# Intel(R) IXP400 Software OSAL API Reference Manual

Automatically generated from sources, February 26, 2007.

*Other names and brands may be claimed as the property of others.

# Table of Contents

# Operating System Abstraction Layer (OSAL) API

This service provides a thin layer of OS dependency services.

## Defines

#define **IX_OSAL_MMU_PHYS_TO_VIRT**(physAddr)
*physical to virtual address translation*

#define **IX_OSAL_MMU_VIRT_TO_PHYS**(virtAddr)
*virtual to physical address translation*

#define **IX_OSAL_CACHE_FLUSH**(addr, size)
*cache to memory flush*

#define **IX_OSAL_CACHE_INVALIDATE**(addr, size)
*cache line invalidate*

#define **IX_OSAL_CACHE_PRELOAD**(addr, size)
*cache line preload*

#define **IX_OSAL_TIMEVAL_TO_TICKS**(tv)
*Converts ixOsalTimeVal into ticks.*

#define **IX_OSAL_TICKS_TO_TIMEVAL**(ticks, pTv)
*Converts ticks into ixOsalTimeVal.*

#define **IX_OSAL_TIMEVAL_TO_MS**(tv)
*Converts ixOsalTimeVal to milliseconds.*

#define **IX_OSAL_MS_TO_TIMEVAL**(milliseconds, pTv)
*Converts milliseconds to **IxOsalTimeval**.*

#define **IX_OSAL_TIME_EQ**(tvA, tvB)
*"equal" comparison for **IxOsalTimeval***

#define **IX_OSAL_TIME_LT**(tvA, tvB)
*"less than" comparison for **IxOsalTimeval***

#define **IX_OSAL_TIME_GT**(tvA, tvB)
*"greater than" comparison for **IxOsalTimeval***

#define **IX_OSAL_TIME_ADD**(tvA, tvB)
*"add" operator for **IxOsalTimeval***

#define **IX_OSAL_TIME_SUB**(tvA, tvB)
*"subtract" operator for **IxOsalTimeval***

#define **IX_OSAL_UDIV64_32**(dividend, divisor)
*UINT64 data type division with 32 bit divisor.*

#define **IX_OSAL_UMOD64_32**(dividend, divisor)
*UINT64 data type mod with 32 bit divisor.*

# Functions

PUBLIC
**IX_STATUS ixOsalIrqBind** (UINT32 irqLevel, **IxOsalVoidFnVoidPtr** irqHandler, void *parameter)
*Binds an interrupt handler to an interrupt level.*

PUBLIC
**IX_STATUS ixOsalIrqUnbind** (UINT32 irqLevel)
*Unbinds an interrupt handler from an interrupt level.*

PUBLIC UINT32 **ixOsalIrqLock** (void)
*Disables all interrupts.*

PUBLIC void **ixOsalIrqUnlock** (UINT32 irqEnable)
*Enables all interrupts.*

PUBLIC UINT32 **ixOsalIrqLevelSet** (UINT32 irqLevel)
*Selectively disables interrupts.*

PUBLIC void **ixOsalIrqEnable** (UINT32 irqLevel)
*Enables an interrupt level.*

PUBLIC void **ixOsalIrqDisable** (UINT32 irqLevel)
*Disables an interrupt level.*

PUBLIC void * **ixOsalMemAlloc** (UINT32 size)
*Allocates memory.*

PUBLIC void **ixOsalMemFree** (void *ptr)
*Frees memory.*

PUBLIC void * **ixOsalMemCopy** (void *dest, void *src, UINT32 count)
*Copies memory zones.*

PUBLIC void * **ixOsalMemSet** (void *ptr, UINT8 filler, UINT32 count)
*Fills a memory zone.*

PUBLIC void * **ixOsalCacheDmaMalloc** (UINT32 size)
*Allocates cache−safe memory.*

PUBLIC void **ixOsalCacheDmaFree** (void *ptr)
*Frees cache−safe memory.*

PUBLIC **ixOsalThreadCreate** (**IxOsalThread** *thread, **IxOsalThreadAttr** *threadAttr,
**IX_STATUS** **IxOsalVoidFnVoidPtr** startRoutine, void *arg)
*Creates a new thread.*

PUBLIC
**IX_STATUS** **ixOsalThreadStart** (**IxOsalThread** *thread)
*Starts a newly created thread.*

PUBLIC
**IX_STATUS** **ixOsalThreadKill** (**IxOsalThread** *thread)
*Kills an existing thread.*

PUBLIC void **ixOsalThreadExit** (void)
*Exits a running thread.*

PUBLIC
**IX_STATUS** **ixOsalThreadPrioritySet** (**IxOsalThread** *thread, UINT32 priority)
*Sets the priority of an existing thread.*

PUBLIC
**IX_STATUS** **ixOsalThreadSuspend** (**IxOsalThread** *thread)
*Suspends thread execution.*

PUBLIC
**IX_STATUS** **ixOsalThreadResume** (**IxOsalThread** *thread)
*Resumes thread execution.*

PUBLIC BOOL **ixOsalThreadStopCheck** (void)
*Check if thread should stop execution.*

PUBLIC **ixOsalMessageQueueCreate** (**IxOsalMessageQueue** *queue, UINT32 msgCount,
**IX_STATUS** UINT32 msgLen)
*Creates a message queue.*

PUBLIC
**IX_STATUS** **ixOsalMessageQueueDelete** (**IxOsalMessageQueue** *queue)
*Deletes a message queue.*

PUBLIC
**IX_STATUS** **ixOsalMessageQueueSend** (**IxOsalMessageQueue** *queue, UINT8 *message)
*Sends a message to a message queue.*

PUBLIC
**IX_STATUS** **ixOsalMessageQueueReceive** (**IxOsalMessageQueue** *queue, UINT8 *message)
*Receives a message from a message queue.*

PUBLIC
**IX_STATUS** **ixOsalMutexInit** (**IxOsalMutex** *mutex)
*initializes a mutex*

**ixOsalMutexLock** (**IxOsalMutex** *mutex, INT32 timeout)

PUBLIC
**IX_STATUS**
*locks a mutex*

PUBLIC
**IX_STATUS ixOsalMutexUnlock** (**IxOsalMutex** *mutex)
*Unlocks a mutex.*

PUBLIC
**IX_STATUS ixOsalMutexTryLock** (**IxOsalMutex** *mutex)
*Non−blocking attempt to lock a mutex.*

PUBLIC
**IX_STATUS ixOsalMutexDestroy** (**IxOsalMutex** *mutex)
*Destroys a mutex object.*

PUBLIC
**IX_STATUS ixOsalFastMutexInit** (**IxOsalFastMutex** *mutex)
*Initializes a fast mutex.*

PUBLIC
**IX_STATUS ixOsalFastMutexTryLock** (**IxOsalFastMutex** *mutex)
*Non−blocking attempt to lock a fast mutex.*

PUBLIC
**IX_STATUS ixOsalFastMutexUnlock** (**IxOsalFastMutex** *mutex)
*Unlocks a fast mutex.*

PUBLIC
**IX_STATUS ixOsalFastMutexDestroy** (**IxOsalFastMutex** *mutex)
*Destroys a fast mutex object.*

PUBLIC
**IX_STATUS ixOsalSemaphoreInit** (**IxOsalSemaphore** *semaphore, UINT32 value)
*Initializes a semaphore.*

PUBLIC
**IX_STATUS ixOsalSemaphorePost** (**IxOsalSemaphore** *semaphore)
*Posts to (increments) a semaphore.*

PUBLIC
**IX_STATUS ixOsalSemaphoreWait** (**IxOsalSemaphore** *semaphore, INT32 timeout)
*Waits on (decrements) a semaphore.*

PUBLIC
**IX_STATUS ixOsalSemaphoreTryWait** (**IxOsalSemaphore** *semaphore)
*Non−blocking wait on semaphore.*

PUBLIC
**IX_STATUS ixOsalSemaphoreGetValue** (**IxOsalSemaphore** *semaphore, UINT32 *value)

*Gets semaphore value.*

PUBLIC
**IX_STATUS ixOsalSemaphoreDestroy** (**IxOsalSemaphore** *semaphore)
*Destroys a semaphore object.*

PUBLIC void **ixOsalYield** (void)
*Yields execution of current thread.*

PUBLIC void **ixOsalSleep** (UINT32 milliseconds)
*Yielding sleep for a number of milliseconds.*

PUBLIC void **ixOsalBusySleep** (UINT32 microseconds)
*Busy sleep for a number of microseconds.*

PUBLIC UINT32 **ixOsalTimestampGet** (void)
*Retrieves the current timestamp.*

PUBLIC UINT32 **ixOsalTimestampResolutionGet** (void)
*Resolution of the timestamp counter.*

PUBLIC UINT32 **ixOsalSysClockRateGet** (void)
*System clock rate, in ticks.*

PUBLIC void **ixOsalTimeGet** (**IxOsalTimeval** *tv)
*Current system time.*

PUBLIC INT32 **ixOsalLog** (**IxOsalLogLevel** level, **IxOsalLogDevice** device, char *format, int arg1, int arg2, int arg3, int arg4, int arg5, int arg6)
*Interrupt−safe logging function.*

PUBLIC UINT32 **ixOsalLogLevelSet** (UINT32 level)
*sets the current logging verbosity level*

PUBLIC **IX_STATUS ixOsalRepeatingTimerSchedule** (**IxOsalTimer** *timer, UINT32 period, UINT32 priority, **IxOsalVoidFnVoidPtr** callback, void *param)
*Schedules a repeating timer.*

PUBLIC **IX_STATUS ixOsalSingleShotTimerSchedule** (**IxOsalTimer** *timer, UINT32 period, UINT32 priority, **IxOsalVoidFnVoidPtr** callback, void *param)
*Schedules a single−shot timer.*

PUBLIC
**IX_STATUS ixOsalTimerCancel** (**IxOsalTimer** *timer)
*Cancels a running timer.*

PUBLIC void **ixOsalTimersShow** (void)
*displays all the running timers*

PUBLIC
**IX_STATUS ixOsalOsNameGet** (INT8 *osName, INT32 maxSize)

*provides the name of the Operating System running*

PUBLIC
**IX_STATUS ixOsalOsVersionGet** (INT8 *osVersion, INT32 maxSize)
*provides the version of the Operating System running*

---

# Detailed Description

This service provides a thin layer of OS dependency services.

This file contains the API to the functions which are some what OS dependant and would require porting to a particular OS. A primary focus of the component development is to make them as OS independent as possible. All other components should abstract their OS dependency to this module. Services overview

1. Data types, constants, defines
2. Interrupts
    - ♦ bind interrupts to handlers
    - ♦ unbind interrupts from handlers
    - ♦ disables all interrupts
    - ♦ enables all interrupts
    - ♦ selectively disables interrupts
    - ♦ enables an interrupt level
    - ♦ disables an interrupt level

Memory

- allocates memory
- frees memory
- copies memory zones
- fills a memory zone
- allocates cache−safe memory
- frees cache−safe memory
- physical to virtual address translation
- virtual to physical address translation
- cache to memory flush
- cache line invalidate

Threads

- creates a new thread
- starts a newly created thread
- kills an existing thread
- exits a running thread
- sets the priority of an existing thread
- suspends thread execution
- resumes thread execution

IPC

- creates a message queue
- deletes a message queue
- sends a message to a message queue
- receives a message from a message queue

Thread Synchronisation

- initializes a mutex
- locks a mutex
- unlocks a mutex
- non−blocking attempt to lock a mutex
- destroys a mutex object
- initializes a fast mutex
- non−blocking attempt to lock a fast mutex
- unlocks a fast mutex
- destroys a fast mutex object
- initializes a semaphore
- posts to (increments) a semaphore
- waits on (decrements) a semaphore
- non−blocking wait on semaphore
- gets semaphore value
- destroys a semaphore object
- yields execution of current thread

Time functions

- yielding sleep for a number of milliseconds
- busy sleep for a number of microseconds
- value of the timestamp counter
- resolution of the timestamp counter
- system clock rate, in ticks
- current system time
- converts ixOsalTimeVal into ticks
- converts ticks into ixOsalTimeVal
- converts ixOsalTimeVal to milliseconds
- converts milliseconds to **IxOsalTimeval**
- "equal" comparison for **IxOsalTimeval**
- "less than" comparison for **IxOsalTimeval**
- "greater than" comparison for **IxOsalTimeval**
- "add" operator for **IxOsalTimeval**
- "subtract" operator for **IxOsalTimeval**

Logging

- sets the current logging verbosity level
- interrupt−safe logging function

Timer services

- schedules a repeating timer
- schedules a single−shot timer

- cancels a running timer
- displays all the running timers

PCI Support

- Find PCI device.
- Read 8 bits from the configuration space
- Read 16 bits from the configuration space
- Read 32 bits from the configuration space
- Write 8 bits to the configuration space
- Write 16 bits to the configuration space
- Write 32 bits to the configuration space
- Free PCI device

SpinLock Support

- Initializes the SpinLock object
- Acquires a spin lock
- Releases the spin lock
- Tries to acquire the spin lock
- Destroy the spin lock object

Optional Modules

- Device drivers kernel
- Buffer management module
- I/O memory and endianess support module

---

# Define Documentation

```
#define IX_OSAL_CACHE_FLUSH ( addr,
                              size   )
```

cache to memory flush

***Parameters:***

*addr* − memory address to flush from cache

*size* − number of bytes to flush (rounded up to a cache line)

Flushes the cached value of the memory zone pointed by "addr" into memory, rounding up to a cache line. Use before the zone is to be read by a processing unit which is not cache coherent with the main CPU.

- Reentrant: no
- IRQ safe: yes

***Returns:***

− none

Definition at line **457** of file **IxOsal.h**.

```
#define IX_OSAL_CACHE_INVALIDATE ( addr,
                                   size )
```

cache line invalidate

***Parameters:***

*addr* − memory address to invalidate in cache
*size* − number of bytes to invalidate (rounded up to a cache line)

Invalidates the cached value of the memory zone pointed by "addr", rounding up to a cache line. Use before reading the zone from the main CPU, if the zone has been updated by a processing unit which is not cache coherent with the main CPU.

- Reentrant: no
- IRQ safe: yes

***Returns:***

− none

Definition at line **479** of file **IxOsal.h**.

```
#define IX_OSAL_CACHE_PRELOAD ( addr,
                                size )
```

cache line preload

***Parameters:***

*addr* − memory address to cache
*size* − number of bytes to cache (rounded up to a cache line)

Preloads a section of memory to the cache memory in multiples of cache line size.

- Reentrant: no
- IRQ safe: yes

***Returns:***

− none

Definition at line **497** of file **IxOsal.h**.

```
#define IX_OSAL_MMU_PHYS_TO_VIRT ( physAddr )
```

physical to virtual address translation

***Parameters:***

*physAddr* − physical address

Converts a physical address into its equivalent MMU−mapped virtual address

- Reentrant: no
- IRQ safe: yes

***Returns:***
Corresponding virtual address, as UINT32

Definition at line **372** of file **IxOsal.h**.

```
#define IX_OSAL_MMU_VIRT_TO_PHYS ( virtAddr )
```

virtual to physical address translation

***Parameters:***
*virtAddr* − virtual address

Converts a virtual address into its equivalent MMU−mapped physical address

- Reentrant: no
- IRQ safe: yes

***Returns:***
Corresponding physical address, as UINT32

Definition at line **389** of file **IxOsal.h**.

```
#define IX_OSAL_MS_TO_TIMEVAL ( milliseconds,
                                pTv          )
```

Converts milliseconds to **IxOsalTimeval**.

***Parameters:***
*milliseconds* − number of milliseconds to convert
*pTv* − pointer to the destination structure

Converts a millisecond value into an **IxOsalTimeval** structure

- Reentrant: yes
- IRQ safe: yes

***Returns:***
− Corresponding **IxOsalTimeval** structure Note: This function is OS−independent. Implemented by core.

Definition at line **1296** of file **IxOsal.h**.

```
#define IX_OSAL_TICKS_TO_TIMEVAL ( ticks,
                                   pTv   )
```

Converts ticks into ixOsalTimeVal.

***Parameters:***

*ticks* − number of ticks

*pTv* − pointer to the destination structure

Converts the specified number of ticks into an **IxOsalTimeval** structure

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− Corresponding **IxOsalTimeval** structure Note: This function is OS−independent. Implemented by core.

Definition at line **1256** of file **IxOsal.h**.

```
#define IX_OSAL_TIME_ADD ( tvA,
                           tvB  )
```

"add" operator for **IxOsalTimeval**

***Parameters:***

*tvA,tvB* − **IxOsalTimeval** structures to add

Adds the second IxOsalTimevalStruct to the first one (equivalent to tvA += tvB)

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− none Note: This function is OS−independent.

Definition at line **1381** of file **IxOsal.h**.

```
#define IX_OSAL_TIME_EQ ( tvA,
                          tvB  )
```

"equal" comparison for **IxOsalTimeval**

***Parameters:***

*tvA,tvB* − **IxOsalTimeval** structures to compare

Compares two **IxOsalTimeval** structures for equality

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− TRUE if the structures are equal
◊ FALSE otherwise Note: This function is OS−independant

Definition at line **1317** of file **IxOsal.h**.

```
#define IX_OSAL_TIME_GT ( tvA,
                          tvB  )
```

"greater than" comparison for **IxOsalTimeval**

***Parameters:***

*tvA,tvB* − **IxOsalTimeval** structures to compare

Compares two **IxOsalTimeval** structures to determine if the first one is greater than the second one

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− TRUE if tvA > tvB
◊ FALSE otherwise Note: This function is OS−independent.

Definition at line **1360** of file **IxOsal.h**.

```
#define IX_OSAL_TIME_LT ( tvA,
                          tvB  )
```

"less than" comparison for **IxOsalTimeval**

***Parameters:***

*tvA,tvB* − **IxOsalTimeval** structures to compare

Compares two **IxOsalTimeval** structures to determine if the first one is less than the second one

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− TRUE if tvA < tvB
◊ FALSE otherwise Note: This function is OS−independent. Implemented by core.

Definition at line **1338** of file **IxOsal.h**.

```
#define IX_OSAL_TIME_SUB ( tvA,
                           tvB )
```

"subtract" operator for **IxOsalTimeval**

***Parameters:***

*tvA,tvB* − **IxOsalTimeval** structures to subtract

Subtracts the second IxOsalTimevalStruct from the first one (equivalent to tvA −= tvB)

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− none Note: This function is OS−independent. Implemented by core.

Definition at line **1406** of file **IxOsal.h**.

```
#define IX_OSAL_TIMEVAL_TO_MS ( tv )
```

Converts ixOsalTimeVal to milliseconds.

***Parameters:***

*tv* − **IxOsalTimeval** structure to convert

Converts an **IxOsalTimeval** structure into milliseconds

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− Corresponding number of milliseconds Note: This function is OS−independent. Implemented by core.

Definition at line **1277** of file **IxOsal.h**.

```
#define IX_OSAL_TIMEVAL_TO_TICKS ( tv )
```

Converts ixOsalTimeVal into ticks.

***Parameters:***

*tv* − an **IxOsalTimeval** structure

Converts an **IxOsalTimeval** structure into OS ticks

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− Corresponding number of ticks

Note: This function is OS−independent. Implemented by core.

Definition at line **1228** of file **IxOsal.h**.

```
#define IX_OSAL_UDIV64_32 ( dividend,
                            divisor     )
```

UINT64 data type division with 32 bit divisor.

***Parameters:***

*dividend* − dividend (UINT64)
*divisor* − divisor (UINT32)

This function enable UINT64 datatype division for 32 bit system.

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− quotient (UINT64)

Definition at line **1819** of file **IxOsal.h**.

```
#define IX_OSAL_UMOD64_32 ( dividend,
                            divisor     )
```

UINT64 data type mod with 32 bit divisor.

***Parameters:***

*dividend* − dividend (UINT64)
*divisor* − divisor (UINT32)

This function enable UINT64 datatype mod for 32 bit system.

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− remainder (UINT32)

Definition at line **1838** of file **IxOsal.h**.

---

# Function Documentation

PUBLIC void ixOsalBusySleep ( UINT32 *microseconds* )

Busy sleep for a number of microseconds.

***Parameters:***
*microseconds* − number of microseconds to sleep

Sleeps for the specified number of microseconds, without explicitly yielding thread execution to the OS scheduler

- Reentrant: yes
- IRQ safe: yes

***Returns:***
− none

PUBLIC void ixOsalCacheDmaFree ( void * *ptr* )

Frees cache−safe memory.

***Parameters:***
*ptr* − pointer to the memory zone

Frees a memory zone previously allocated with **ixOsalCacheDmaMalloc()**

- Reentrant: no
- IRQ safe: no

***Returns:***
− none

PUBLIC void* ixOsalCacheDmaMalloc ( UINT32 *size* )

Allocates cache−safe memory.

***Parameters:***
*size* − size, in bytes, of the allocated zone

Allocates a cache−safe memory zone of at least "size" bytes and returns the pointer to the memory zone. This memory zone, depending on the platform, is either uncached or aligned on a cache line boundary to make the CACHE_FLUSH and CACHE_INVALIDATE macros safe to use. The memory allocated with this function MUST be freed with **ixOsalCacheDmaFree()**, otherwise memory corruption can occur.

- Reentrant: no

- IRQ safe: no

***Returns:***

Pointer to the memory zone or NULL if allocation failed

***Note:***

It is important to note that cache coherence is maintained in software by using the IX_OSAL_CACHE_FLUSH and IX_OSAL_CACHE_INVALIDATE macros to maintain consistency between cache and external memory.

PUBLIC **IX_STATUS** ixOsalFastMutexDestroy ( **IxOsalFastMutex** * *mutex* )

Destroys a fast mutex object.

***Parameters:***

*mutex* − fast mutex handle

Destroys a fast mutex object

- Reentrant: yes
- IRQ safe: yes

***Returns:***

–

IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalFastMutexInit ( **IxOsalFastMutex** * *mutex* )

Initializes a fast mutex.

***Parameters:***

*mutex* − fast mutex handle

Initializes a fast mutex object

- Reentrant: yes
- IRQ safe: yes

***Returns:***

–

IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalFastMutexTryLock ( **IxOsalFastMutex** * *mutex* )

Non−blocking attempt to lock a fast mutex.

***Parameters:***

*mutex* − fast mutex handle

Attempts to lock a fast mutex object, returning immediately with IX_SUCCESS if the lock was successful or IX_FAIL if the lock failed

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalFastMutexUnlock ( **IxOsalFastMutex** * *mutex* )

Unlocks a fast mutex.

***Parameters:***

*mutex* − fast mutex handle

Unlocks a fast mutex object

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− IX_SUCCESS/IX_FAIL

```
PUBLIC IX_STATUS ixOsalIrqBind ( UINT32                 irqLevel,
                                 IxOsalVoidFnVoidPtr    irqHandler,
                                 void *                 parameter
                               )
```

Binds an interrupt handler to an interrupt level.

***Parameters:***

*irqLevel* (in) − interrupt level
*irqHandler* (in) − interrupt handler
*parameter* (in) − custom parameter to be passed to the interrupt handler

Binds an interrupt handler to an interrupt level. The operation will fail if the wrong level is selected, if the handler is NULL, or if the interrupt is already bound. This functions binds the specified C routine to an interrupt level. When called, the "parameter" value will be passed to the routine.

Reentrant: no IRQ safe: no

***Returns:***

IX_SUCCESS if the operation succeeded or IX_FAIL otherwise

PUBLIC void ixOsalIrqDisable ( UINT32 *irqLevel* )

Disables an interrupt level.

***Parameters:***

*irqLevel* - interrupt level to disable

Disables the specified interrupt level

- Reentrant: no
- IRQ safe: yes

***Returns:***

− none

PUBLIC void ixOsalIrqEnable ( UINT32 *irqLevel* )

Enables an interrupt level.

***Parameters:***

*irqLevel* - interrupt level to enable

Enables the specified interrupt level

- Reentrant: no
- IRQ safe: yes

***Returns:***

− none

PUBLIC UINT32 ixOsalIrqLevelSet ( UINT32 *irqLevel* )

Selectively disables interrupts.

***Parameters:***

*irqLevel* - new interrupt level

Disables the interrupts below the specified interrupt level

- Reentrant: no
- IRQ safe: yes

***Note:***

Depending on the implementation this function can disable all the interrupts

***Returns:***
previous interrupt level

PUBLIC UINT32 ixOsalIrqLock ( void )

Disables all interrupts.

***Parameters:***
- none

Disables all the interrupts and prevents tasks scheduling

- Reentrant: no
- IRQ safe: yes

***Returns:***
interrupt enable status prior to locking

PUBLIC **IX_STATUS** ixOsalIrqUnbind ( UINT32 *irqLevel* )

Unbinds an interrupt handler from an interrupt level.

***Parameters:***
*irqLevel* (in) − interrupt level

Unbinds the selected interrupt level from any previously registered handler

- Reentrant: no
- IRQ safe: no

***Returns:***
IX_SUCCESS if the operation succeeded or IX_FAIL otherwise

PUBLIC void ixOsalIrqUnlock ( UINT32 *irqEnable* )

Enables all interrupts.

***Parameters:***
*irqEnable* (in) − interrupt enable status, prior to interrupt locking

Enables the interrupts and task scheduling, cancelling the effect of **ixOsalIrqLock()**

- Reentrant: no
- IRQ safe: yes

***Returns:***

IX_SUCCESS if the operation succeeded or IX_FAIL otherwise

```
PUBLIC INT32 ixOsalLog ( IxOsalLogLevel   level,
                         IxOsalLogDevice  device,
                         char *           format,
                         int              arg1,
                         int              arg2,
                         int              arg3,
                         int              arg4,
                         int              arg5,
                         int              arg6
                       )
```

Interrupt−safe logging function.

***Parameters:***

*level* − identifier prefix for the message
*device* − output device
*format* − message format, in a printf format
... − up to 6 arguments to be printed

IRQ−safe logging function, similar to printf. Accepts up to 6 arguments to print (excluding the level, device and the format). This function will actually display the message only if the level is lower than the current verbosity level or if the IX_OSAL_LOG_USER level is used. An output device must be specified (see **IxOsalTypes.h**).

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− Beside the exceptions documented in the note below, the returned value is the number of printed characters, or −1 if the parameters are incorrect (NULL format, unknown output device)

***Note:***

The exceptions to the return value are: VxWorks*: The return value is 32 if the specified level is 1 and 64 if the specified level is greater than 1 and less or equal than 9. WinCE*: If compiled for EBOOT then the return value is always 0.

The given print format should take into account the specified output device. IX_OSAL_STDOUT supports all the usual print formats, however a custom hex display specified by IX_OSAL_HEX would support only a fixed number of hexadecimal digits.

```
PUBLIC UINT32 ixOsalLogLevelSet ( UINT32 level )
```

sets the current logging verbosity level

***Parameters:***

*level* − new log verbosity level

Sets the log verbosity level. The default value is IX_OSAL_LOG_ERROR.

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− Old log verbosity level

```
PUBLIC void* ixOsalMemAlloc ( UINT32 size )
```

Allocates memory.

***Parameters:***

*size* − memory size to allocate, in bytes

Allocates a memory zone of a given size

- Reentrant: no
- IRQ safe: no

***Returns:***

Pointer to the allocated zone or NULL if the allocation failed

```
PUBLIC void* ixOsalMemCopy ( void *    dest,
                             void *    src,
                             UINT32 count
                           )
```

Copies memory zones.

***Parameters:***

*dest* − destination memory zone

*src* − source memory zone

*count* − number of bytes to copy

Copies count bytes from the source memory zone pointed by src into the memory zone pointed by dest.

- Reentrant: no
- IRQ safe: yes

***Returns:***

Pointer to the destination memory zone

```
PUBLIC void ixOsalMemFree ( void * ptr )
```

Frees memory.

***Parameters:***

*ptr* − pointer to the memory zone

Frees a previously allocated memory zone

- Reentrant: no
- IRQ safe: no

***Returns:***

− none

```
PUBLIC void* ixOsalMemSet ( void *    ptr,
                            UINT8   filler,
                            UINT32  count
                          )
```

Fills a memory zone.

***Parameters:***

*ptr* − pointer to the memory zone
*filler* − byte to fill the memory zone with
*count* − number of bytes to fill

Fills a memory zone with a given constant byte

- Reentrant: no
- IRQ safe: yes

***Returns:***

Pointer to the memory zone

```
PUBLIC IX_STATUS ixOsalMessageQueueCreate ( IxOsalMessageQueue *  queue,
                                            UINT32                msgCount,
                                            UINT32                msgLen
                                          )
```

Creates a message queue.

***Parameters:***

*queue* − queue handle
*msgCount* − maximum number of messages to hold in the queue
*msgLen* − maximum length of each message, in bytes

Creates a message queue of msgCount messages, each containing msgLen bytes

- Reentrant: no
- IRQ safe: no

***Returns:***
− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMessageQueueDelete ( **IxOsalMessageQueue** * *queue* )

Deletes a message queue.

***Parameters:***
*queue* − queue handle

Deletes a message queue

- Reentrant: no
- IRQ safe: no

***Returns:***
−
IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMessageQueueReceive ( **IxOsalMessageQueue** * *queue*,
UINT8 * *message*
)

Receives a message from a message queue.

***Parameters:***
*queue* − queue handle
*message* − pointer to where the message should be copied to

Retrieves the first message from the message queue

- Reentrant: yes
- IRQ safe: yes

***Returns:***
− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMessageQueueSend ( **IxOsalMessageQueue** * *queue*,
UINT8 * *message*
)

Sends a message to a message queue.

***Parameters:***

*queue* − queue handle
*message* − message to send

Sends a message to the message queue. The message will be copied (at the configured size of the message) into the queue.

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMutexDestroy ( **IxOsalMutex** * *mutex* )

Destroys a mutex object.

***Parameters:***

*mutex* − mutex handle
Destroys a mutex object; the caller should ensure that no thread is blocked on this mutex

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMutexInit ( **IxOsalMutex** * *mutex* )

initializes a mutex

***Parameters:***

*mutex* − mutex handle

Initializes a mutex object

- Reentrant: no
- IRQ safe: no

***Returns:***

−
IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMutexLock ( **IxOsalMutex** * *mutex*,
INT32 *timeout*
)

locks a mutex

***Parameters:***

*mutex* − mutex handle

*timeout* − timeout in ms; IX_OSAL_WAIT_FOREVER (−1) to wait forever or IX_OSAL_WAIT_NONE to return immediately

Locks a mutex object

- Reentrant: yes
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMutexTryLock ( **IxOsalMutex** * *mutex* )

Non−blocking attempt to lock a mutex.

***Parameters:***

*mutex* − mutex handle

Attempts to lock a mutex object, returning immediately with IX_SUCCESS if the lock was successful or IX_FAIL if the lock failed

- Reentrant: yes
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalMutexUnlock ( **IxOsalMutex** * *mutex* )

Unlocks a mutex.

***Parameters:***

*mutex* − mutex handle

Unlocks a mutex object

- Reentrant: yes
- IRQ safe: yes

***Returns:***

–

IX_SUCCESS/IX_FAIL

```
PUBLIC IX_STATUS ixOsalOsNameGet  ( INT8 *  osName,
                                    INT32   maxSize
                                  )
```

provides the name of the Operating System running

***Parameters:***

*osName* − Pointer to a NULL−terminated string of characters that holds the name of the OS running. This is both an input and an ouput parameter

*maxSize* − Input parameter that defines the maximum number of bytes that can be stored in osName

Returns a string of characters that describe the Operating System name

- Reentrant: yes
- IRQ safe: yes

return − IX_SUCCESS for successful retrieval

- IX_FAIL if (osType == NULL | maxSize =< 0)

```
PUBLIC IX_STATUS ixOsalOsVersionGet  ( INT8 *  osVersion,
                                       INT32   maxSize
                                     )
```

provides the version of the Operating System running

***Parameters:***

*osVersion* − Pointer to a NULL terminated string of characters that holds the version of the OS running. This is both an input and an ouput parameter

*maxSize* − Input parameter that defines the maximum number of bytes that can be stored in osVersion

Returns a string of characters that describe the Operating System's version

- Reentrant: yes
- IRQ safe: yes

return − IX_SUCCESS for successful retrieval

- IX_FAIL if (osVersion == NULL | maxSize =< 0)

PUBLIC **IX_STATUS** ixOsalRepeatingTimerSchedule ( **IxOsalTimer** * *timer*,
UINT32 *period*,
UINT32 *priority*,
**IxOsalVoidFnVoidPtr** *callback*,
void * *param*
)

Schedules a repeating timer.

***Parameters:***

*timer* − handle of the timer object
*period* − timer trigger period, in milliseconds
*priority* − timer priority (0 being the highest)
*callback* − user callback to invoke when the timer triggers
*param* − custom parameter passed to the callback

Schedules a timer to be called every period milliseconds. The timer will invoke the specified callback function possibly in interrupt context, passing the given parameter. If several timers trigger at the same time contention issues are dealt according to the specified timer priorities.

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSemaphoreDestroy ( **IxOsalSemaphore** * *semaphore* )

Destroys a semaphore object.

***Parameters:***

*semaphore* − semaphore handle

Destroys a semaphore object; the caller should ensure that no thread is blocked on this semaphore

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSemaphoreGetValue ( **IxOsalSemaphore** * *semaphore*,
UINT32 * *value*
)

Gets semaphore value.

***Parameters:***

*semaphore* − semaphore handle
*value* − location to store the semaphore value

Retrieves the current value of a semaphore object

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSemaphoreInit ( **IxOsalSemaphore** * *semaphore*,
UINT32 *value*
)

Initializes a semaphore.

***Parameters:***

*semaphore* − semaphore handle
*value* − initial semaphore value

Initializes a semaphore object

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSemaphorePost ( **IxOsalSemaphore** * *semaphore* )

Posts to (increments) a semaphore.

***Parameters:***

*semaphore* − semaphore handle

Increments a semaphore object

- Reentrant: no
- IRQ safe: yes

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSemaphoreTryWait ( **IxOsalSemaphore** * *semaphore* )

Non−blocking wait on semaphore.

***Parameters:***

*semaphore* − semaphore handle

Decrements a semaphore, not blocking the calling thread if the semaphore is unavailable

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSemaphoreWait ( **IxOsalSemaphore** * *semaphore*,
INT32 *timeout*
)

Waits on (decrements) a semaphore.

***Parameters:***

*semaphore* − semaphore handle

*timeout* − timeout, in ms; IX_OSAL_WAIT_FOREVER (−1) if the thread is to block indefinitely or IX_OSAL_WAIT_NONE (0) if the thread is to return immediately even if the call fails

Decrements a semaphore, blocking if the semaphore is unavailable (value is 0).

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalSingleShotTimerSchedule ( **IxOsalTimer** * *timer*,
UINT32 *period*,
UINT32 *priority*,
**IxOsalVoidFnVoidPtr** *callback*,
void * *param*
)

Schedules a single−shot timer.

***Parameters:***

*timer* − handle of the timer object
*period* − timer trigger period, in milliseconds
*priority* − timer priority (0 being the highest)
*callback* − user callback to invoke when the timer triggers
*param* − custom parameter passed to the callback

Schedules a timer to be called after period milliseconds. The timer will cease to function past its first trigger. The timer will invoke the specified callback function, possibly in interrupt context, passing the given parameter. If several timers trigger at the same time contention issues are dealt according to the specified timer priorities.

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC void ixOsalSleep ( UINT32 *milliseconds* )

Yielding sleep for a number of milliseconds.

***Parameters:***

*milliseconds* − number of milliseconds to sleep

The calling thread will sleep for the specified number of milliseconds. This sleep is yielding, hence other tasks will be scheduled by the operating system during the sleep period. Calling this function with an argument of 0 will place the thread at the end of the current scheduling loop.

- Reentrant: no
- IRQ safe: no

***Returns:***

− none

PUBLIC UINT32 ixOsalSysClockRateGet ( void )

System clock rate, in ticks.

Retrieves the resolution (number of ticks per second) of the system clock

- Reentrant: no
- IRQ safe: no

***Returns:***

− The system clock rate

***Note:***

The implementation of this function is platform and OS−specific. The system clock rate is not always available −

```
PUBLIC IX_STATUS ixOsalThreadCreate ( IxOsalThread *        thread,
                                      IxOsalThreadAttr *    threadAttr,
                                      IxOsalVoidFnVoidPtr   startRoutine,
                                      void *                arg
                                    )
```

Creates a new thread.

***Parameters:***

| | |
|---|---|
| *thread* | − handle of the thread to be created |
| *threadAttr* | − pointer to a thread attribute object |
| *startRoutine* | − thread entry point |
| *arg* | − argument given to the thread |

Creates a thread given a thread handle and a thread attribute object. The same thread attribute object can be used to create separate threads. "NULL" can be specified as the attribute, in which case the default values will be used. The thread needs to be explicitly started using **ixOsalThreadStart()**.

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

```
PUBLIC void ixOsalThreadExit ( void )
```

Exits a running thread.

Terminates the calling thread

- Reentrant: no
- IRQ safe: no

***Returns:***

− This function never returns

```
PUBLIC IX_STATUS ixOsalThreadKill ( IxOsalThread * thread )
```

Kills an existing thread.

***Parameters:***

*thread* − handle of the thread to be killed

Kills a thread given its thread handle.

- Reentrant: no
- IRQ safe: no

***Note:***

This function does not guarentee to kill the thread immediately. The thread must use **ixOsalThreadStopCheck()** to check if it should perform cleanup and suicide.

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalThreadPrioritySet ( **IxOsalThread** * *thread*,
UINT32 *priority*
)

Sets the priority of an existing thread.

***Parameters:***

*thread* − handle of the thread
*priority* − new priority, between 0 and 255 (0 being the highest)

Sets the thread priority

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalThreadResume ( **IxOsalThread** * *thread* )

Resumes thread execution.

***Parameters:***

*thread* − handle of the thread

Resumes the thread execution

- Reentrant: no
- IRQ safe: no

***Returns:***

–
IX_SUCCESS/IX_FAIL

PUBLIC **IX_STATUS** ixOsalThreadStart ( **IxOsalThread** * *thread* )

Starts a newly created thread.

***Parameters:***

*thread* − handle of the thread to be started

Starts a thread given its thread handle. This function is to be called only once, following the thread initialization.

- Reentrant: no
- IRQ safe: no

***Returns:***

− IX_SUCCESS/IX_FAIL

PUBLIC BOOL ixOsalThreadStopCheck ( void )

Check if thread should stop execution.

Check if ixOsalThreadKill has been called. When this API return TRUE, the thread should perform cleanup and exit.

- Reentrant: no
- IRQ safe: no

***Returns:***

− TRUE/FALSE

PUBLIC **IX_STATUS** ixOsalThreadSuspend ( **IxOsalThread** * *thread* )

Suspends thread execution.

***Parameters:***

*thread* − handle of the thread

Suspends the thread execution

- Reentrant: no
- IRQ safe: no

***Returns:***

–
IX_SUCCESS/IX_FAIL

PUBLIC void ixOsalTimeGet ( **IxOsalTimeval** * *tv* )

Current system time.

***Parameters:***
*tv* − pointer to an **IxOsalTimeval** structure to store the current time in

Retrieves the current system time (real−time)

- Reentrant: no
- IRQ safe: no

***Returns:***
− none

***Note:***
The implementation of this function is platform−specific. Not all platforms have a real−time clock.

PUBLIC **IX_STATUS** ixOsalTimerCancel ( **IxOsalTimer** * *timer* )

Cancels a running timer.

***Parameters:***
*timer* − handle of the timer object

Cancels a single−shot or repeating timer.

- Reentrant: no
- IRQ safe: yes

***Returns:***
− IX_SUCCESS/IX_FAIL

PUBLIC void ixOsalTimersShow ( void )

displays all the running timers

Displays a list with all the running timers and their parameters (handle, period, type, priority, callback and user parameter)

- Reentrant: no
- IRQ safe: no

***Returns:***

− none

PUBLIC UINT32 ixOsalTimestampGet ( void )

Retrieves the current timestamp.

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− The current timestamp

***Note:***

The implementation of this function is platform−specific. Not all the platforms provide a high−resolution timestamp counter.

PUBLIC UINT32 ixOsalTimestampResolutionGet ( void )

Resolution of the timestamp counter.

Retrieves the resolution (frequency) of the timestamp counter.

- Reentrant: yes
- IRQ safe: yes

***Returns:***

− The resolution of the timestamp counter

***Note:***

The implementation of this function is platform−specific. Not all the platforms provide a high−resolution timestamp counter.

PUBLIC void ixOsalYield ( void )

Yields execution of current thread.

Yields the execution of the current thread

- Reentrant: no
- IRQ safe: no

***Returns:***

− none

# OSAL Buffer Management Module.

Buffer management module for IxOsal.

## Defines

#define **IX_OSAL_MBUF_MAX_POOLS**
*The maximum number of pools that can be allocated, must be a multiple of 32 as required by implementation logic.*

#define **IX_OSAL_MBUF_POOL_NAME_LEN**
*The maximum string length of the pool name.*

#define **IX_OSAL_MBUF_NEXT_PKT_IN_CHAIN_PTR**(m_blk_ptr)
*Return pointer to the next packet in the chain.*

#define **IX_OSAL_MBUF_MDATA**(m_blk_ptr)
*Return pointer to the data in the mbuf.*

#define **IX_OSAL_MBUF_MLEN**(m_blk_ptr)
*Return the data length.*

#define **IX_OSAL_MBUF_MTYPE**(m_blk_ptr)
*Return the data type in the mbuf.*

#define **IX_OSAL_MBUF_FLAGS**(m_blk_ptr)
*Return the buffer flags.*

#define **IX_OSAL_MBUF_NET_POOL**(m_blk_ptr)
*Return pointer to a network pool.*

#define **IX_OSAL_MBUF_PKT_LEN**(m_blk_ptr)
*Return the total length of all the data in the mbuf chain for this packet.*

#define **IX_OSAL_MBUF_PRIV**(m_blk_ptr)
*Return the private field.*

#define **IX_OSAL_MBUF_SIGNATURE**(m_blk_ptr)
*Return the signature field of IX_OSAL_MBUF.*

#define **IX_OSAL_MBUF_OSBUF_PTR**(m_blk_ptr)
*Return ix_osbuf_ptr field of IX_OSAL_MBUF, which is used to store OS−specific buffer pointer during a buffer conversion.*

#define **IX_OSAL_MBUF_ALLOCATED_BUFF_LEN**(m_blk_ptr)
*Return the allocated buffer size.*

#define **IX_OSAL_MBUF_ALLOCATED_BUFF_DATA**(m_blk_ptr)

*Return the allocated buffer pointer.*

#define **IX_OSAL_MBUF_POOL_SIZE_ALIGN**(size)
*This macro takes an integer as an argument and rounds it up to be a multiple of the memory cache−line size.*

#define **IX_OSAL_MBUF_POOL_MBUF_AREA_SIZE_ALIGNED**(count)
*This macro calculates, from the number of mbufs required, the size of the memory area required to contain the mbuf headers for the buffers in the pool. The size to be used for each mbuf header is rounded up to a multiple of the cache−line size, to ensure each mbuf header aligns on a cache−line boundary. This macro is used by **IX_OSAL_MBUF_POOL_MBUF_AREA_ALLOC()**.*

#define **IX_OSAL_MBUF_POOL_DATA_AREA_SIZE_ALIGNED**(count, size)
*This macro calculates, from the number of mbufs required and the size of the data portion for each mbuf, the size of the data memory area required. The size is adjusted to ensure alignment on cache line boundaries. This macro is used by **IX_OSAL_MBUF_POOL_DATA_AREA_ALLOC()**.*

#define **IX_OSAL_MBUF_POOL_MBUF_AREA_ALLOC**(count, memAreaSize)
*Allocates the memory area needed for the number of mbuf headers specified by count. This macro ensures the mbuf headers align on cache line boundaries. This macro evaluates to a pointer to the memory allocated.*

#define **IX_OSAL_MBUF_POOL_DATA_AREA_ALLOC**(count, size, memAreaSize)
*Allocates the memory pool for the data portion of the pool mbufs. The number of mbufs is specified by count. The size of the data portion of each mbuf is specified by size. This macro ensures the mbufs are aligned on cache line boundaries This macro evaluates to a pointer to the memory allocated.*

#define **IX_OSAL_MBUF_POOL_INIT**(count, size, name)
*Wrapper macro for **ixOsalPoolInit()** See function description below for details.*

#define **IX_OSAL_MBUF_NO_ALLOC_POOL_INIT**(bufPtr, dataPtr, count, size, name)
*Wrapper macro for **ixOsalNoAllocPoolInit()** See function description below for details.*

#define **IX_OSAL_MBUF_POOL_GET**(poolPtr)
*Wrapper macro for **ixOsalMbufAlloc()** See function description below for details.*

#define **IX_OSAL_MBUF_POOL_PUT**(bufPtr)
*Wrapper macro for **ixOsalMbufFree()** See function description below for details.*

#define **IX_OSAL_MBUF_POOL_PUT_CHAIN**(bufPtr)
*Wrapper macro for **ixOsalMbufChainFree()** See function description below for details.*

#define **IX_OSAL_MBUF_POOL_SHOW**(poolPtr)
*Wrapper macro for **ixOsalMbufPoolShow()** See function description below for details.*

#define **IX_OSAL_NPE_PRIV_SECTION_PTR**(buffer)
*Wrapper macro for getting pointer to NPE private section.*

#define **IX_OSAL_MBUF_POOL_MDATA_RESET**(bufPtr)
*Wrapper macro for **ixOsalMbufDataPtrReset()** See function description below for details.*

#define **IX_OSAL_MBUF_POOL_UNINIT**(m_pool_ptr)
*Wrapper macro for **ixOsalBuffPoolUninit()** See function description below for details.*

#define **IX_OSAL_CONVERT_OSBUF_TO_IXPBUF**(osBufPtr, ixpBufPtr)
*Convert pre−allocated os−specific buffer format to OSAL IXP_BUF (IX_OSAL_MBUF) format. It is users' responsibility to provide pre−allocated and valid buffer pointers.*

#define **IX_OSAL_CONVERT_IXPBUF_TO_OSBUF**(ixpBufPtr, osBufPtr)
*Convert pre−allocated OSAL IXP_BUF (IX_OSAL_MBUF) format to os−specific buffer pointers.*

---

# Detailed Description

Buffer management module for IxOsal.

---

# Define Documentation

```
#define IX_OSAL_CONVERT_IXPBUF_TO_OSBUF ( ixpBufPtr,
                                          osBufPtr   )
```

Convert pre−allocated OSAL IXP_BUF (IX_OSAL_MBUF) format to os−specific buffer pointers.

***Parameters:***

*ixpBufPtr* (in) − OSAL IXP_BUF pointer
*osBufPtr* (out) − os−specific buffer pointer.

***Returns:***

None

Definition at line **639** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_CONVERT_OSBUF_TO_IXPBUF ( osBufPtr,
                                          ixpBufPtr  )
```

Convert pre−allocated os−specific buffer format to OSAL IXP_BUF (IX_OSAL_MBUF) format. It is users' responsibility to provide pre−allocated and valid buffer pointers.

***Parameters:***

*osBufPtr* (in) − a pre−allocated os−specific buffer pointer.
*ixpBufPtr* (in)− a pre−allocated OSAL IXP_BUF pointer

***Returns:***

None

Definition at line **624** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_ALLOCATED_BUFF_DATA ( m_blk_ptr )

Return the allocated buffer pointer.

Definition at line **305** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_ALLOCATED_BUFF_LEN ( m_blk_ptr )

Return the allocated buffer size.

Definition at line **295** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_FLAGS ( m_blk_ptr )

Return the buffer flags.

Definition at line **224** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_MAX_POOLS

The maximum number of pools that can be allocated, must be a multiple of 32 as required by implementation logic.

***Note:***
This can safely be increased if more pools are required.

Definition at line **69** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_MDATA ( m_blk_ptr )

Return pointer to the data in the mbuf.

Definition at line **194** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_MLEN ( m_blk_ptr )

Return the data length.

Definition at line **203** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_MTYPE ( m_blk_ptr )

Return the data type in the mbuf.

Definition at line **213** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_NET_POOL ( m_blk_ptr )
```

Return pointer to a network pool.

Definition at line **235** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_NEXT_PKT_IN_CHAIN_PTR ( m_blk_ptr )
```

Return pointer to the next packet in the chain.

Definition at line **183** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_NO_ALLOC_POOL_INIT ( bufPtr,
                                          dataPtr,
                                          count,
                                          size,
                                          name     )
```

Wrapper macro for **ixOsalNoAllocPoolInit()** See function description below for details.

***Returns:***
Pointer to the new pool or NULL if the initialization failed.

Definition at line **490** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_OSBUF_PTR ( m_blk_ptr )
```

Return ix_osbuf_ptr field of IX_OSAL_MBUF, which is used to store OS−specific buffer pointer during a buffer conversion.

Definition at line **284** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_PKT_LEN ( m_blk_ptr )
```

Return the total length of all the data in the mbuf chain for this packet.

Definition at line **248** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_DATA_AREA_ALLOC ( count,
                                            size,
                                            memAreaSize )
```

Allocates the memory pool for the data portion of the pool mbufs. The number of mbufs is specified by *count*. The size of the data portion of each mbuf is specified by *size*. This macro ensures the mbufs are aligned on cache line boundaries This macro evaluates to a pointer to the memory allocated.

***Parameters:***

*int* [in] count − the number of mbufs the pool will contain

*int* [in] size − the desired size (in bytes) required for the data portion of each mbuf. Note that this size may be rounded up to ensure alignment on cache−line boundaries.

*int* [out] memAreaSize − the total amount of memory allocated

***Returns:***

void * − a pointer to the allocated memory area

Definition at line **456** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_DATA_AREA_SIZE_ALIGNED ( count,
                                                   size )
```

This macro calculates, from the number of mbufs required and the size of the data portion for each mbuf, the size of the data memory area required. The size is adjusted to ensure alignment on cache line boundaries. This macro is used by **IX_OSAL_MBUF_POOL_DATA_AREA_ALLOC()**.

***Parameters:***

*int* [in] count − The number of mbufs in the pool.

*int* [in] size − The desired size for each mbuf data portion. This size will be rounded up to a multiple of the cache−line size to ensure alignment on cache−line boundaries for each data block.

***Returns:***

int − the total size required for the pool data area (aligned)

Definition at line **402** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_GET ( poolPtr )
```

Wrapper macro for **ixOsalMbufAlloc()** See function description below for details.

Definition at line **501** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_INIT ( count,
                                 size,
```

```
                                        name )
```

Wrapper macro for **ixOsalPoolInit()** See function description below for details.

Definition at line **476** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_MBUF_AREA_ALLOC ( count,
                                            memAreaSize )
```

Allocates the memory area needed for the number of mbuf headers specified by *count*. This macro ensures the mbuf headers align on cache line boundaries. This macro evaluates to a pointer to the memory allocated.

***Parameters:***

*int* [in] count − the number of mbufs the pool will contain

*int* [out] memAreaSize − the total amount of memory allocated

***Returns:***

void * − a pointer to the allocated memory area

Definition at line **423** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_MBUF_AREA_SIZE_ALIGNED ( count )
```

This macro calculates, from the number of mbufs required, the size of the memory area required to contain the mbuf headers for the buffers in the pool. The size to be used for each mbuf header is rounded up to a multiple of the cache−line size, to ensure each mbuf header aligns on a cache−line boundary. This macro is used by **IX_OSAL_MBUF_POOL_MBUF_AREA_ALLOC()**.

***Parameters:***

*int* [in] count − the number of buffers the pool will contain

***Returns:***

int − the total size required for the pool mbuf area (aligned)

Definition at line **375** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_MDATA_RESET ( bufPtr )
```

Wrapper macro for **ixOsalMbufDataPtrReset()** See function description below for details.

Definition at line **555** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_POOL_NAME_LEN
```

The maximum string length of the pool name.

Definition at line **311** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_POOL_PUT ( bufPtr )

Wrapper macro for **ixOsalMbufFree()** See function description below for details.

Definition at line **512** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_POOL_PUT_CHAIN ( bufPtr )

Wrapper macro for **ixOsalMbufChainFree()** See function description below for details.

Definition at line **523** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_POOL_SHOW ( poolPtr )

Wrapper macro for **ixOsalMbufPoolShow()** See function description below for details.

Definition at line **534** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_POOL_SIZE_ALIGN ( size )

This macro takes an integer as an argument and rounds it up to be a multiple of the memory cache−line size.

***Parameters:***

*int* [in] size − the size integer to be rounded up

***Returns:***

int − the size, rounded up to a multiple of the cache−line size

Definition at line **344** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_POOL_UNINIT ( m_pool_ptr )

Wrapper macro for **ixOsalBuffPoolUninit()** See function description below for details.

Definition at line **566** of file **IxOsalBufferMgt.h**.

#define IX_OSAL_MBUF_PRIV ( m_blk_ptr )

Return the private field.

Definition at line **261** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_MBUF_SIGNATURE ( m_blk_ptr )
```

Return the signature field of IX_OSAL_MBUF.

Definition at line **273** of file **IxOsalBufferMgt.h**.

```
#define IX_OSAL_NPE_PRIV_SECTION_PTR ( buffer )
```

Wrapper macro for getting pointer to NPE private section.

Definition at line **545** of file **IxOsalBufferMgt.h**.

# Osal IoMem module

I/O memory and endianess support.

## Data Structures

struct **IxOsalMemoryMap**
*IxOsalMemoryMap structure.*

## Defines

#define **IX_OSAL_MEM_MAP**(physAddr, size)
*Map an I/O mapped physical memory zone to virtual zone and return virtual pointer.*

#define **IX_OSAL_MEM_UNMAP**(virtAddr)
*Unmap a previously mapped I/O memory zone using virtual pointer obtained during the mapping operation. pointer.*

#define **IX_OSAL_MMAP_VIRT_TO_PHYS**(virtAddr)
*This function Converts a virtual address into a physical address, including the dynamically mapped memory.*

#define **IX_OSAL_MMAP_PHYS_TO_VIRT**(physAddr)
*This function Converts a virtual address into a physical address, including the dynamically mapped memory.*

## Enumerations

enum **IxOsalMapEntryType** {
**IX_OSAL_STATIC_MAP**,
**IX_OSAL_DYNAMIC_MAP**
}
*This is an emum for OSAL I/O mem map type.*

enum **IxOsalMapEndianessType** {
**IX_OSAL_BE**,
**IX_OSAL_LE_AC**,
**IX_OSAL_LE_DC**,
**IX_OSAL_LE**
}
*This is an emum for OSAL I/O mem Endianess and Coherency mode.*

# Detailed Description

I/O memory and endianess support.

# Define Documentation

```
#define IX_OSAL_MEM_MAP ( physAddr,
                          size     )
```

Map an I/O mapped physical memory zone to virtual zone and return virtual pointer.

***Parameters:***

*physAddr* − the physical address
*size* − the size

***Returns:***

start address of the virtual memory zone.

***Note:***

This function maps an I/O mapped physical memory zone of the given size into a virtual memory zone accessible by the caller and returns a cookie − the start address of the virtual memory zone. IX_OSAL_MMAP_PHYS_TO_VIRT should NOT therefore be used on the returned virtual address. The memory zone is to be unmapped using IX_OSAL_MEM_UNMAP once the caller has finished using this zone (e.g. on driver unload) using the cookie as parameter. The IX_OSAL_READ/WRITE_LONG/SHORT macros should be used to read and write the mapped memory, adding the necessary offsets to the address cookie.

Definition at line **186** of file **IxOsalIoMem.h**.

```
#define IX_OSAL_MEM_UNMAP ( virtAddr )
```

Unmap a previously mapped I/O memory zone using virtual pointer obtained during the mapping operation. pointer.

***Parameters:***

*virtAddr* − the virtual pointer to the zone to be unmapped.

***Returns:***

none

***Note:***

This function unmaps a previously mapped I/O memory zone using the cookie obtained in the mapping operation. The memory zone in question becomes unavailable to the caller once unmapped and the cookie should be discarded.

This function cannot fail if the given parameter is correct and does not return a value.

Definition at line **209** of file **IxOsalIoMem.h**.

#define IX_OSAL_MMAP_PHYS_TO_VIRT ( physAddr )

This function Converts a virtual address into a physical address, including the dynamically mapped memory.

***Parameters:***

*physAddr* − physical address to convert Return value: corresponding virtual address, or NULL

Definition at line **241** of file **IxOsalIoMem.h**.

#define IX_OSAL_MMAP_VIRT_TO_PHYS ( virtAddr )

This function Converts a virtual address into a physical address, including the dynamically mapped memory.

***Parameters:***

*virtAddr* − virtual address to convert Return value: corresponding physical address, or NULL

Definition at line **225** of file **IxOsalIoMem.h**.

# Enumeration Type Documentation

enum IxOsalMapEndianessType

This is an emum for OSAL I/O mem Endianess and Coherency mode.

***Enumeration values:***

| | |
|---|---|
| *IX_OSAL_BE* | Set map endian mode to Big Endian. |
| *IX_OSAL_LE_AC* | Set map endian mode to Little Endian, Address Coherent. |
| *IX_OSAL_LE_DC* | Set map endian mode to Little Endian, Data Coherent. |
| *IX_OSAL_LE* | Set map endian mode to Little Endian without specifying coherency mode. |

Definition at line **71** of file **IxOsalIoMem.h**.

enum IxOsalMapEntryType

This is an emum for OSAL I/O mem map type.

***Enumeration values:***

| | |
|---|---|
| *IX_OSAL_STATIC_MAP* | Set map entry type to static |

| | |
|---|---|
| | map. |
| *IX_OSAL_DYNAMIC_MAP* | Set map entry type to dynamic map. |

Definition at line **59** of file **IxOsalIoMem.h**.

# Osal basic data types.

Basic data types for Osal.

## Data Structures

struct **IxOsalThreadAttr**
*Thread Attribute.*

struct **IxOsalTimeval**
*Timeval structure.*

## Defines

#define **IX_OSAL_BILLION**
*Alias for 1,000,000,000.*

#define **IX_SUCCESS**
*Success status.*

#define **IX_FAIL**
*Failure status.*

#define **PRIVATE**
*#defined as static, except for debug builds*

#define **IX_OSAL_INLINE**
*Alias for __inline.*

#define **IX_OSAL_INLINE_EXTERN**
*Alias for __inline extern.*

#define **IX_OSAL_LOG_ERROR**
*Alias for −1, used as log function error status.*

#define **IX_OSAL_WAIT_FOREVER**
*Definition for timeout forever, OS−specific.*

#define **IX_OSAL_WAIT_NONE**
*Definition for timeout 0, OS−specific.*

#define **IX_OSAL_THREAD_DEFAULT_SCHED_POLICY**
*Default Thread Scheduling Policy, OS−specific.*

#define **IX_OSAL_THREAD_DEFAULT_STACK_SIZE**
*Default thread stack size, OS−specific.*

#define **IX_OSAL_THREAD_MAX_STACK_SIZE**
*Max stack size, OS−specific.*

#define **IX_OSAL_MAX_THREAD_NAME_LEN**
*Max size of thread name.*

#define **IX_OSAL_MIN_THREAD_PRIORITY**
*Min thread priority, OS−specific.*

#define **IX_OSAL_DEFAULT_THREAD_PRIORITY**
*Default thread priority, OS−specific.*

#define **IX_OSAL_MAX_THREAD_PRIORITY**
*Max thread priority, OS−specific.*

## Typedefs

typedef UINT32 **IX_STATUS**
*OSAL status.*

typedef volatile UINT32 **VUINT32**
*VUINT32.*

typedef volatile INT32 **VINT32**
*VINT32.*

typedef void(* **IxOsalVoidFnVoidPtr** )(void *)
*Void function pointer prototype.*

typedef UINT32 **IxOsalTimer**
*IxOsalTimer.*

typedef IxOsalOsMutex **IxOsalMutex**
*IxOsalMutex.*

typedef IxOsalOsFastMutex **IxOsalFastMutex**
*IxOsalFastMutex.*

typedef IxOsalOsThread **IxOsalThread**
*IxOsalThread.*

typedef IxOsalOsSemaphore **IxOsalSemaphore**
*IxOsalSemaphore.*

typedef IxOsalOsMessageQueue **IxOsalMessageQueue**
*IxOsalMessageQueue.*

## Enumerations

enum **IxOsalLogDevice** {
**IX_OSAL_LOG_DEV_STDOUT**,
**IX_OSAL_LOG_DEV_STDERR**,
**IX_OSAL_LOG_DEV_HEX_DISPLAY**,
**IX_OSAL_LOG_DEV_ASCII_DISPLAY**
}
*This is an emum for OSAL log devices.*

enum **IxOsalLogLevel** {
**IX_OSAL_LOG_LVL_NONE**,
**IX_OSAL_LOG_LVL_USER**,
**IX_OSAL_LOG_LVL_FATAL**,
**IX_OSAL_LOG_LVL_ERROR**,
**IX_OSAL_LOG_LVL_WARNING**,
**IX_OSAL_LOG_LVL_MESSAGE**,
**IX_OSAL_LOG_LVL_DEBUG1**,
**IX_OSAL_LOG_LVL_DEBUG2**,
**IX_OSAL_LOG_LVL_DEBUG3**,
**IX_OSAL_LOG_LVL_ALL**
}
*This is an emum for OSAL log trace level.*

## Detailed Description

Basic data types for Osal.

## Define Documentation

#define IX_FAIL

Failure status.

Definition at line **107** of file **IxOsalTypes.h**.

#define IX_OSAL_BILLION

Alias for 1,000,000,000.

Definition at line **64** of file **IxOsalTypes.h**.

#define IX_OSAL_DEFAULT_THREAD_PRIORITY

Default thread priority, OS−specific.

Definition at line **409** of file **IxOsalTypes.h**.

#define IX_OSAL_INLINE

Alias for __inline.

Definition at line **131** of file **IxOsalTypes.h**.

#define IX_OSAL_INLINE_EXTERN

Alias for __inline extern.

Definition at line **163** of file **IxOsalTypes.h**.

#define IX_OSAL_LOG_ERROR

Alias for −1, used as log function error status.

Definition at line **201** of file **IxOsalTypes.h**.

#define IX_OSAL_MAX_THREAD_NAME_LEN

Max size of thread name.

Definition at line **385** of file **IxOsalTypes.h**.

#define IX_OSAL_MAX_THREAD_PRIORITY

Max thread priority, OS−specific.

Definition at line **419** of file **IxOsalTypes.h**.

#define IX_OSAL_MIN_THREAD_PRIORITY

Min thread priority, OS−specific.

Definition at line **398** of file **IxOsalTypes.h**.

#define IX_OSAL_THREAD_DEFAULT_SCHED_POLICY

Default Thread Scheduling Policy, OS−specific.

Definition at line **354** of file **IxOsalTypes.h**.

#define IX_OSAL_THREAD_DEFAULT_STACK_SIZE

Default thread stack size, OS−specific.

Definition at line **365** of file **IxOsalTypes.h**.

#define IX_OSAL_THREAD_MAX_STACK_SIZE

Max stack size, OS−specific.

Definition at line **375** of file **IxOsalTypes.h**.

#define IX_OSAL_WAIT_FOREVER

Definition for timeout forever, OS−specific.

Definition at line **265** of file **IxOsalTypes.h**.

#define IX_OSAL_WAIT_NONE

Definition for timeout 0, OS−specific.

Definition at line **275** of file **IxOsalTypes.h**.

#define IX_SUCCESS

Success status.

Definition at line **95** of file **IxOsalTypes.h**.

#define PRIVATE

#defined as static, except for debug builds

Definition at line **115** of file **IxOsalTypes.h**.

# Typedef Documentation

typedef UINT32 IX_STATUS

OSAL status.

***Note:***
Possible OSAL return status include IX_SUCCESS and IX_FAIL.

Definition at line **34** of file **IxOsalTypes.h**.

typedef IxOsalOsFastMutex IxOsalFastMutex

IxOsalFastMutex.

***Note:***
FastMutex handle, OS−specific

Definition at line **294** of file **IxOsalTypes.h**.

typedef IxOsalOsMessageQueue IxOsalMessageQueue

IxOsalMessageQueue.

***Note:***
Message Queue handle, OS−specific

Definition at line **321** of file **IxOsalTypes.h**.

typedef IxOsalOsMutex IxOsalMutex

IxOsalMutex.

***Note:***
Mutex handle, OS−specific

Definition at line **285** of file **IxOsalTypes.h**.

typedef IxOsalOsSemaphore IxOsalSemaphore

IxOsalSemaphore.

***Note:***

Semaphore handle, OS−specific

Definition at line **312** of file **IxOsalTypes.h**.

typedef IxOsalOsThread IxOsalThread

IxOsalThread.

***Note:***
Thread handle, OS−specific

Definition at line **303** of file **IxOsalTypes.h**.

typedef UINT32 IxOsalTimer

IxOsalTimer.

***Note:***
OSAL timer handle

Definition at line **254** of file **IxOsalTypes.h**.

typedef void(* IxOsalVoidFnVoidPtr)(void *)

Void function pointer prototype.

***Note:***
accepts a void pointer parameter and does not return a value.

Definition at line **230** of file **IxOsalTypes.h**.

typedef volatile INT32 VINT32

VINT32.

***Note:***
volatile INT32

Definition at line **48** of file **IxOsalTypes.h**.

typedef volatile UINT32 VUINT32

VUINT32.

***Note:***

volatile UINT32

Definition at line **41** of file **IxOsalTypes.h**.

# Enumeration Type Documentation

enum IxOsalLogDevice

This is an emum for OSAL log devices.

***Enumeration values:***

| | |
|---|---|
| *IX_OSAL_LOG_DEV_STDOUT* | standard output (implemented by default) |
| *IX_OSAL_LOG_DEV_STDERR* | standard error (implemented |
| *IX_OSAL_LOG_DEV_HEX_DISPLAY* | hexadecimal display (not implemented) |
| *IX_OSAL_LOG_DEV_ASCII_DISPLAY* | ASCII−capable display (not implemented). |

Definition at line **184** of file **IxOsalTypes.h**.

enum IxOsalLogLevel

This is an emum for OSAL log trace level.

***Enumeration values:***

| | |
|---|---|
| *IX_OSAL_LOG_LVL_NONE* | No trace level. |
| *IX_OSAL_LOG_LVL_USER* | Set trace level to user. |
| *IX_OSAL_LOG_LVL_FATAL* | Set trace level to fatal. |
| *IX_OSAL_LOG_LVL_ERROR* | Set trace level to error. |
| *IX_OSAL_LOG_LVL_WARNING* | Set trace level to warning. |
| *IX_OSAL_LOG_LVL_MESSAGE* | Set trace level to message. |
| *IX_OSAL_LOG_LVL_DEBUG1* | Set trace level to debug1. |
| *IX_OSAL_LOG_LVL_DEBUG2* | Set trace level to debug2. |
| *IX_OSAL_LOG_LVL_DEBUG3* | Set trace level to debug3. |
| *IX_OSAL_LOG_LVL_ALL* | Set trace level to all. |

Definition at line **208** of file **IxOsalTypes.h**.

# IxOsalMemoryMap Struct Reference

## [Osal IoMem module]

IxOsalMemoryMap structure.

---

## Detailed Description

IxOsalMemoryMap structure.

---

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

- **IxOsalIoMem.h**

# IxOsalThreadAttr Struct Reference

## [Osal basic data types.]

Thread Attribute.

## Data Fields

char * **name**
*name*

UINT32 **stackSize**
*stack size*

UINT32 **priority**
*priority*

---

## Detailed Description

Thread Attribute.

***Note:***
Default thread attribute

Definition at line **339** of file **IxOsalTypes.h**.

---

## Field Documentation

char* IxOsalThreadAttr::name

name

Definition at line **341** of file **IxOsalTypes.h**.

UINT32 IxOsalThreadAttr::priority

priority

Definition at line **343** of file **IxOsalTypes.h**.

UINT32 IxOsalThreadAttr::stackSize

stack size

Definition at line **342** of file **IxOsalTypes.h**.

---

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

- **IxOsalTypes.h**

# IxOsalTimeval Struct Reference

## [Osal basic data types.]

Timeval structure.

## Data Fields

UINT32 **secs**
*seconds*

UINT32 **nsecs**
*nanoseconds*

---

## Detailed Description

Timeval structure.

***Note:***
Contain subfields of seconds and nanoseconds..

Definition at line **240** of file **IxOsalTypes.h**.

---

## Field Documentation

UINT32 IxOsalTimeval::nsecs

nanoseconds

Definition at line **243** of file **IxOsalTypes.h**.

UINT32 IxOsalTimeval::secs

seconds

Definition at line **242** of file **IxOsalTypes.h**.

---

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

- **IxOsalTypes.h**