MangoBIOS

Host API Guide

Table of Contents

MangoBIOS

1 Introduction

The code for Mango Library is written in C but can work with C or C++.

The Mango BIOS Library uses the same abstraction for all supported O/Ss allowing portability of code between supported operating systems. All APIs in the MANGOBIOS LIB are built with the ability to add new features and new capabilities while still maintaining the same C/C++ interfaces. This makes all code compiled with this library forward compatible.

2 Overview

The Mango BIOS library is a generic library for all Mango DSPs boards. The library enables accessing the board via its PCI registers and base addresses:

Accesses device memory space using PCI BAR registers.

Enables interrupting of devices through respective device's doorbell registers or doorbell bits.

Handles incoming PCI interrupts from the target board.

Reads and writes PCI configuration registers.

Mango BIOS allows allocation of shared memory on PC side. This memory serves as a common memory for both Host and DSP and enables transferring data and implementing various protocols for communication between Host and DSP.

Mango BIOS library uses an operating-system dependent implementation driver. Accessing the driver utilities from the application layer is not done directly but rather using the Mango BIOS API. This separates between the operating system dependent layer and the application non-operating system dependant layer, thus enabling portability of the application on various operating systems.

Mango DSP provides all software pacakages including the Mango Library and additonal board-specific documentation package. Utilization and abstraction are implemented in this package.

2.1 Processes and Threads

Process and thread usage

Processes and Threads

Mango BIOS is usable within different thread contexts, and different processes with some constraints. Mango memory allocated and mapped within one process is not available to other processes (this may change in future versions), but is available to other threads within the same process. For all Mango BIOS functions, and libraries working with Mango BIOS, there is no protection of resources, therefore all resources must be protected at the user level (ie semaphores and locks). Dependent on the card and the split up of devices, this is either recommended but not needed, or very necessary.

Example: Seagull_PMC.lib: write_memory does two things when it writes memory, it changes a page register on the target dsp, and then performs the write. If performing 2 write_memory functions from two different threads and to the same target dsp, you may have problems with thread safety.

Thread A wants to write memory to the beginning of SDRAM (0x80000000).

Thread B wants to write memory to the beginning of ISRAM (0x00000000).

Thread A Thread B Running DSPP on target

Time 0 Neither Unknown

Time 1 Change DSPP A 512

Time 2 Change DSPP B 0

Time 3 Write Memory B 0

Time 4 Write Memory A 0

What can be seen is that without some form of user level protections, thread A's write can be ruined.

Although when working on the same Seagull PMC, if both threads work with different target DSP's, then these problems will not occur, and user level protections of this sort will not be necessary. In contrast, work on the Advanced Seagull Board will still require protection even when working with different target DSP's if they both are behind the same non-transparent PCI to PCI bridge. In all cases when the exact makeup of the boards and libraries is unknown, user-level protections are recommended.

3 Symbol Reference

Files

File	Description
MangoBios.h (⧉ see page 35)	MangoBios header file
MangoError.h (⧉ see page 36)	MangoBios header file

Functions

Function	Description
MANGOBIOS_close (⧉ see page 8)	Closes MangoBios library for this process
MANGOBIOS_deviceClose (⧉ see page 9)	Closes MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceGetProperty (⧉ see page 10)	Gets a MANGOBIOS_deviceProp_t
MANGOBIOS_deviceOpen (⧉ see page 11)	Opens MANGOBIOS_deviceHandle_t
MANGOBIOS_devicePciRegRead (⧉ see page 12)	Reads a pci register from a MANGOBIOS_deviceHandle_t
MANGOBIOS_devicePciRegWrite (⧉ see page 13)	Writes a pci register to a MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceRead (⧉ see page 14)	Reads from a MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceSetProperty (⧉ see page 15)	Sets a MANGOBIOS_deviceProp_t
MANGOBIOS_deviceWrite (⧉ see page 16)	Writes to a MANGOBIOS_deviceHandle_t
MANGOBIOS_getDeviceHandles (⧉ see page 17)	Fills a MANGOBIOS_deviceHandle_t array
MANGOBIOS_getNumDevices (⧉ see page 19)	Gets number of devices of MANGOBIOS_deviceType_t type
MANGOBIOS_getVersion (⧉ see page 20)	Gets MANGOBIOS_version_t
MANGOBIOS_isrConnect (⧉ see page 21)	Connects an ISR to a MANGOBIOS_deviceHandle_t
MANGOBIOS_isrDisconnect (⧉ see page 22)	Disconnects an ISR from a MANGOBIOS_deviceHandle_t
MANGOBIOS_memoryAlloc (⧉ see page 23)	Allocates physical memory to MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryFree (⧉ see page 24)	Frees a MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryMap (⧉ see page 25)	Maps physical memory of a MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryUnmap (⧉ see page 26)	Unmaps a MANGOBIOS_memoryHandle_t
MANGOBIOS_open (⧉ see page 27)	Initializes MangoBios Library for this process

Macros

Macro	Description
IS_BIG_ENDIAN (⧉ see page 31)	defined(_BIG_ENDIAN) defined(__vxworks) -> defined(linux) || defined(__SVR4)
MANGOBIOS_version_build (⧉ see page 32)	builds versioning information
POSIX_COMPLIANT (⧉ see page 33)	defined(linux) || defined(__SVR4)

Types

Type	Description
MANGOERROR_error_t (⧉ see page 29)	typedef of structure MANGOERROR_error_e (⧉ see page 6)

Structs, Records, Enums

Struct, Record, Enum	Description
MANGOERROR_error_e (⧉ see page 6)	enum of possible errors returned in MangoBios based projects

3.1 Structs, Records, Enums

Enumerations

Enumeration	Description
MANGOERROR_error_e (see page 6)	enum of possible errors returned in MangoBios based projects

3.1.1 MANGOERROR_error_e

```
enum MANGOERROR_error_e {
  MANGOERROR_SUCCESS = 0x20000000,
  MANGOERROR_FAILURE,
  MANGOERROR_TIMEOUT,
  MANGOERROR_ERR_INVALID_HANDLE,
  MANGOERROR_ERR_NOT_IMPLEMENTED,
  MANGOERROR_COFF_FORMAT_ERROR,
  MANGOERROR_ERR_INVALID_PARAMETER,
  MANGOERROR_INSUFFICIENT_RESOURCES,
  MANGOERROR_INVALID_CONFIGURATION,
  MANGOERROR_RESOURCE_NOT_READY
};
```

File

MangoError.h (see page 36)

Members

Members	Description
`MANGOERROR_SUCCESS = 0x20000000`	success
`MANGOERROR_FAILURE`	failure
`MANGOERROR_TIMEOUT`	timeout
`MANGOERROR_ERR_INVALID_HANDLE`	invalid handle
`MANGOERROR_ERR_NOT_IMPLEMENTED`	not implemented
`MANGOERROR_COFF_FORMAT_ERROR`	coff format error
`MANGOERROR_ERR_INVALID_PARAMETER`	invalid parameter
`MANGOERROR_INSUFFICIENT_RESOURCES`	insufficient resources
`MANGOERROR_INVALID_CONFIGURATION`	invalid configuration
`MANGOERROR_RESOURCE_NOT_READY`	resource not ready

Description

enum of possible errors returned in MangoBios based projects

3.2 Functions

Functions

Function	Description
MANGOBIOS_close (☐ see page 8)	Closes MangoBios library for this process
MANGOBIOS_deviceClose (☐ see page 9)	Closes MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceGetProperty (☐ see page 10)	Gets a MANGOBIOS_deviceProp_t
MANGOBIOS_deviceOpen (☐ see page 11)	Opens MANGOBIOS_deviceHandle_t
MANGOBIOS_devicePciRegRead (☐ see page 12)	Reads a pci register from a MANGOBIOS_deviceHandle_t
MANGOBIOS_devicePciRegWrite (☐ see page 13)	Writes a pci register to a MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceRead (☐ see page 14)	Reads from a MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceSetProperty (☐ see page 15)	Sets a MANGOBIOS_deviceProp_t
MANGOBIOS_deviceWrite (☐ see page 16)	Writes to a MANGOBIOS_deviceHandle_t
MANGOBIOS_getDeviceHandles (☐ see page 17)	Fills a MANGOBIOS_deviceHandle_t array
MANGOBIOS_getNumDevices (☐ see page 19)	Gets number of devices of MANGOBIOS_deviceType_t type
MANGOBIOS_getVersion (☐ see page 20)	Gets MANGOBIOS_version_t
MANGOBIOS_isrConnect (☐ see page 21)	Connects an ISR to a MANGOBIOS_deviceHandle_t
MANGOBIOS_isrDisconnect (☐ see page 22)	Disconnects an ISR from a MANGOBIOS_deviceHandle_t
MANGOBIOS_memoryAlloc (☐ see page 23)	Allocates physical memory to MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryFree (☐ see page 24)	Frees a MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryMap (☐ see page 25)	Maps physical memory of a MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryUnmap (☐ see page 26)	Unmaps a MANGOBIOS_memoryHandle_t
MANGOBIOS_open (☐ see page 27)	Initializes MangoBios Library for this process

3.2.1 MANGOBIOS_close

```
MANGOERROR_error_t MANGOBIOS_close();
```

Summary

Closes MangoBios library for this process

File

MangoBios.h (see page 35)

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success

Description

Closes all handles previously opened in MANGOBIOS_open (see page 27)

Remarks

None

Example

```
int errorCode;
errorCode = MANGOBIOS_close(
 );
```

3.2.2 MANGOBIOS_deviceClose

```
MANGOERROR_error_t MANGOBIOS_deviceClose(MANGOBIOS_deviceHandle_t * handle);
```

Summary

Closes MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Invalid handle

Description

Closes device handle which was previously opened with MANGOBIOS_deviceOpen (see page 11)

Remarks

None

Example

```
int errorCode;
errorCode = MANGOBIOS_deviceClose(
 &device_handle
 );
```

3.2.3 MANGOBIOS_deviceGetProperty

```
MANGOERROR_error_t MANGOBIOS_deviceGetProperty(MANGOBIOS_deviceHandle_t * handle,
MANGOBIOS_deviceProp_t property, int * val);
```

Summary

Gets a MANGOBIOS_deviceProp_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`MANGOBIOS_deviceProp_t property`	Property to get
`int * val`	Pointer for value

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Invalid handle Illegal property choice

Description

Gets a property for this handle

Remarks

None

Example

```
int errorCode;
int bus_no;
errorCode = MANGOBIOS_deviceGetProperty(
 &device_handle,
 MANGOBIOS_deviceProp_Bus,
 &bus_no
 );
```

3.2.4 MANGOBIOS_deviceOpen

```
MANGOERROR_error_t MANGOBIOS_deviceOpen(MANGOBIOS_deviceHandle_t * handle, const
MANGOBIOS_deviceAttrs_t * attrs);
```

Summary

Opens MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`const MANGOBIOS_deviceAttrs_t * attrs`	NULL

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Invalid handle
`MANGOERROR_INSUFFICIENT_RESOURCES`	Failed malloc
`Other value`	Error from OS

Description

Opens device handle

Remarks

None

Example

```
int errorCode;
errorCode = MANGOBIOS_deviceOpen(
 handles[i],
 NULL
 );
```

3.2.5 MANGOBIOS_devicePciRegRead

```
MANGOERROR_error_t MANGOBIOS_devicePciRegRead(MANGOBIOS_deviceHandle_t * handle, int
regOffset, void * regVal, int size);
```

Summary

Reads a pci register from a MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`int regOffset`	Byte offset into pci register space
`void * regVal`	Pointer for value
`int size`	Number of bytes to read (1,2,4)

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Handle is invalid
`Other value`	Error from OS

Description

Reads a pci register from a device opened with MANGOBIOS_deviceOpen (see page 11)

Remarks

None

Example

```
int errorCode;
int dev_ven_id;
errorCode = MANGOBIOS_devicePciRegRead(
 &device_handle,
 0x0, (Offset in PCI register space for the Device/Vendor ID)
 &dev_ven_id,
 4
 );
```

3.2.6 MANGOBIOS_devicePciRegWrite

```
MANGOERROR_error_t MANGOBIOS_devicePciRegWrite(MANGOBIOS_deviceHandle_t * handle, int
regOffset, const void * regVal, int size);
```

Summary

Writes a pci register to a MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`int regOffset`	Byte offset into pci register space
`const void * regVal`	Pointer to value
`int size`	Number of bytes to read (1,2,4)

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Handle is invalid
`Other value`	Error from OS

Description

Writes a pci register to a device opened with MANGOBIOS_deviceOpen (see page 11)

Remarks

None

Example

```
int errorCode;
int bar0 = 0xffa00000;
errorCode = MANGOBIOS_devicePciRegWrite(
 &device_handle,
 0x10, (Offset in PCI register space for Base Address Register 0)
 &bar0,
 4
 );
```

3.2.7 MANGOBIOS_deviceRead

```
MANGOERROR_error_t MANGOBIOS_deviceRead(MANGOBIOS_deviceHandle_t * handle, int bar, int
offset, void * buff, int size, MANGOBIOS_quanta_t quanta, int increment_flag);
```

Summary

Reads from a MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`int bar`	Number of PCI BAR
`int offset`	Offset in bytes from start of given BAR
`void * buff`	Pointer for received data
`int size`	Number of bytes to read
`MANGOBIOS_quanta_t quanta`	Number of bytes to be read on each access of the PCI bus
`int increment_flag`	True increments the address being read from by the quanta being read after each read, False rereads from the same PCI location each time

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Handle is invalid Increment_flag is false and the quanta is Q_ANY
`Other value`	Error from OS

Description

Reads from any bar on a device opened with MANGOBIOS_open (see page 27)

Remarks

Whether increment_flag is true or false, the buff variable will be fully filled. It is only the PCI address that is being incremented dependent on the increment_flag, the local buffer is always incremented.

Example

```
int errorCode;
void * buffer = malloc(0x1000);
if(!buffer)
 return -1;
errorCode = MANGOBIOS_deviceRead(
 &device_handle,
 2,
 0,
 buffer,
 0x1000,
 Q_32,
 1
 );
```

3.2.8 MANGOBIOS_deviceSetProperty

```
MANGOERROR_error_t MANGOBIOS_deviceSetProperty(MANGOBIOS_deviceHandle_t * handle,
MANGOBIOS_deviceProp_t property, int val);
```

Summary

Sets a MANGOBIOS_deviceProp_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device : Value
`MANGOBIOS_deviceProp_t property`	Property to be set
`int val`	Value

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_ERR_INVALID_PARAMETER`	Failure, not implemented

Description

Sets a property for this handle

Remarks

None

3.2.9 MANGOBIOS_deviceWrite

```
MANGOERROR_error_t MANGOBIOS_deviceWrite(MANGOBIOS_deviceHandle_t * handle, int bar, int
offset, const void * buff, int size, MANGOBIOS_quanta_t quanta, int increment_flag);
```

Summary

Writes to a MANGOBIOS_deviceHandle_t

File

MangoBios.h (⧉ see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`int bar`	Number of PCI BAR
`int offset`	Offset in bytes from start of given BAR
`const void * buff`	Pointer for received data
`int size`	Number of bytes to read
`MANGOBIOS_quanta_t quanta`	Number of bytes to be read on each access of the PCI bus
`int increment_flag`	True increments the address being read from by the quanta being read after each read, False rereads from the same PCI location each time

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	Handle is invalid Increment_flag is false and the quanta is Q_ANY
`Other value`	Error from OS

Description

Writes to any bar on a device opened with MANGOBIOS_open (⧉ see page 27)

Remarks

Whether increment_flag is true or false, the buff variable will be fully filled. It is only the PCI address that is being incremented dependent on the increment_flag, the local buffer is always incremented.

Example

```
int errorCode;
void * buffer = malloc(0x1000);
memset(buffer, 0, 0x1000);
if(!buffer)
 return -1;
errorCode = MANGOBIOS_deviceWrite(
 &device_handle,
 2,
 0,
 buffer,
 0x1000,
 Q_32,
 1
 );
```

3.2.10 MANGOBIOS_getDeviceHandles

```
MANGOERROR_error_t MANGOBIOS_getDeviceHandles(const MANGOBIOS_deviceType_t * type,
MANGOBIOS_deviceHandle_t * handle);
```

Summary

Fills a MANGOBIOS_deviceHandle_t array

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`const MANGOBIOS_deviceType_t * type`	The device type being requested A NULL value will return all supported devices
`MANGOBIOS_deviceHandle_t * handle`	Previously allocated array of MANGOBIOS_deviceHandle_t handles

Returns

Status

Return Values

Return Values	Description
MANGOERROR_SUCCESS	Success
MANGOERROR_ERR_INVALID_PARAMETER	'handle' is NULL
MANGOERROR_ERR_INVALID_HANDLE	Failed CreateFile for device
Other value	Error from OS

Description

Fills in the handle array with devices matching type

Remarks

MANGOBIOS_getDeviceHandles expects a non-NULL input for 'handle.' To retrieve the number of devices matching 'type' in the system, call MANGOBIOS_getNumDevices (see page 19) first.

'type' should be the same in both MANGOBIOS_getNumDevices (see page 19), and in MANGOBIOS_getDeviceHandles, otherwise the number of devices that the user will allocate could be insufficient to receive the number of devices that MANGOBIOS_getDeviceHandles will return, causing an overflow.

Example

```
int errorCode;
int num;
MANGOBIOS_deviceType_t type = {0x8086, 0xB555, 0x0000, 0x0000};
MANGOBIOS_deviceHandle_t * handles;

errorCode = MANGOBIOS_getNumDevices(
 &type,
 &num
 );
if(errorCode != MANGOERROR_SUCCESS)
 return -1;
handles = (MANGOBIOS_deviceHandle_t *)malloc(
 sizeof(MANGOBIOS_deviceHandle_t) * num
 );
errorCode = MANGOBIOS_getDeviceHandles(
 &type,
 handles
 );
```

```
if(errorCode != MANGOERROR_SUCCESS)
 return -1;
```

3.2.11 MANGOBIOS_getNumDevices

```
MANGOERROR_error_t MANGOBIOS_getNumDevices(const MANGOBIOS_deviceType_t * type, int *
numDevices);
```

Summary

Gets number of devices of MANGOBIOS_deviceType_t type

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`const MANGOBIOS_deviceType_t * type`	The device type being requested A NULL value will return all supported devices
`int * numDevices`	Pointer for number of devices found

Returns

Status

Return Values

Return Values	Description
MANGOERROR_SUCCESS	Success
MANGOERROR_ERR_INVALID_PARAMETER	numDevices is NULL
MANGOERROR_INSUFFICIENT_RESOURCES	Failed malloc
MANGOERROR_ERR_INVALID_HANDLE	Failed CreateFile for device (Needed for matching 'type')
Other value	Error from OS

Description

Sets numDevices to number of matching devices to type found in the system

Remarks

'type' should be the same in both MANGOBIOS_getNumDevices, and in MANGOBIOS_getDeviceHandles (see page 17), otherwise the number of devices that the user will allocate could be insufficient to receive the number of devices that MANGOBIOS_getDeviceHandles (see page 17) will return, causing an overflow.

Example

```
int errorCode;
int num;
MANGOBIOS_deviceType_t type = {0x8086, 0xB555, 0x0000, 0x0000};
MANGOBIOS_deviceHandle_t * handles;

errorCode = MANGOBIOS_getNumDevices(
 &type,
 &num
 );
if(errorCode != MANGOERROR_SUCCESS)
 return -1;
handles = (MANGOBIOS_deviceHandle_t *)malloc(
 sizeof(MANGOBIOS_deviceHandle_t) * num
 );
errorCode = MANGOBIOS_getDeviceHandles(
 &type,
 handles
 );
if(errorCode != MANGOERROR_SUCCESS)
 return -1;
```

3.2.12 MANGOBIOS_getVersion

```
MANGOERROR_error_t MANGOBIOS_getVersion(MANGOBIOS_version_t * version);
```

Summary

Gets MANGOBIOS_version_t

File

MangoBios.h (see page 35)

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success

Description

Gets versioning information about this library

Remarks

None

Example

```
int errorCode;
MANGOBIOS_version_t version;
errorCode = MANGOBIOS_getVersion(
 &version
 );
```

3.2.13 MANGOBIOS_isrConnect

```
MANGOERROR_error_t MANGOBIOS_isrConnect(MANGOBIOS_deviceHandle_t * handle, int isrInitNum,
int isrNum, int isrShutdownNum, void * buff);
```

Summary

Connects an ISR to a MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	Handle to device
`int isrInitNum`	Array index into IsrInitFuncs for initializing this ISR
`int isrNum`	Array index into IsrFuncs for attaching the interrupt vector to
`void * buff`	Pointer to a buffer that IsrInitFuncs[isrInitNum] will receive

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Status
`MANGOERROR_ERR_INVALID_PARAMETER`	'handle' is invalid
`Other value`	Error from OS

Description

Directs the MangoWDM driver to use an ISR that was compiled into the MangoWDM1 driver

Remarks

None

Example

```
int errorCode;
int isrInitNum = 0;
int isrNum = 0;
HANDLE config;

config = CreateEvent(
 NULL,
 FALSE,
 FALSE,
 NULL
 );
errorCode = MANGOBIOS_isrConnect(
 &device_handle,
 isrInitNum,
 isrNum,
 &config (this will allow the isr to reference this object and set this event on each
interrupt)
 );
```

3.2.14 MANGOBIOS_isrDisconnect

```
MANGOERROR_error_t MANGOBIOS_isrDisconnect(MANGOBIOS_deviceHandle_t * handle);
```

Summary

Disconnects an ISR from a MANGOBIOS_deviceHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_deviceHandle_t * handle`	handle to device
`isrShutdownNum`	Array index into IsrShutdownFuncs for erasing everything done with the previously used IsrInitFuncs[isrNum] (as performed in MANGOBIOS_isrConnect (see page 21))

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`MANGOERROR_ERR_INVALID_PARAMETER`	'handle' is invalid
`Other value`	Error from OS

Description

Directs the MangoWDM driver to disconnect the ISR that was connected to 'handle'

Remarks

None

Example

```
int errorCode;
int isrShutdownNum = 0;

errorCode = MANGOBIOS_isrConnect(
 &device_handle,
 isrShutdownNum
 );
```

3.2.15 MANGOBIOS_memoryAlloc

```
MANGOERROR_error_t MANGOBIOS_memoryAlloc(int size, MANGOBIOS_memoryHandle_t * handle,
unsigned int * physicalAdr, const MANGOBIOS_memoryAllocAttrs_t * attrs);
```

Summary

Allocates physical memory to MANGOBIOS_memoryHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`int size`	Length in bytes of requested memory buffer
`MANGOBIOS_memoryHandle_t * handle`	handle for memory buffer
`unsigned int * physicalAdr`	Pointer for physical address of memory buffer
`const MANGOBIOS_memoryAllocAttrs_t * attrs`	NULL

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`Other value`	Error from OS

Description

Allocates 'size' bytes of physical memory using the MangoMem driver

Remarks

Actual size of buffer is based on granularity in the MangoMem driver, therefore will most likely be up to 4Kbytes larger than requested.

Example

```
int errorCode;
int size = 32768;
MANGOBIOS_memoryHandle_t handle;
unsigned int physicalAdr;

errorCode = MANGOBIOS_memoryAlloc(
 size,
 &handle,
 &physicalAdr,
 NULL
 );
```

3.2.16 MANGOBIOS_memoryFree

```
MANGOERROR_error_t MANGOBIOS_memoryFree(MANGOBIOS_memoryHandle_t handle);
```

Summary

Frees a MANGOBIOS_memoryHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_memoryHandle_t handle`	Handle to memory buffer

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`Other value`	Error from OS

Description

Frees a non-mapped memory buffer which was previously allocated with MANGOBIOS_memoryAlloc (see page 23)

Remarks

Will return an error if the memory is mapped. Call MANGOBIOS_memoryUnmap (see page 26) on a memory buffer to unmap it.

Example

```
int errorCode;

errorCode = MANGOBIOS_memoryFree(
 memory_handle
 );
```

3.2.17 MANGOBIOS_memoryMap

```
MANGOERROR_error_t MANGOBIOS_memoryMap(MANGOBIOS_memoryHandle_t handle, void ** virtualAdr,
const MANGOBIOS_memoryMapAttrs_t * attrs);
```

Summary

Maps physical memory of a MANGOBIOS_memoryHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_memoryHandle_t handle`	handle of memory buffer
`void ** virtualAdr`	Pointer for virtual address of memory buffer
`const MANGOBIOS_memoryMapAttrs_t * attrs`	Set of attributes used when mapping the memory

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`Other value`	Error from OS

Description

Maps memory associated with 'handle' to virtual memory

Remarks

Actual size of buffer is based on granularity in the MangoMem driver, therefore will most likely be up to 4Kbytes larger than requested.

Example

```
int errorCode;
int size = 32768;
MANGOBIOS_memoryHandle_t handle;
unsigned int physicalAdr;
char * buffer;
MANGOBIOS_memoryMapAttrs_t attrs;

attrs.cacheEnable = 1;
errorCode = MANGOBIOS_memoryAlloc(
 size,
 &handle,
 &physicalAdr,
 NULL
 );
errorCode = MANGOBIOS_memoryMap(
 handle,
 &buffer,
 attrs
 );
memset(buffer, 0, size);
```

3.2.18 MANGOBIOS_memoryUnmap

```
MANGOERROR_error_t MANGOBIOS_memoryUnmap(MANGOBIOS_memoryHandle_t handle);
```

Summary

Unmaps a MANGOBIOS_memoryHandle_t

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`MANGOBIOS_memoryHandle_t handle`	Handle to mapped memory buffer

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_SUCCESS`	Success
`Other value`	Error from OS

Description

Unmaps a mapped memory buffer which was previously allocated with MANGOBIOS_memoryAlloc (see page 23) and mapped with MANGOBIOS_memoryMap (see page 25)

Remarks

Will fail if the memory was not successfully mapped using MANGOBIOS_memoryMap (see page 25).

Example

```
int errorCode;

errorCode = MANGOBIOS_memoryUnmap(
 memory_handle
 );
```

3.2.19 MANGOBIOS_open

```
MANGOERROR_error_t MANGOBIOS_open(const MANGOBIOS_attrs_t * attrs);
```

Summary

Initializes MangoBios Library for this process

File

MangoBios.h (see page 35)

Parameters

Parameters	Description
`const MANGOBIOS_attrs_t * attrs`	NULL

Returns

Status

Return Values

Return Values	Description
`MANGOERROR_INVALID_CONFIGURATION`	Failed LoadLibrary for SetupApi.dll Failed GetProcAddress for SetupDiEnumDeviceInterfaces or SetupDiGetDeviceInterfaceDetailA or SetupDiDestroyDeviceInfoList or SetupDiGetClassDevsA No MangoMem device exists
`MANGOERROR_ERR_INVALID_HANDLE`	Failed SetupDiGetClassDevs_p for MangoWdm driver or MangoMem driver. Failed CreateFile on MangoMem
`MANGOERROR_INSUFFICIENT_RESOURCES`	Failed malloc
`Other value`	Error from OS

Description

Opens the MangoWdm driver and MangoMem driver.

Remarks

There must be one MangoMem instance in the system, and the MangoWdm (pci device driver) must be known to the system for MANGOBIOS_open to pass.

Example

```
int errorCode;
errorCode = MANGOBIOS_open(
 NULL
 );
```

3.3 Types

Types

Type	Description
MANGOERROR_error_t (⧉ see page 29)	typedef of structure MANGOERROR_error_e (⧉ see page 6)

3.3.1 MANGOERROR_error_t

```
typedef enum MANGOERROR_error_e MANGOERROR_error_t;
```

File

MangoError.h (see page 36)

Description

typedef of structure MANGOERROR_error_e (see page 6)

3.4 Macros

Macros

Macro	Description
IS_BIG_ENDIAN (↗ see page 31)	defined(_BIG_ENDIAN) defined(__vxworks) -> defined(linux) || defined(__SVR4)
MANGOBIOS_version_build (↗ see page 32)	builds versioning information
POSIX_COMPLIANT (↗ see page 33)	defined(linux) || defined(__SVR4)

3.4.1 IS_BIG_ENDIAN

```
#define IS_BIG_ENDIAN
```

File

MangoBios.h (see page 35)

Description

defined(_BIG_ENDIAN) defined(__vxworks) -> defined(linux) || defined(__SVR4)

3.4.2 MANGOBIOS_version_build

```
#define MANGOBIOS_version_build(major, minor) ((major << 16) | minor)
```

File

MangoBios.h (see page 35)

Description

builds versioning information

3.4.3 POSIX_COMPLIANT

```
#define POSIX_COMPLIANT
```

File

MangoBios.h (see page 35)

Description

defined(linux) || defined(__SVR4)

3.5 Files

Files

File	Description
MangoBios.h (⧉ see page 35)	MangoBios header file
MangoError.h (⧉ see page 36)	MangoBios header file

3.5.1 MangoBios.h

MangoBios header file

Description

MangoBios API declarations

History

Author	Change Description
Nachum Kanovsky	Created

Functions

Function	Description
MANGOBIOS_close (⧉ see page 8)	Closes MangoBios library for this process
MANGOBIOS_deviceClose (⧉ see page 9)	Closes MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceGetProperty (⧉ see page 10)	Gets a MANGOBIOS_deviceProp_t
MANGOBIOS_deviceOpen (⧉ see page 11)	Opens MANGOBIOS_deviceHandle_t
MANGOBIOS_devicePciRegRead (⧉ see page 12)	Reads a pci register from a MANGOBIOS_deviceHandle_t
MANGOBIOS_devicePciRegWrite (⧉ see page 13)	Writes a pci register to a MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceRead (⧉ see page 14)	Reads from a MANGOBIOS_deviceHandle_t
MANGOBIOS_deviceSetProperty (⧉ see page 15)	Sets a MANGOBIOS_deviceProp_t
MANGOBIOS_deviceWrite (⧉ see page 16)	Writes to a MANGOBIOS_deviceHandle_t
MANGOBIOS_getDeviceHandles (⧉ see page 17)	Fills a MANGOBIOS_deviceHandle_t array
MANGOBIOS_getNumDevices (⧉ see page 19)	Gets number of devices of MANGOBIOS_deviceType_t type
MANGOBIOS_getVersion (⧉ see page 20)	Gets MANGOBIOS_version_t
MANGOBIOS_isrConnect (⧉ see page 21)	Connects an ISR to a MANGOBIOS_deviceHandle_t
MANGOBIOS_isrDisconnect (⧉ see page 22)	Disconnects an ISR from a MANGOBIOS_deviceHandle_t
MANGOBIOS_memoryAlloc (⧉ see page 23)	Allocates physical memory to MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryFree (⧉ see page 24)	Frees a MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryMap (⧉ see page 25)	Maps physical memory of a MANGOBIOS_memoryHandle_t
MANGOBIOS_memoryUnmap (⧉ see page 26)	Unmaps a MANGOBIOS_memoryHandle_t
MANGOBIOS_open (⧉ see page 27)	Initializes MangoBios Library for this process

Macros

Macro	Description
IS_BIG_ENDIAN (⧉ see page 31)	defined(_BIG_ENDIAN) defined(__vxworks) -> defined(linux) || defined(__SVR4)
MANGOBIOS_version_build (⧉ see page 32)	builds versioning information
POSIX_COMPLIANT (⧉ see page 33)	defined(linux) || defined(__SVR4)

3.5.2 MangoError.h

MangoBios header file

Description

Error codes for all MangoBios based libraries and functions

History

Author	Change Description
Nachum Kanovsky	Created

Enumerations

Enumeration	Description
MANGOERROR_error_e (see page 6)	enum of possible errors returned in MangoBios based projects

Types

Type	Description
MANGOERROR_error_t (see page 29)	typedef of structure MANGOERROR_error_e (see page 6)

Index