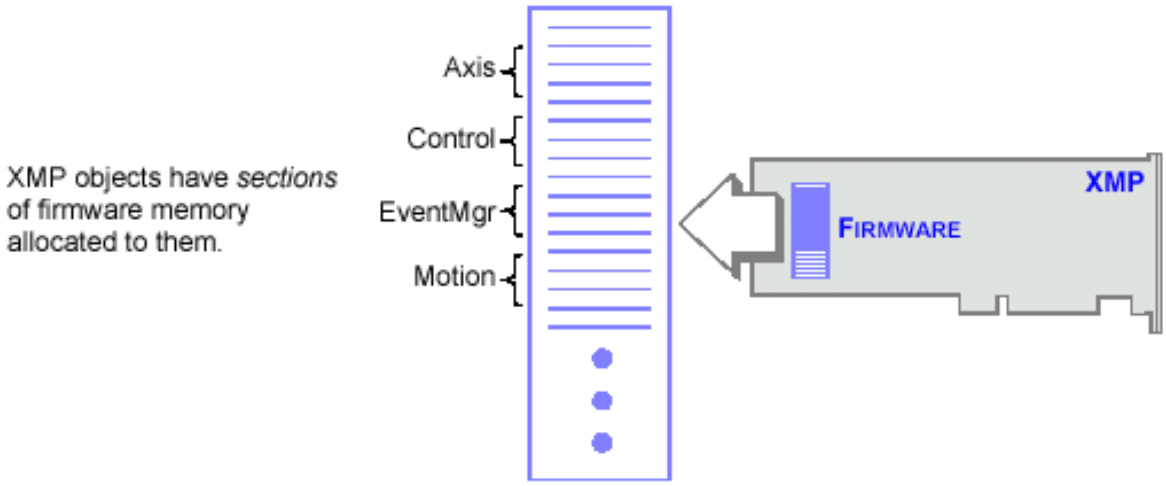
Platform Objects

Introduction

The **Platform** module provides a common interface to platform-specific functionality, such as memory allocation, resource locking, interrupts, signalling, and others.

The Platform object provides low level *platform-specific* functionality and depends upon the combination of the operating system and the C compiler used for development. Although there are circumstances when your application will want to use Platform functions directly, your application won't typically call Platform functions; usually the MPI functions will call Platform functions.

The **meiObjectGive/Take(...)** methods all use the **meiPlatformLockGive/Take(...)** methods. When you take a lock, you take exclusive access to the resource (i.e., the section of XMP firmware memory associated with that Object). When you give a lock, you release (give up) that exclusive access. Think of it as TakeAccessOf and GiveUpAccess.



Methods

meiPlatform**Atol**	Convert a numeric string to a long.
meiPlatform**Key**	Return an input character if an input character is available.
meiPlatform**Trace**	Display printf(...)-style trace information
meiPlatform**TraceEol**	Set the end-of-line (eol) to be used by meiPlatformTrace(...).
meiPlatform**TraceFile**	
meiPlatform**TraceFunction**	

Data Types

MEIPlatform**FileMode**

MEIPlatform**Message**

meiPlatformAtol

Declaration `long meiPlatformAtol(const char *ascii)`

Required Header stdmei.h

Description **PlatformAtol** converts a numeric string to a long. This function returns the converted value as a long.

***ascii**	string to be converted

Returns	converted the numeric text string ascii to a ***long*** and returned it

See Also

meiPlatformKey

Declaration

```
long meiPlatformKey(MPIWait wait)
```

Required Header

stdmei.h

Description

PlatformKey returns an input character (typically a keystroke) if an input character is available.
If an input charater is not available, *PlatformKey* waits ***wait*** milliseconds for an input character to become available.

If "wait" is	*Then*
MPIWaitFOREVER (-1)	*PlatformKey* will wait for an input character forever
MPIWaitPOLL (0)	*PlatformKey* will return immediately
a value (not -1 or 0)	*PlatformKey* will wait for an input character for ***wait*** milliseconds

Return Values	
-1	if no input character was available
0	*PlatformKey* has read a non-zero character (typically a function key or other non-ASCII value), and **meiPlatformKey(...)** should be called AGAIN immediately to receive that non-zero character
a value (not -1 or 0) (an ASCII character)	(typically a keystroke) if an input character is available

See Also

meiPlatformTrace

Declaration

```
long meiPlatformTrace(const char *format, …)
```

Required Header

stdmei.h

Description

PlatformTrace displays **printf(...)**-style trace information. An *end-of-line* character will be appended to the output, newline by default.

Library modules call **meiTrace#(...)**, a macro which can be conditionally compiled to call **meiPlatformTrace(...)** (by defining the symbol MEI_TRACE when building the library).
Otherwise, calls to **meiTrace#(...)** are removed by the C preprocessor.

Return Values

MPIMessageOK	if *PlatformTrace* successfully executes

See Also

meiPlatformTraceEol

Declaration `char meiPlatformTraceEol(char eol)`

Required Header stdmei.h

Description The **PlatformTraceEol** function sets the *end-of-line* (***eol***) character that will be used by meiPlatformTrace(...).

Returns	the previous end-of-line character used by meiPlatformTrace(...)

See Also meiPlatformTrace

meiPlatformTraceFile

Declaration `long meiPlatformTraceFile(const char *fileName)`

Required Header stdmei.h

Description **PlatformTraceFile** redirects trace output to ***fileName***, after first closing any *previously opened* trace file. If no trace file has been explicitly opened, trace output will go to standard output.

Return Values	
MPIMessageOK	if *PlatformTraceFile* successfully closes any *previously opened* trace file and redirects trace output to ***fileName***

See Also meiPlatformTrace

meiPlatformTraceFunction

Declaration

```
MEITraceFunction meiPlatformTraceFunction(MEITraceFunction traceFunction)
```

Required Header

stdmei.h

Description

PlatformTraceFunction displays the trace output using ***traceFunction***, and replaces the internal function that was called by meiPlatformTrace(...) to display the trace output. Use *PlatformTraceFunction* to enable your application to take control of the display of trace output.

Return Values

the previous *traceFunction*	if there is a previous function
NULL	if no ***traceFunction*** has been specified (the default trace function is used)

See Also

meiPlatformTrace

MEIPlatformFileMode

MEIPlatformFileMode

```
typedef enum {
    MEIPlatformFileModeREAD,    /* default */
    MEIPlatformFileModeWRITE,
    MEIPlatformFileModeTEXT,    /* default */
    MEIPlatformFileModeBINARY,
} MEIPlatformFileMode;
```

Description

PlatformFileMode is an enumeration that is used as an argument for methods that open files.

See Also

MEIPlatformMessage

MEIPlatformMessage

```
typedef enum {

    MEIPlatformMessagePLATFORM_INVALID,
    MEIPlatformMessageDEVICE_INVALID,
    MEIPlatformMessageDEVICE_ERROR,
    MEIPlatformMessageDEVICE_MAP_ERROR,
} MEIPlatformMessage;
```

Description

MEIPlatformMessageDEVICE_INVALID	Unable to communicate to XMP device driver. (WIN32 Only: If you receive this error, verify that that device driver has been started.)
MEIPlatformMessageDEVICE_ERROR	A low level device driver call has failed. If you receive this error, check for hardware resource conflicts such as base address and IRQ number.
MEIPlatformMessageDEVICE_MAP_ERROR	The device driver is unable to map the XMP memory to host memory system. If you receive this error, check for hardware base address conflict.

See Also