barelog

0.1

Generated by Doxygen 1.8.9.1

Sat Oct 17 2015 16:57:15

Contents

1 Main Page **1**

2 Data Structure Index **5**

2.1 Data Structures 5

3 File Index **7**

3.1 File List 7

4 Data Structure Documentation **9**

4.1 barelog_device_mem_manager_t Struct Reference 9

4.1.1 Detailed Description 9

4.1.2 Field Documentation 9

4.1.2.1 read 9

4.1.2.2 write 9

4.2 barelog_event_buffer_t Struct Reference 10

4.2.1 Detailed Description 10

4.2.2 Field Documentation 10

4.2.2.1 buffer 10

4.2.2.2 empty 10

4.2.2.3 full 10

4.2.2.4 head 11

4.2.2.5 tail 11

4.3 barelog_host_mem_manager_t Struct Reference 11

4.3.1 Detailed Description 11

4.3.2 Field Documentation 11

4.3.2.1 finalize 11

4.3.2.2 init 11

4.3.2.3 read 12

4.3.2.4 write 12

4.4 barelog_logger_t Struct Reference 12

4.4.1 Detailed Description 13

4.4.2 Field Documentation 13

4.4.2.1 get_clock 13
4.4.2.2 init_clock 13
4.4.2.3 start_clock 13
4.5 barelog_platform_t Struct Reference 13
4.5.1 Detailed Description 13
4.5.2 Field Documentation 14
4.5.2.1 mem_space 14
4.5.2.2 name 14
4.6 barelog_result_buffer_t Struct Reference 14
4.6.1 Detailed Description 14
4.6.2 Field Documentation 14
4.6.2.1 buffer 14
4.6.2.2 buffer_length 14
4.6.2.3 sub_buffer_length 14
4.7 barelog_shared_mem_buffer_t Struct Reference 15
4.7.1 Detailed Description 15
4.7.2 Field Documentation 15
4.7.2.1 events 15
4.7.2.2 imax 15
4.7.2.3 index 15

5 File Documentation **17**
5.1 common/include/barelog_buffer.h File Reference 17
5.1.1 Detailed Description 18
5.2 common/include/barelog_config.h File Reference 19
5.2.1 Detailed Description 20
5.2.2 Macro Definition Documentation 20
5.2.2.1 BARELOG_CHECK_MODE 20
5.2.2.2 BARELOG_EVENT_MAX_SIZE 20
5.2.2.3 BARELOG_EVENT_SHARED_MEM_MAX 20
5.2.2.4 BARELOG_LOCAL_MEM_ATTRIBUTE 20
5.2.2.5 BARELOG_LOCAL_MEM_PER_CORE 20
5.2.2.6 BARELOG_NB_CORES 20
5.2.2.7 BARELOG_PLATFORM_NAME_LENGTH 20
5.3 common/include/barelog_event.h File Reference 21
5.3.1 Detailed Description 22
5.3.2 Macro Definition Documentation 22
5.3.2.1 EVENT_TO_STRING_SIZE 22
5.3.3 Function Documentation 22
5.3.3.1 __attribute__ 22

5.3.3.2 barelog_event_to_string 22
5.3.3.3 barelog_events_to_strings 23
5.3.4 Variable Documentation 23
5.3.4.1 BARELOG_EVENT_INITIALIZER 23
5.4 common/include/barelog_internal.h File Reference 23
5.4.1 Detailed Description 25
5.4.2 Macro Definition Documentation 25
5.4.2.1 BARELOG_BUF_MAX_SIZE 25
5.4.2.2 BARELOG_DEBUG_MEM_SIZE 25
5.4.2.3 BARELOG_DEBUG_MODE_I 25
5.4.2.4 BARELOG_DEBUG_OFF 25
5.4.2.5 BARELOG_ERR 26
5.4.2.6 BARELOG_EVENT_CONVERSION_ERR 26
5.4.2.7 BARELOG_EVENT_PER_CORE_MAX 26
5.4.2.8 BARELOG_EVENT_PER_CORE_SHR_MEM_MAX 26
5.4.2.9 BARELOG_HOST_NB_MEM_SPACE 26
5.4.2.10 BARELOG_INCONSISTENT_PARAM_ERR 26
5.4.2.11 BARELOG_INIT_ERR 26
5.4.2.12 BARELOG_MUTEX_TRY_MAX 26
5.4.2.13 BARELOG_NB_MUTEX_BYTES 26
5.4.2.14 BARELOG_SAFE_MEM_SIZE 27
5.4.2.15 BARELOG_SAFE_MODE_I 27
5.4.2.16 BARELOG_SHARED_MEM_DATA_OFFSET 27
5.4.2.17 BARELOG_SHARED_MEM_MAX 27
5.4.2.18 BARELOG_SHARED_MEM_PER_CORE_MAX 27
5.4.2.19 barelog_shrmem_mutex_t 27
5.4.2.20 BARELOG_SHRMEM_READ_ERR 27
5.4.2.21 BARELOG_SHRMEM_WRITE_ERR 27
5.4.2.22 BARELOG_SUCCESS 27
5.4.2.23 BARELOG_TIMEOUT_ERR 28
5.4.2.24 BARELOG_UNINITIALIZED_PARAM_ERR 28
5.5 common/include/barelog_mem_space.h File Reference 28
5.5.1 Detailed Description 29
5.5.2 Function Documentation 29
5.5.2.1 __attribute__ 29
5.5.3 Variable Documentation 29
5.5.3.1 MEM_SPACE_INITIALIZER 29
5.6 common/include/barelog_platform.h File Reference 30
5.6.1 Detailed Description 31
5.7 common/include/barelog_policy.h File Reference 31

5.7.1 Detailed Description 32
5.7.2 Enumeration Type Documentation 32
5.7.2.1 barelog_policy_t 32
5.8 host/include/barelog_host.h File Reference 32
5.8.1 Detailed Description 33
5.8.2 Macro Definition Documentation 33
5.8.2.1 barelog_host_finalize 33
5.8.2.2 barelog_host_init 33
5.8.2.3 barelog_read_debug 34
5.8.2.4 barelog_read_log 34
5.9 host/include/barelog_host_mem_manager.h File Reference 34
5.9.1 Detailed Description 35
5.9.2 Function Documentation 36
5.9.2.1 host_mem_manager_finalize 36
5.9.2.2 host_mem_manager_init 36
5.9.2.3 host_mem_manager_read_debug 36
5.9.2.4 host_mem_manager_read_mem_space 36
5.10 platforms/barelog_parallella.h File Reference 37
5.10.1 Detailed Description 37
5.11 target/include/barelog_device_mem_manager.h File Reference 38
5.11.1 Detailed Description 39
5.11.2 Function Documentation 40
5.11.2.1 barelog_debug_log 40
5.11.2.2 device_mem_manager_clean 40
5.11.2.3 device_mem_manager_clean_buffer 40
5.11.2.4 device_mem_manager_clean_memory 40
5.11.2.5 device_mem_manager_flush 41
5.11.2.6 device_mem_manager_flush_buffer 42
5.11.2.7 device_mem_manager_init 42
5.11.2.8 device_mem_manager_is_buffer_full 42
5.11.2.9 device_mem_manager_write_buffer 42
5.12 target/include/barelog_logger.h File Reference 43
5.12.1 Detailed Description 44
5.12.2 Macro Definition Documentation 45
5.12.2.1 barelog_clean 45
5.12.2.2 barelog_clean_buffer 45
5.12.2.3 barelog_clean_memory 45
5.12.2.4 barelog_flush 45
5.12.2.5 barelog_flush_buffer 45
5.12.2.6 barelog_is_buffer_full 45

5.12.3 Function Documentation 45
5.12.3.1 barelog_init_logger 45
5.12.3.2 barelog_log 46
5.12.3.3 barelog_start 46

Index **47**

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.

Using

Compiling the modules

1. 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 :

1. Initialize the host: you will have to create the **barelog_platform_t** along with some memory management functions and to register them to the logger on the host by calling the **barelog_host_init()** 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.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

barelog_device_mem_manager_t 9
barelog_event_buffer_t 10
barelog_host_mem_manager_t 11
barelog_logger_t 12
barelog_platform_t 13
barelog_result_buffer_t 14
barelog_shared_mem_buffer_t 15

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 17
common/include/barelog_config.h
Module defining the configurations used by barelog 19
common/include/barelog_event.h
Module defining the events and their related functions 21
common/include/barelog_internal.h
Module defining the internal configurations of barelog 23
common/include/barelog_mem_space.h
Module defining mem_space structure 28
common/include/barelog_platform.h
Module defining a platform to use barelog against 30
common/include/barelog_policy.h
Module defining the different policies that can be used when an events buffer is full 31
host/**barelog_host.c** . . . . ??
host/**barelog_host_mem_manager.c** . . . . ??
host/include/barelog_host.h
Module providing some nice wrapping for the host_mem_manager 32
host/include/barelog_host_mem_manager.h
Module defining all functions offered by barelog for the host program 34
platforms/barelog_parallella.h
Module defining the configurations used by barelog specifically for the Parallella platform . . . 37
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 38
target/include/barelog_logger.h
Module providing some nice wrapping for the device_mem_manager 43
target/include/**barelog_snprintf.h** . . . . ??

Chapter 4

Data Structure Documentation

4.1 barelog_device_mem_manager_t Struct Reference

```
#include <barelog_device_mem_manager.h>
```

Data Fields

- int8_t(∗ read)(const void ∗address, size_t size, void ∗buffer)
- int8_t(∗ write)(void ∗address, size_t size, const void ∗buffer)
- barelog_policy_t **buffer_policy**
- barelog_policy_t **memory_policy**

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 59 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 66 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

- 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 80 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 61 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 68 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 75 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

- 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 50 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 54 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 58 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 62 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

4.7.2.1 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

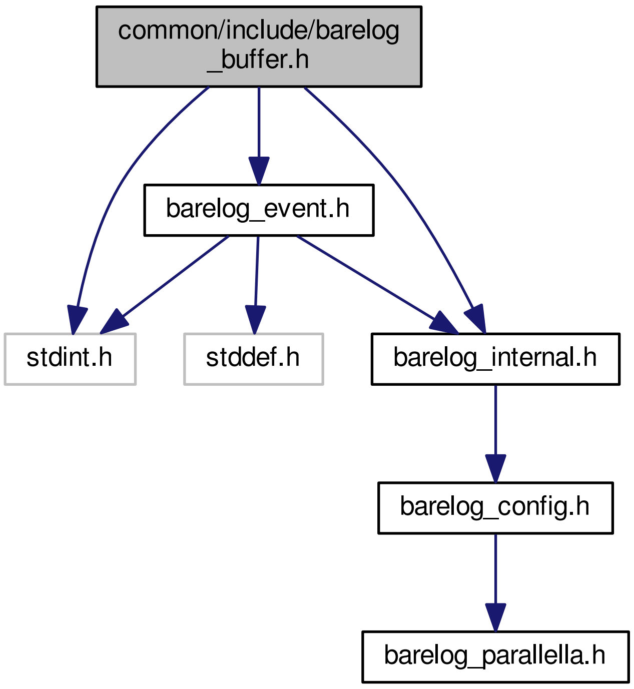
File Documentation

5.1 common/include/barelog_buffer.h File Reference

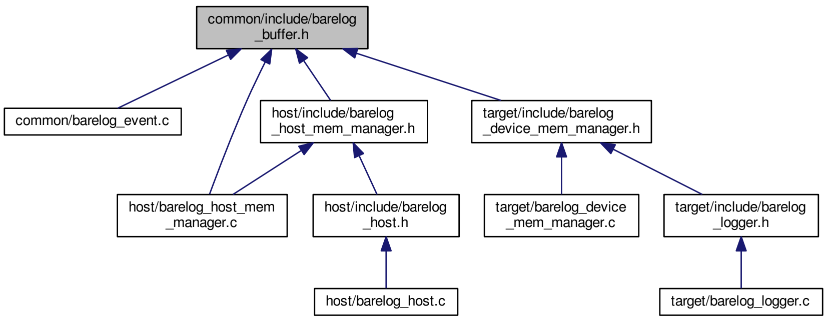
Module defining the different buffers used by barelog's internals.

```
#include <stdint.h>
#include "barelog_internal.h"
#include "barelog_event.h"
```

Include dependency graph for barelog_buffer.h:



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


common/include/barelog_buffer.h
common/barelog_event.c
host/include/barelog_host_mem_manager.h
target/include/barelog_device_mem_manager.h
host/barelog_host_mem_manager.c
host/include/barelog_host.h
target/barelog_device_mem_manager.c
target/include/barelog_logger.h
host/barelog_host.c
target/barelog_logger.c

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

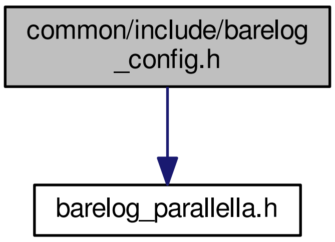
Date

17/10/2015

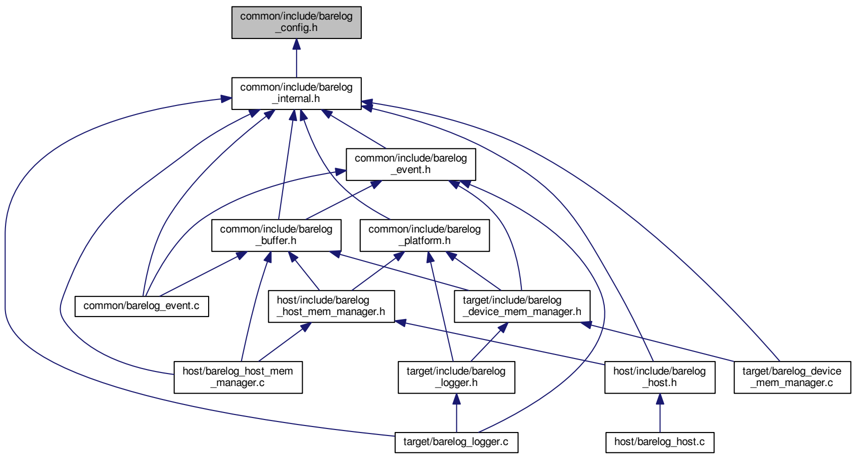
5.2 common/include/barelog_config.h File Reference

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:


common/include/barelog_config.h
common/include/barelog_internal.h
common/include/barelog_event.h
common/include/barelog_buffer.h
common/include/barelog_platform.h
common/barelog_event.c
host/include/barelog_host_mem_manager.h
target/include/barelog_device_mem_manager.h
host/barelog_host_mem_manager.c
target/include/barelog_logger.h
host/include/barelog_host.h
target/barelog_device_mem_manager.c
target/barelog_logger.c
host/barelog_host.c

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

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) :

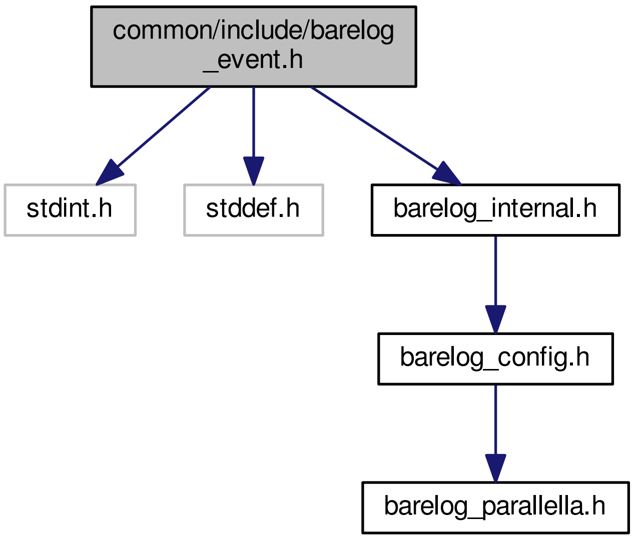
Definition at line 57 of file barelog_config.h.

5.3 common/include/barelog_event.h File Reference

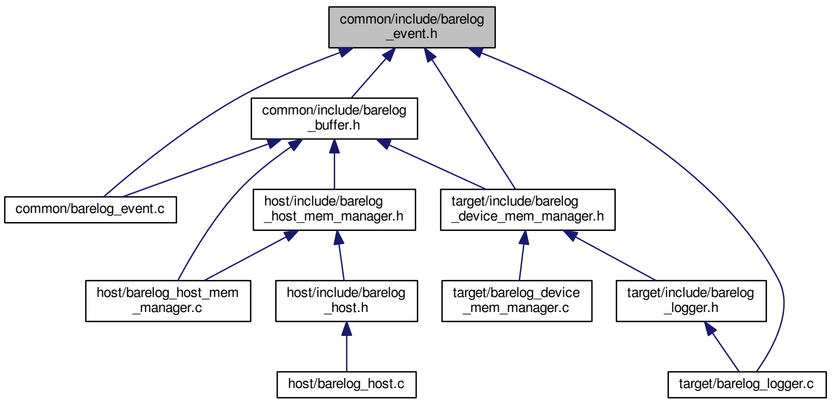
Module defining the events and their related functions.

```
#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:


common/include/barelog_event.h
common/include/barelog_buffer.h
common/barelog_event.c
host/include/barelog_host_mem_manager.h
target/include/barelog_device_mem_manager.h
host/barelog_host_mem_manager.c
host/include/barelog_host.h
target/barelog_device_mem_manager.c
target/include/barelog_logger.h
host/barelog_host.c
target/barelog_logger.c

Macros

- #define EVENT_TO_STRING_SIZE BARELOG_EVENT_MAX_SIZE∗2

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

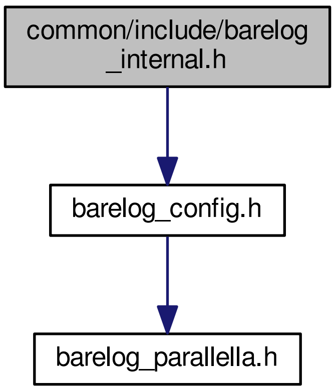
Event initializer, every field is set to 0 except for data, set to "".

Definition at line 33 of file barelog_event.c.

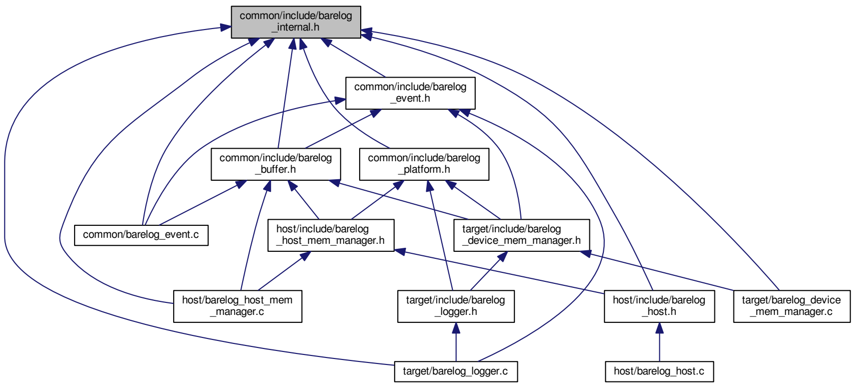
5.4 common/include/barelog_internal.h File Reference

Module defining the internal configurations of barelog.

```
#include "barelog_config.h"
```
Include dependency graph for barelog_internal.h:



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


common/include/barelog_internal.h
common/include/barelog_event.h
common/include/barelog_buffer.h
common/include/barelog_platform.h
common/barelog_event.c
host/include/barelog_host_mem_manager.h
target/include/barelog_device_mem_manager.h
host/barelog_host_mem_manager.c
target/include/barelog_logger.h
host/include/barelog_host.h
target/barelog_device_mem_manager.c
target/barelog_logger.c
host/barelog_host.c

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_SHARED_MEM_DATA_OFFSET (BARELOG_NB_MUTEX_BYTES + BARELOG_DE↩ BUG_MEM_SIZE)
- #define BARELOG_SHARED_MEM_MAX (BARELOG_EVENT_SHARED_MEM_MAX + BARELOG_SHA↩ RED_MEM_DATA_OFFSET)
- #define BARELOG_BUF_MAX_SIZE (BARELOG_EVENT_MAX_SIZE - 2∗sizeof(uint32_t))
- #define BARELOG_EVENT_PER_CORE_MAX (BARELOG_LOCAL_MEM_PER_CORE/BARELOG_EV↩ ENT_MAX_SIZE)
- #define BARELOG_SHARED_MEM_PER_CORE_MAX (BARELOG_EVENT_SHARED_MEM_MAX/BAR↩ ELOG_NB_CORES)
- #define BARELOG_EVENT_PER_CORE_SHR_MEM_MAX (BARELOG_SHARED_MEM_PER_CORE_↩ MAX/BARELOG_EVENT_MAX_SIZE)
- #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.

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

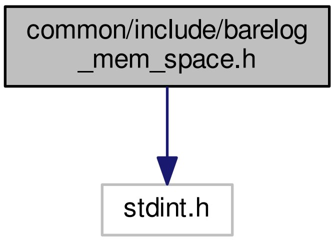
Definition at line 51 of file barelog_internal.h.

5.5 common/include/barelog_mem_space.h File Reference

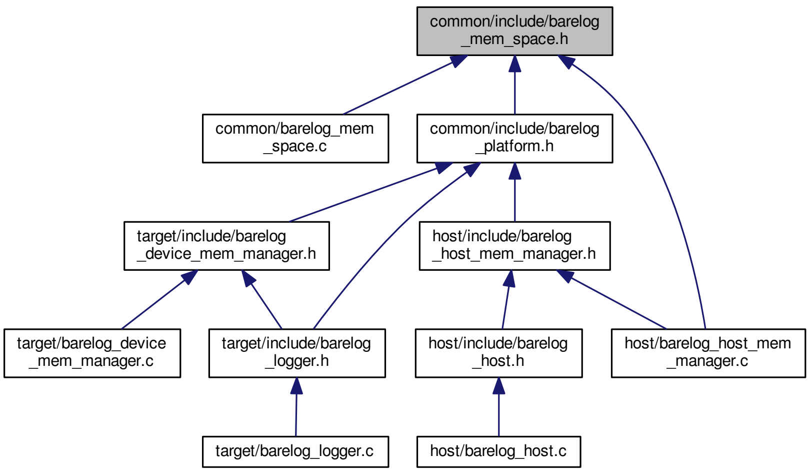
Module defining mem_space structure.

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

Include dependency graph for barelog_mem_space.h:



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


common/include/barelog_mem_space.h
common/barelog_mem_space.c
common/include/barelog_platform.h
target/include/barelog_device_mem_manager.h
host/include/barelog_host_mem_manager.h
target/barelog_device_mem_manager.c
target/include/barelog_logger.h
host/include/barelog_host.h
host/barelog_host_mem_manager.c
target/barelog_logger.c
host/barelog_host.c

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.

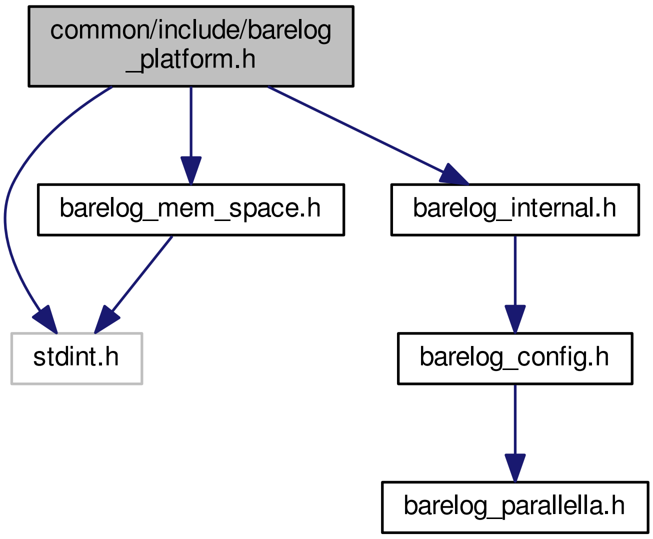
Definition at line 26 of file barelog_mem_space.c.

5.6 common/include/barelog_platform.h File Reference

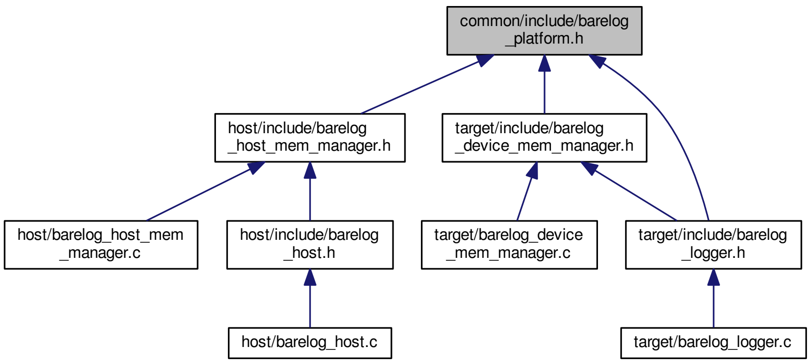
Module defining a platform to use barelog against.

```
#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:


common/include/barelog_platform.h
host/include/barelog_host_mem_manager.h
target/include/barelog_device_mem_manager.h
host/barelog_host_mem_manager.c
host/include/barelog_host.h
target/barelog_device_mem_manager.c
target/include/barelog_logger.h
host/barelog_host.c
target/barelog_logger.c

Data Structures

- struct barelog_platform_t

5.6.1 Detailed Description

Module defining a platform to use barelog against.

Author

Thomas Bertauld

Date

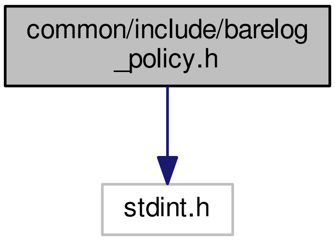
17/10/2015

5.7 common/include/barelog_policy.h File Reference

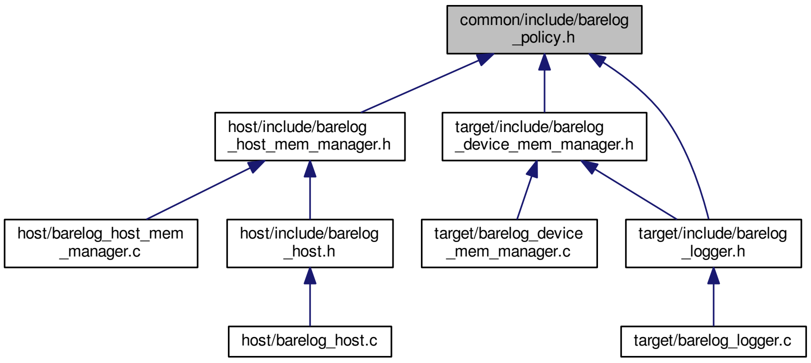
Module defining the different policies that can be used when an events buffer is full.

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

Include dependency graph for barelog_policy.h:



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


common/include/barelog_policy.h
host/include/barelog_host_mem_manager.h
target/include/barelog_device_mem_manager.h
host/barelog_host_mem_manager.c
host/include/barelog_host.h
target/barelog_device_mem_manager.c
target/include/barelog_logger.h
host/barelog_host.c
target/barelog_logger.c

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.

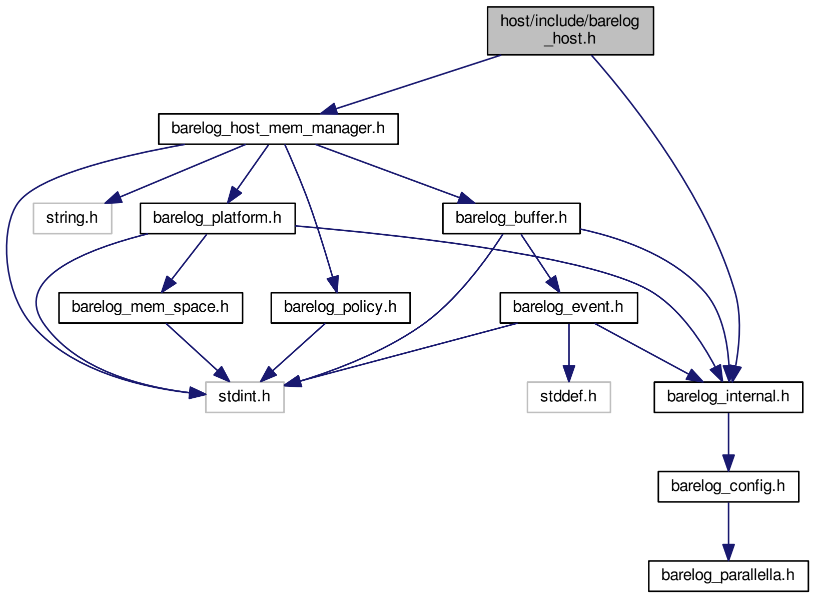
Definition at line 41 of file barelog_policy.h.

5.8 host/include/barelog_host.h File Reference

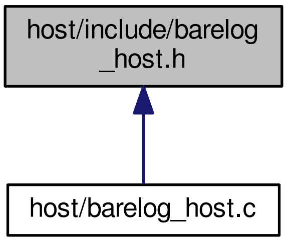
Module providing some nice wrapping for the host_mem_manager.

```
#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

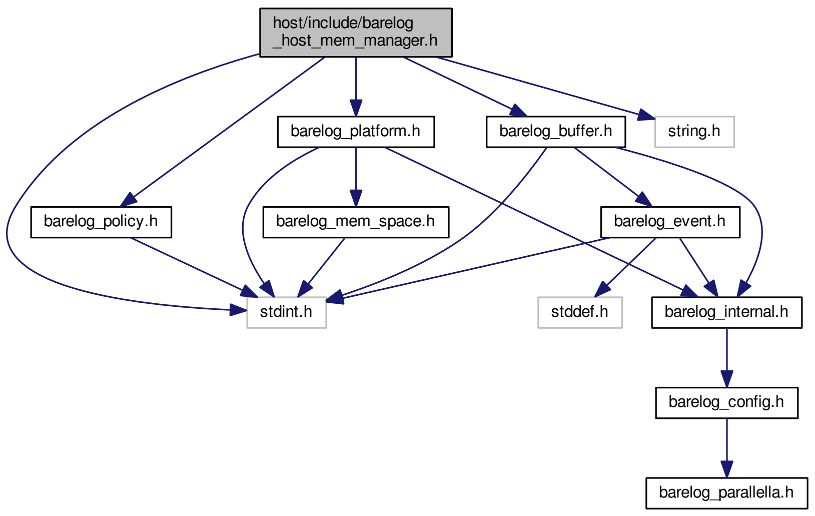
Definition at line 55 of file barelog_host.h.

5.9 host/include/barelog_host_mem_manager.h File Reference

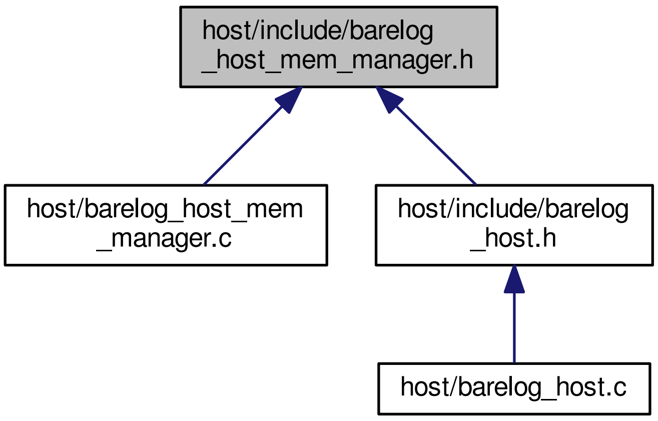
Module defining all functions offered by barelog for the host program.

```
#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

17/10/2015

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 87 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 235 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

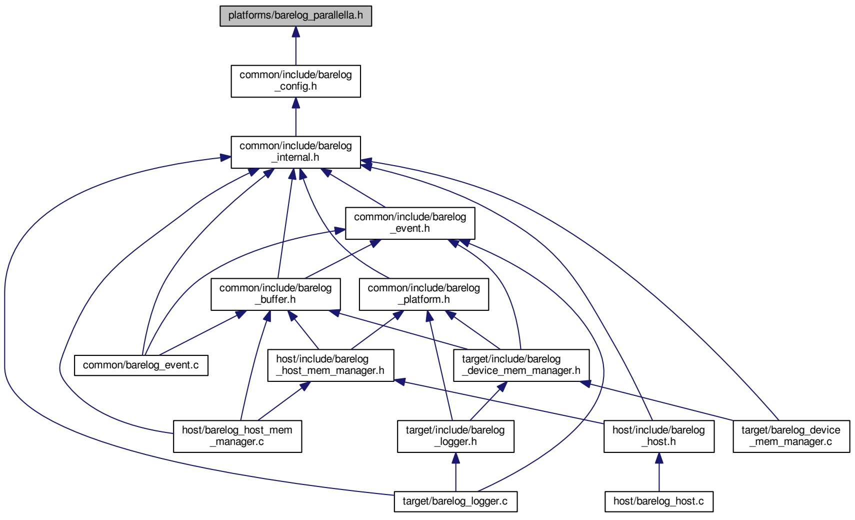
the number of events read from shared memory.

Definition at line 194 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:


platforms/barelog_parallella.h
common/include/barelog _config.h
common/include/barelog _internal.h
common/include/barelog _event.h
common/include/barelog _buffer.h
common/include/barelog _platform.h
common/barelog_event.c
host/include/barelog _host_mem_manager.h
target/include/barelog _device_mem_manager.h
host/barelog_host_mem _manager.c
target/include/barelog _logger.h
host/include/barelog _host.h
target/barelog_device _mem_manager.c
target/barelog_logger.c
host/barelog_host.c

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

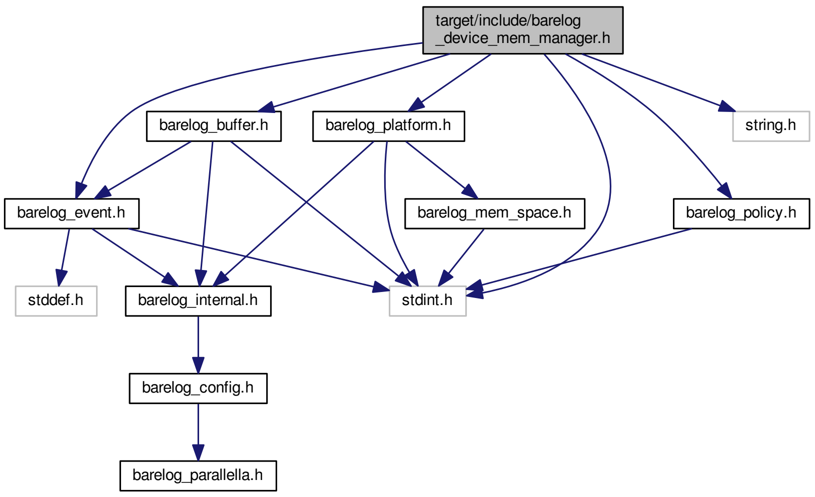
17/10/2015

5.11 target/include/barelog_device_mem_manager.h File Reference

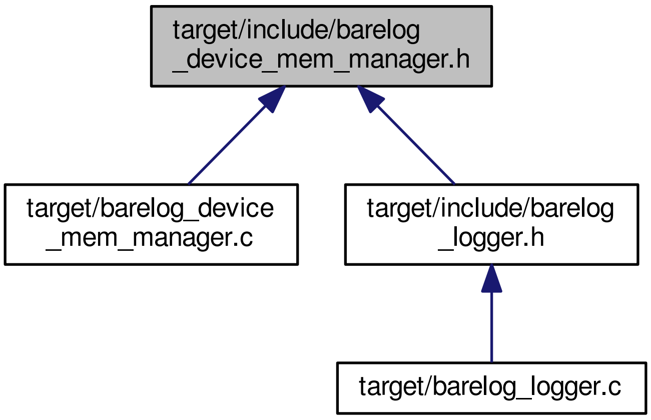
Module defining all functions offered by barelog for the host program.

```
#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:


target/include/barelog_device_mem_manager.h
target/barelog_device_mem_manager.c
target/include/barelog_logger.h
target/barelog_logger.c

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 barelog_debug_log (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

17/10/2015

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 memcpy 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 **LINE**).
errcode	the error code to return.
message	a description message to go along with the error code.

Definition at line 88 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 229 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 220 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 383 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 270 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 261 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 information).
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 106 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 393 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.

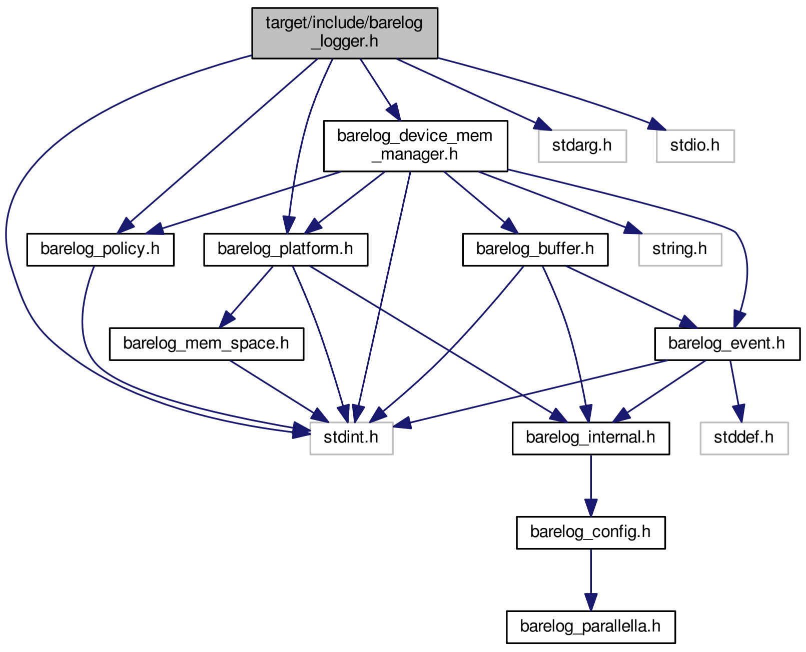
Definition at line 159 of file barelog_device_mem_manager.c.

5.12 target/include/barelog_logger.h File Reference

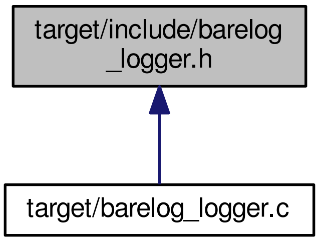
Module providing some nice wrapping for the device_mem_manager.

```
#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_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()

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 (const char ∗format,...) __attribute__((hot))

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

17/10/2015

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 114 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 109 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 134 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 124 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 119 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 129 of file barelog_logger.h.

5.12.3 Function Documentation

5.12.3.1 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.

5.12.3.2 int8_t barelog_log (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.

Definition at line 75 of file barelog_logger.c.

5.12.3.3 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 61 of file barelog_logger.c.

Index

__attribute__
barelog_event.h, 22
barelog_mem_space.h, 29

BARELOG_BUF_MAX_SIZE
barelog_internal.h, 25
BARELOG_CHECK_MODE
barelog_config.h, 20
BARELOG_DEBUG_MEM_SIZE
barelog_internal.h, 25
BARELOG_DEBUG_MODE_I
barelog_internal.h, 25
BARELOG_DEBUG_OFF
barelog_internal.h, 25
BARELOG_ERR
barelog_internal.h, 25
BARELOG_EVENT_CONVERSION_ERR
barelog_internal.h, 26
BARELOG_EVENT_INITIALIZER
barelog_event.h, 23
BARELOG_EVENT_MAX_SIZE
barelog_config.h, 20
BARELOG_EVENT_PER_CORE_MAX
barelog_internal.h, 26
BARELOG_EVENT_PER_CORE_SHR_MEM_MAX
barelog_internal.h, 26
BARELOG_EVENT_SHARED_MEM_MAX
barelog_config.h, 20
BARELOG_HOST_NB_MEM_SPACE
barelog_internal.h, 26
BARELOG_INCONSISTENT_PARAM_ERR
barelog_internal.h, 26
BARELOG_INIT_ERR
barelog_internal.h, 26
BARELOG_LOCAL_MEM_ATTRIBUTE
barelog_config.h, 20
BARELOG_LOCAL_MEM_PER_CORE
barelog_config.h, 20
BARELOG_MUTEX_TRY_MAX
barelog_internal.h, 26
BARELOG_NB_CORES
barelog_config.h, 20
BARELOG_NB_MUTEX_BYTES
barelog_internal.h, 26
BARELOG_PLATFORM_NAME_LENGTH
barelog_config.h, 20
BARELOG_SAFE_MEM_SIZE
barelog_internal.h, 26
BARELOG_SAFE_MODE_I
barelog_internal.h, 27
BARELOG_SHARED_MEM_DATA_OFFSET
barelog_internal.h, 27
BARELOG_SHARED_MEM_MAX
barelog_internal.h, 27
BARELOG_SHARED_MEM_PER_CORE_MAX
barelog_internal.h, 27
BARELOG_SHRMEM_READ_ERR
barelog_internal.h, 27
BARELOG_SHRMEM_WRITE_ERR
barelog_internal.h, 27
BARELOG_SUCCESS
barelog_internal.h, 27
BARELOG_TIMEOUT_ERR
barelog_internal.h, 27
BARELOG_UNINITIALIZED_PARAM_ERR
barelog_internal.h, 28
barelog_clean
barelog_logger.h, 45
barelog_clean_buffer
barelog_logger.h, 45
barelog_clean_memory
barelog_logger.h, 45
barelog_config.h
BARELOG_CHECK_MODE, 20
BARELOG_EVENT_MAX_SIZE, 20
BARELOG_EVENT_SHARED_MEM_MAX, 20
BARELOG_LOCAL_MEM_ATTRIBUTE, 20
BARELOG_LOCAL_MEM_PER_CORE, 20
BARELOG_NB_CORES, 20
BARELOG_PLATFORM_NAME_LENGTH, 20
barelog_debug_log
barelog_device_mem_manager.h, 40
barelog_device_mem_manager.h
barelog_debug_log, 40
device_mem_manager_clean, 40
device_mem_manager_clean_buffer, 40
device_mem_manager_clean_memory, 40
device_mem_manager_flush, 40
device_mem_manager_flush_buffer, 42
device_mem_manager_init, 42
device_mem_manager_is_buffer_full, 42
device_mem_manager_write_buffer, 42
barelog_device_mem_manager_t, 9
read, 9
write, 9
barelog_event.h
__attribute__, 22
BARELOG_EVENT_INITIALIZER, 23
barelog_event_to_string, 22

barelog_events_to_strings, 23
EVENT_TO_STRING_SIZE, 22
barelog_event_buffer_t, 10
buffer, 10
empty, 10
full, 10
head, 10
tail, 11
barelog_event_to_string
barelog_event.h, 22
barelog_events_to_strings
barelog_event.h, 23
barelog_flush
barelog_logger.h, 45
barelog_flush_buffer
barelog_logger.h, 45
barelog_host.h
barelog_host_finalize, 33
barelog_host_init, 33
barelog_read_debug, 33
barelog_read_log, 34
barelog_host_finalize
barelog_host.h, 33
barelog_host_init
barelog_host.h, 33
barelog_host_mem_manager.h
host_mem_manager_finalize, 36
host_mem_manager_init, 36
host_mem_manager_read_debug, 36
host_mem_manager_read_mem_space, 36
barelog_host_mem_manager_t, 11
finalize, 11
init, 11
read, 12
write, 12
barelog_init_logger
barelog_logger.h, 45
barelog_internal.h
BARELOG_BUF_MAX_SIZE, 25
BARELOG_DEBUG_MEM_SIZE, 25
BARELOG_DEBUG_MODE_I, 25
BARELOG_DEBUG_OFF, 25
BARELOG_ERR, 25
BARELOG_EVENT_CONVERSION_ERR, 26
BARELOG_EVENT_PER_CORE_MAX, 26
BARELOG_EVENT_PER_CORE_SHR_MEM_↩
MAX, 26
BARELOG_HOST_NB_MEM_SPACE, 26
BARELOG_INCONSISTENT_PARAM_ERR, 26
BARELOG_INIT_ERR, 26
BARELOG_MUTEX_TRY_MAX, 26
BARELOG_NB_MUTEX_BYTES, 26
BARELOG_SAFE_MEM_SIZE, 26
BARELOG_SAFE_MODE_I, 27
BARELOG_SHARED_MEM_DATA_OFFSET, 27
BARELOG_SHARED_MEM_MAX, 27
BARELOG_SHARED_MEM_PER_CORE_MAX,
27
BARELOG_SHRMEM_READ_ERR, 27
BARELOG_SHRMEM_WRITE_ERR, 27
BARELOG_SUCCESS, 27
BARELOG_TIMEOUT_ERR, 27
BARELOG_UNINITIALIZED_PARAM_ERR, 28
barelog_shrmem_mutex_t, 27
barelog_is_buffer_full
barelog_logger.h, 45
barelog_log
barelog_logger.h, 46
barelog_logger.h
barelog_clean, 45
barelog_clean_buffer, 45
barelog_clean_memory, 45
barelog_flush, 45
barelog_flush_buffer, 45
barelog_init_logger, 45
barelog_is_buffer_full, 45
barelog_log, 46
barelog_start, 46
barelog_logger_t, 12
get_clock, 13
init_clock, 13
start_clock, 13
barelog_mem_space.h
__attribute__, 29
MEM_SPACE_INITIALIZER, 29
barelog_platform_t, 13
mem_space, 14
name, 14
barelog_policy.h
barelog_policy_t, 32
DESTROY, 32
FLUSH, 32
REPLACE, 32
SKIP, 32
barelog_policy_t
barelog_policy.h, 32
barelog_read_debug
barelog_host.h, 33
barelog_read_log
barelog_host.h, 34
barelog_result_buffer_t, 14
buffer, 14
buffer_length, 14
sub_buffer_length, 14
barelog_shared_mem_buffer_t, 15
events, 15
imax, 15
index, 15
barelog_shrmem_mutex_t
barelog_internal.h, 27
barelog_start
barelog_logger.h, 46
buffer
barelog_event_buffer_t, 10
barelog_result_buffer_t, 14
buffer_length

barelog_result_buffer_t, 14

common/include/barelog_buffer.h, 17
common/include/barelog_config.h, 19
common/include/barelog_event.h, 21
common/include/barelog_internal.h, 23
common/include/barelog_mem_space.h, 28
common/include/barelog_platform.h, 30
common/include/barelog_policy.h, 31

DESTROY
barelog_policy.h, 32
device_mem_manager_clean
barelog_device_mem_manager.h, 40
device_mem_manager_clean_buffer
barelog_device_mem_manager.h, 40
device_mem_manager_clean_memory
barelog_device_mem_manager.h, 40
device_mem_manager_flush
barelog_device_mem_manager.h, 40
device_mem_manager_flush_buffer
barelog_device_mem_manager.h, 42
device_mem_manager_init
barelog_device_mem_manager.h, 42
device_mem_manager_is_buffer_full
barelog_device_mem_manager.h, 42
device_mem_manager_write_buffer
barelog_device_mem_manager.h, 42

EVENT_TO_STRING_SIZE
barelog_event.h, 22
empty
barelog_event_buffer_t, 10
events
barelog_shared_mem_buffer_t, 15

FLUSH
barelog_policy.h, 32
finalize
barelog_host_mem_manager_t, 11
full
barelog_event_buffer_t, 10

get_clock
barelog_logger_t, 13

head
barelog_event_buffer_t, 10
host/include/barelog_host.h, 32
host/include/barelog_host_mem_manager.h, 34
host_mem_manager_finalize
barelog_host_mem_manager.h, 36
host_mem_manager_init
barelog_host_mem_manager.h, 36
host_mem_manager_read_debug
barelog_host_mem_manager.h, 36
host_mem_manager_read_mem_space
barelog_host_mem_manager.h, 36

imax
barelog_shared_mem_buffer_t, 15
index
barelog_shared_mem_buffer_t, 15
init
barelog_host_mem_manager_t, 11
init_clock
barelog_logger_t, 13

MEM_SPACE_INITIALIZER
barelog_mem_space.h, 29
mem_space
barelog_platform_t, 14

name
barelog_platform_t, 14

platforms/barelog_parallella.h, 37

REPLACE
barelog_policy.h, 32
read
barelog_device_mem_manager_t, 9
barelog_host_mem_manager_t, 12

SKIP
barelog_policy.h, 32
start_clock
barelog_logger_t, 13
sub_buffer_length
barelog_result_buffer_t, 14

tail
barelog_event_buffer_t, 11
target/include/barelog_device_mem_manager.h, 38
target/include/barelog_logger.h, 43

write
barelog_device_mem_manager_t, 9
barelog_host_mem_manager_t, 12