_MODE, 22
BARELOG_EVENT_PER_CORE_SHR_MEM_MAX	BARELOG_EVENT_MAX_SIZE, 22
barelog_internal.h, 28	BARELOG EVENT SHARED MEM MAX, 22
BARELOG_EVENT_SHARED_MEM_MAX	BARELOG LOCAL MEM ATTRIBUTE, 22
barelog_config.h, 22	BARELOG LOCAL MEM PER CORE, 22
BARELOG_HOST_NB_MEM_SPACE	
barelog_internal.h, 28	BARELOG_NB_CORES, 22
BARELOG_INCONSISTENT_PARAM_ERR	BARELOG_PLATFORM_NAME_LENGTH, 22
barelog_internal.h, 28	barelog_debug_log
BARELOG_INIT_ERR	barelog_device_mem_manager.h, 42
barelog_internal.h, 28	barelog_device_mem_manager.h
BARELOG_LOCAL_MEM_ATTRIBUTE	barelog_debug_log, 42
barelog_config.h, 22	device_mem_manager_clean, 42
BARELOG_LOCAL_MEM_PER_CORE	device_mem_manager_clean_buffer, 42
barelog_config.h, 22	device_mem_manager_clean_memory, 42
BARELOG_MUTEX_TRY_MAX	device_mem_manager_flush, 42
barelog_internal.h, 28	device_mem_manager_flush_buffer, 44
BARELOG_NB_CORES	device_mem_manager_init, 44
barelog_config.h, 22	device_mem_manager_is_buffer_full, 44
BARELOG_NB_MUTEX_BYTES	device_mem_manager_write_buffer, 44
barelog_internal.h, 28	barelog_device_mem_manager_t, 9
BARELOG_PLATFORM_NAME_LENGTH	read, 10
barelog_config.h, 22	write, 10
BARELOG_SAFE_MEM_SIZE	barelog_event.h
barelog_internal.h, 28	attribute, 24
BARELOG_SAFE_MODE_I	BARELOG_EVENT_INITIALIZER, 25
barelog_internal.h, 29	barelog_event_to_string, 24

52 INDEX

barelog_events_to_strings, 25	BARELOG_SHARED_MEM_PER_CORE_MAX,
EVENT_TO_STRING_SIZE, 24	29
barelog_event_buffer_t, 10	BARELOG_SHRMEM_READ_ERR, 29
buffer, 11	BARELOG_SHRMEM_WRITE_ERR, 29
empty, 11	BARELOG_SUCCESS, 29
full, 11	BARELOG_TIMEOUT_ERR, 29
head, 11	BARELOG_UNINITIALIZED_PARAM_ERR, 30
tail, 11	barelog_shrmem_mutex_t, 29
barelog_event_to_string	barelog_is_buffer_full
barelog_event.h, 24	barelog_logger.h, 47
barelog_events_to_strings	barelog_log
barelog_event.h, 25	barelog_logger.h, 48
barelog_flush	barelog_logger.h
barelog_logger.h, 47	barelog_clean, 47
barelog_flush_buffer	barelog_clean_buffer, 47
barelog_logger.h, 47	barelog_clean_memory, 47
barelog_host.h	barelog flush, 47
barelog_host_finalize, 35	barelog_flush_buffer, 47
barelog_host_init, 35	barelog_immediate_log, 48
barelog read debug, 35	barelog_init_logger, 48
barelog_read_log, 36	barelog is buffer full, 47
barelog host finalize	barelog_log, 48
barelog_host.h, 35	barelog_log, 40 barelog_lvl_t, 48
barelog_host_init	barelog_start, 49
barelog_host.h, 35	barelog_logger_t, 14
<del>-</del>	
barelog_host_mem_manager.h	get_clock, 14 init_clock, 14
host_mem_manager_finalize, 38	
host_mem_manager_init, 38	start_clock, 14
host_mem_manager_read_debug, 38	barelog_lvl_t
host_mem_manager_read_mem_space, 38	barelog_logger.h, 48
barelog_host_mem_manager_t, 11	barelog_mem_space.h
finalize, 12	attribute, 31
init, 12	MEM_SPACE_INITIALIZER, 31
read, 12	barelog_platform_t, 15
write, 12	mem_space, 15
barelog_immediate_log	name, 15
barelog_logger.h, 48	barelog_policy.h
barelog_init_logger	barelog_policy_t, 34
barelog_logger.h, 48	DESTROY, 34
barelog_internal.h	FLUSH, 34
BARELOG_BUF_MAX_SIZE, 27	REPLACE, 34
BARELOG_DEBUG_MEM_SIZE, 27	SKIP, 34
BARELOG_DEBUG_MODE_I, 27	barelog_policy_t
BARELOG_DEBUG_OFF, 27	barelog_policy.h, 34
BARELOG_ERR, 27	barelog_read_debug
BARELOG_EVENT_CONVERSION_ERR, 28	barelog_host.h, 35
BARELOG_EVENT_PER_CORE_MAX, 28	barelog_read_log
$BARELOG\_EVENT\_PER\_CORE\_SHR\_MEM\_{\leftarrow}$	barelog_host.h, 36
MAX, 28	barelog_result_buffer_t, 15
BARELOG_HOST_NB_MEM_SPACE, 28	buffer, 16
BARELOG_INCONSISTENT_PARAM_ERR, 28	buffer_length, 16
BARELOG_INIT_ERR, 28	sub_buffer_length, 16
BARELOG_MUTEX_TRY_MAX, 28	barelog_shared_mem_buffer_t, 16
BARELOG_NB_MUTEX_BYTES, 28	events, 17
BARELOG_SAFE_MEM_SIZE, 28	imax, 17
BARELOG_SAFE_MODE_I, 29	index, 17
BARELOG_SHARED_MEM_DATA_OFFSET, 29	barelog_shrmem_mutex_t
BARELOG_SHARED_MEM_MAX, 29	barelog_internal.h, 29

INDEX 53

barelog_start	host_mem_manager_read_debug
barelog_logger.h, 49	barelog_host_mem_manager.h, 38
buffer	host_mem_manager_read_mem_space
barelog_event_buffer_t, 11	barelog_host_mem_manager.h, 38
barelog_result_buffer_t, 16	to and
buffer_length	imax
barelog_result_buffer_t, 16	barelog_shared_mem_buffer_t, 17
	index
common/include/barelog_buffer.h, 19	barelog_shared_mem_buffer_t, 17
common/include/barelog_config.h, 21	init
common/include/barelog_event.h, 23	barelog_host_mem_manager_t, 12
common/include/barelog_internal.h, 25	init_clock
common/include/barelog_mem_space.h, 30	barelog_logger_t, 14
common/include/barelog_platform.h, 32	
common/include/barelog_policy.h, 33	MEM_SPACE_INITIALIZER
	barelog_mem_space.h, 31
DESTROY	mem_space
barelog_policy.h, 34	barelog_platform_t, 15
device_mem_manager_clean	
barelog_device_mem_manager.h, 42	name
device_mem_manager_clean_buffer	barelog_platform_t, 15
barelog_device_mem_manager.h, 42	
device_mem_manager_clean_memory	platforms/barelog_parallella.h, 39
barelog_device_mem_manager.h, 42	
device_mem_manager_flush	REPLACE
barelog_device_mem_manager.h, 42	barelog_policy.h, 34
device_mem_manager_flush_buffer	read
barelog_device_mem_manager.h, 44	barelog_device_mem_manager_t, 10
device_mem_manager_init	barelog_host_mem_manager_t, 12
barelog_device_mem_manager.h, 44	
device_mem_manager_is_buffer_full	SKIP
	barelog_policy.h, 34
barelog_device_mem_manager.h, 44	start_clock
device_mem_manager_write_buffer	barelog_logger_t, 14
barelog_device_mem_manager.h, 44	sub_buffer_length
EVENT_TO_STRING_SIZE	barelog_result_buffer_t, 16
barelog_event.h, 24	tail
empty	barelog_event_buffer_t, 11
barelog_event_buffer_t, 11	target/include/barelog_device_mem_manager.h, 40
events	target/include/barelog_logger.h, 45
barelog_shared_mem_buffer_t, 17	3_ 35
FLUSH	write
	barelog_device_mem_manager_t, 10
barelog_policy.h, 34	barelog_host_mem_manager_t, 12
finalize	
barelog_host_mem_manager_t, 12	
full	
barelog_event_buffer_t, 11	
get_clock	
barelog_logger_t, 14	
hood	
head	
barelog_event_buffer_t, 11	
host/include/barelog_host.h, 34	
host/include/barelog_host_mem_manager.h, 36	
host_mem_manager_finalize	
barelog_host_mem_manager.h, 38	
host_mem_manager_init	
barelog_host_mem_manager.h, 38	