

Lock handling library

IN THIS DOCUMENT

- ▶ Hardware lock API
 - ▶ Software lock API
-

This module provides access to hardware and software locks for use in concurrent C programs. In general it is not safe to use these to marshall within XC due to the assumptions XC makes about safe concurrent data access.

Two types of locks are provided. Hardware locks are fast and power efficient but there are a limited number per tile. Software locks are slower but you can use an unlimited number of them.

1 Hardware lock API

hwlock_t

This type represents a hardware lock.

hwlock_alloc()

Allocate a hardware lock.

This function will allocate a new hardware lock from the pool of hardware locks available on the xCORE. The hardware has a limited number of hardware locks (for example, current L and S series devices have 4 locks per tile).

Type

```
hwlock_t hwlock_alloc(void)
```

Returns

the allocated lock if allocation is successful or the value `HWLOCK_NOT_ALLOCATED` if not.

hwlock_free()

Free a hardware lock.

This function frees a given hardware lock and returns it to the hardware pool to be reallocated elsewhere.

Type

```
void hwlock_free(hwlock_t lock)
```

Parameters

lock	the hardware lock to be freed. If this is an invalid lock id or not an currently allocated lock then the function will trap.
------	--

hwlock_acquire()

Acquire a hardware lock.

This function acquires a lock for the current logical core. If another core holds the lock the function will pause until the lock is released.

Type

```
void hwlock_acquire(hwlock_t lock)
```

Parameters

lock	the hardware lock to acquire
------	------------------------------

hwlock_release()

Release a hardware lock.

This function releases a lock from the current logical core. The lock should have been previously claimed by [hwlock_acquire\(\)](#).

Type

```
void hwlock_release(hwlock_t lock)
```

Parameters

lock	the hardware lock to release
------	------------------------------

2 Software lock API

swlock_t

Type that represents a software lock.

SWLOCK_INITIAL_VALUE

This define should be used to initialize a software lock e.g.

```
swlock_t my_lock = SWLOCK_INITIAL_VALUE;
```

If you initialize this way there is no need to call [swlock_init\(\)](#).

swlock_init()

Initialize a software lock.

This function will initialize a software lock for use. Note that unlike hardware locks, there is no need to allocate or free a software lock from a limited pool.

Type

```
void swlock_init(swlock_t &lock)
```

swlock_try_acquire()

Try and acquire a software lock.

This function tries to acquire a lock for the current logical core. If another core holds the lock then the function will fail and return.

Type

```
int swlock_try_acquire(swlock_t &lock)
```

Parameters

lock the software lock to acquire.

Returns

a value that is equal to `SWLOCK_NOT_ACQUIRED` if the attempt fails. Any other value indicates that the acquisition has succeeded.

swlock_acquire()

Acquire a software lock.

This function acquires a lock for the current logical core. If another core holds the lock then the function will wait until it becomes available.

Type

```
void swlock_acquire(swlock_t &lock)
```

Parameters

lock the software lock to acquire.

swlock_release()

Release a software lock.

This function releases a previously acquired software lock for other cores to use.

Type

```
void swlock_release(swlock_t &lock)
```

Parameters

`lock` the software lock to release.



Copyright © 2014, All Rights Reserved.

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.

XM-UNKNOWNNA