MangoAV_HW Lib

Table of Contents

MangoAV_HW Lib

1 Introduction

Versions

This manual applies to the following library versions:

MangoAV_HW_DM v1.15

MangoAV_HW_C6412 v1.19

MangoAV_HW_C6415 v1.6

Introduction

The MangoAV_HW library is intended to simplify creation of multimedia software based on Mango hardware. It is intended for the DSP software developer. Our goal in this library is to offer a consistent API that hides the details necessary for audio and video I/O, as well as taking care of most of the differences between the various Mango hardware platforms.

Supported Hardware

Current hardware supported: Seagull PMC, Lark (MangoAV_HW_C6415 lib); Seagull PC-104 Plus (Phoenix), Raven-X, HX, PH (MangoAV_HW_DM lib); Raven-D, V, C(MangoAV_HW_C6412 lib).

Differences between cards

Each card has a slightly different initialization process and procedure for working with its peripherals. For information pertaining to your particular card, please refer to the MangoAV_HW User's Manual and contact us if you have any questions.

In Development

This library is in development, and not all features are available for all hardware platforms.

Open Issues

- Seagull PMC: No audio.

2 Symbol Reference

2.1 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

Enumeration	Description
Baud_Rate_T (see page 2)	This is type Baud_Rate_T.
Bit_Count_T (see page 2)	This is type Bit_Count_T.
I2C_Command_T (see page 3)	Commands for MangoAV_HW_C64_i2c_load (see page 6)
Image_Formats_T (see page 3)	Image formats for MangoAV_HW_get_img_from_cam (see page 11)
Parity_T (see page 3)	This is type Parity_T.
Stop_Bit_t (see page 4)	This is type Stop_Bit_t.

Structures

Structure	Description
TimeOfDay_T (see page 4)	This is type TimeOfDay_T.

2.1.1 Baud_Rate_T Enumeration

```
typedef enum {
  BAUD_300 = 384,
  BAUD_600 = 192,
  BAUD_1200 = 96,
  BAUD_2400 = 48,
  BAUD_4800 = 24,
  BAUD_9600 = 12,
  BAUD_19200 = 6,
  BAUD_38400 = 3,
  BAUD_56600 = 2,
  BAUD_115200 = 1
} Baud_Rate_T;
```

File

MangoAV_HWExp.h (see page 26)

Description

This is type Baud_Rate_T.

2.1.2 Bit_Count_T Enumeration

```
typedef enum {
  BIT_COUNT_5 = 0,
  BIT_COUNT_6 = 1,
  BIT_COUNT_7 = 2,
  BIT_COUNT_8 = 3
} Bit_Count_T;
```

File

MangoAV_HWExp.h (see page 26)

Description

This is type Bit_Count_T.

2.1.3 I2C_Command_T Enumeration

```
typedef enum {
  LOAD_DEFAULT_A2D,
  LOAD_DEFAULT_D2A,
  SET_NTSC_A2D,
  SET_PAL_A2D,
  SET_NTSC_D2A,
  SET_PAL_D2A,
  SET_NTSC_VGA_A2D
} I2C_Command_T;
```

File

MangoAV_HWExp.h (see page 26)

Description

Commands for MangoAV_HW_C64_i2c_load (see page 6)

2.1.4 Image_Formats_T Enumeration

```
typedef enum {
  MANGOAV_HW_420PLANAR,
  MANGOAV_HW_422PLANAR,
  MANGOAV_HW_422RAW
} Image_Formats_T;
```

File

MangoAV_HWExp.h (see page 26)

Description

Image formats for MangoAV_HW_get_img_from_cam (see page 11)

2.1.5 Parity_T Enumeration

```
typedef enum {
  PARITY_NONE,
  PARITY_ODD,
  PARITY_EVEN
} Parity_T;
```

File

MangoAV_HWExp.h (see page 26)

Description

This is type Parity_T.

2.1.6 Stop_Bit_t Enumeration

```
typedef enum {
  STOP_BIT_1 = 0,
  STOP_BIT_15 = 1,
  STOP_BIT_2 = 1
} Stop_Bit_t;
```

File

MangoAV_HWExp.h (see page 26)

Description

This is type Stop_Bit_t.

2.1.7 TimeOfDay_T Structure

```
typedef struct {
  Uint8 Year;
  Uint8 Month;
  Uint8 Day;
  Uint8 DayOfWeek;
  Uint8 Hour;
  Uint8 Minute;
  Uint8 Second;
} TimeOfDay_T;
```

File

MangoAV_HWExp.h (see page 26)

Members

Members	Description
`Uint8 Year;`	Last two digits of the year, 2000-2099.
`Uint8 Month;`	1-12
`Uint8 Day;`	1-31
`Uint8 DayOfWeek;`	0-6 corresponds to Sun-Sat
`Uint8 Hour;`	0-23 (24-hour format)
`Uint8 Minute;`	0-59
`Uint8 Second;`	0-59

Description

This is type TimeOfDay_T.

2.2 Functions

The following table lists functions in this documentation.

Functions

Function	Description
MangoAV_HW_C64_get_gpStreams (see page 5)	Returns SIO stream handles for inter-DSP general-purpose communications.
MangoAV_HW_C64_i2c_load (see page 6)	Setup video chips (C64 based devices only)
MangoAV_HW_close_cam_input (see page 6)	Closes video input and frees resources.
MangoAV_HW_close_mic_input (see page 7)	Closes audio channel and frees resources.

MangoAV_HW_close_spk_output (see page 7)	Closes output audio channel and frees resources.
MangoAV_HW_close_tv_output (see page 7)	Closes TV output and frees resources.
MangoAV_HW_Close_UART (see page 8)	Closes a UART port
MangoAV_HW_Config_UART (see page 8)	Initializes and configures a UART port
MangoAV_HW_E2PROM_Read (see page 9)	Reads data from E2PROM.
MangoAV_HW_E2PROM_Write (see page 9)	Writes data to E2PROM.
MangoAV_HW_get_audio_frame_from_mic (see page 10)	Gets audio data from audio input channel.
MangoAV_HW_get_cam_input_param (see page 10)	Get video input parameter
MangoAV_HW_get_img_from_cam (see page 11)	Get image from video in
MangoAV_HW_get_ttl_in (see page 12)	Read status of TTL inputs (on boards supporting TTL).
MangoAV_HW_get_ttl_out (see page 12)	Get current status of TTL outputs (on boards supporting TTL).
MangoAV_HW_Get_Version (see page 12)	Returns lib version.
MangoAV_HW_get_vid_status (see page 13)	Check video input status
MangoAV_HW_init (see page 13)	Initialize hardware and library
MangoAV_HW_init_main (see page 14)	Perform further initializations that can only be called from main().
MangoAV_HW_open_cam_input (see page 14)	Opens camera (video in) stream.
MangoAV_HW_open_mic_input (see page 15)	Opens audio input channel
MangoAV_HW_open_spk_output (see page 16)	Opens audio output channel
MangoAV_HW_open_tv_output (see page 16)	Open analog monitor (TV) output
MangoAV_HW_query_cam (see page 17)	Test whether image is ready to be received from video input.
MangoAV_HW_query_monitor (see page 17)	Check whether monitor is ready for output
MangoAV_HW_Read_UART (see page 17)	Reads a requested byte count from the UART input buffer
MangoAV_HW_resetBoard (see page 18)	Resets the board, equivalent to pressing the board's Reset button. Flash-based boards will reboot.
MangoAV_HW_RTC_Get_Status (see page 18)	Reads status from Real Time Clock chip.
MangoAV_HW_RTC_Get_Time (see page 19)	Reads time and date from Real Time Clock chip.
MangoAV_HW_RTC_Reset (see page 19)	Resets the Real Time Clock chip.
MangoAV_HW_RTC_Set_Time (see page 20)	Sets time and date on Real Time Clock chip.
MangoAV_HW_send_audio_frame_to_spk (see page 20)	Sends audio data to audio output channel.
MangoAV_HW_send_img_to_monitor (see page 20)	Sends image to monitor (TV) output
MangoAV_HW_set_cam_input_param (see page 21)	Set video input parameter
MangoAV_HW_set_led (see page 22)	Sets user-definable LEDs
MangoAV_HW_set_led_control (see page 22)	Determines which DSP controls each LED.
MangoAV_HW_set_ttl_out (see page 22)	Set status of TTL outputs (on boards supporting TTL).
MangoAV_HW_set_video_loopback (see page 23)	Enables hardware video loopback on video input chip. The video input port number to use is the same as in MangoAV_HW_open_cam_input (see page 14). The loopback output will echo the selected input.
MangoAV_HW_Write_UART (see page 23)	Writes data to the UART.
qdma_memcpy (see page 24)	Auxiliary QDMA memory copy function.
qdma_memcpy_array (see page 24)	Auxiliary QDMA 2D memory copy function.
qdma_memcpy_wait (see page 25)	Waits (spins) until all pending qdma_memcpy (see page 24) requests complete.

Legend

	Method

2.2.1 MangoAV_HW_C64_get_gpStreams Function

Returns SIO stream handles for inter-DSP general-purpose communications.

C++

```
MANGOERROR_error_t MangoAV_HW_C64_get_gpStreams(SIO_Handle* sio);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`SIO_Handle* sio`	SIO_Handle array of two entries. After returning, first entry will contain input stream and second entry will contain output stream.

Returns

MANGOERROR_SUCCESS - ok.

Remarks

This function is supported only on the Raven-D and provides support for a networked application running on DSP 0 to communicate with DSP 1 (which does not have its own network node). A buffer issued to the output stream on one DSP will be received on the input stream of the other DSP. These streams are created by the library. The creation parameters (see help for SIO_create) are all left as default except for the usage model, which is set to ISSUE_RECLAIM. The streams should therefore be used with the commands SIO_issue and SIO_reclaim. Please refer to TI's documentation for help regarding these commands.

Important: The size of the buffer sent from one DSP **must** be the same as the size issued by the other DSP (the receiver) to its incoming stream. The stream will not operate properly otherwise. It is recommended to set a certain fixed size, and always issue and reclaim buffers of that size only; perform multiple transfers when you wish to send a larger amount of data.

2.2.2 MangoAV_HW_C64_i2c_load Function

Setup video chips (C64 based devices only)

C++

```
MANGOERROR_error_t MangoAV_HW_C64_i2c_load(I2C_Command_T cmd, int chip_id);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`I2C_Command_T cmd`	Initialization command, see I2C_Command_T (see page 3)
`int chip_id`	Number of chip to load, or -1 to configure all possible chips relevant to this command.

Returns

MANGOERROR_INVALID_PARAMETERS - Some parameter was invalid.

MANGOERROR_SUCCESS - ok.

Description

Accesses registers on Philips video encoder/decoder chips and initializes them according to a predefined batch of values.

2.2.3 MangoAV_HW_close_cam_input Function

Closes video input and frees resources.

C++

```
MANGOERROR_error_t MangoAV_HW_close_cam_input(int cam_num);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int cam_num`	Video input port number

Returns

MANGOERROR_INSUFFICIENT_RESOURCES - I/O error

MANGOERROR_SUCCESS - ok

2.2.4 MangoAV_HW_close_mic_input Function

Closes audio channel and frees resources.

C++

```
MANGOERROR_error_t MangoAV_HW_close_mic_input(int audio_channel);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int audio_channel`	Audio channel to close

Returns

MANGOERROR_SUCCESS - ok

2.2.5 MangoAV_HW_close_spk_output Function

Closes output audio channel and frees resources.

C++

```
MANGOERROR_error_t MangoAV_HW_close_spk_output(int audio_channel);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int audio_channel`	Audio channel to close

Returns

MANGOERROR_SUCCESS - ok

2.2.6 MangoAV_HW_close_tv_output Function

Closes TV output and frees resources.

C++

```
MANGOERROR_error_t MangoAV_HW_close_tv_output();
```

File

MangoAV_HWExp.h (see page 26)

Returns

MANGOERROR_SUCCESS - ok

2.2.7 MangoAV_HW_Close_UART Function

Closes a UART port

C++

```
MANGOERROR_error_t MangoAV_HW_Close_UART(int iUartIndex);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int iUartIndex`	Physical UART index (0/1)

Returns

MANGOERROR_SUCCESS - ok.

MANGOERROR_INVALID_PARAMETERS - UART index invalid or was never opened.

MANGOERROR_ERR_NOT_IMPLEMENTED -this board does not support UART.

2.2.8 MangoAV_HW_Config_UART Function

Initializes and configures a UART port

C++

```
MANGOERROR_error_t MangoAV_HW_Config_UART(int iUartIndex, Baud_Rate_T BaudRate, Bit_Count_T
BitCount, Stop_Bit_t StopBit, Parity_T Parity, int InBuffSize);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int iUartIndex`	Physical UART index (0/1)
`Baud_Rate_T BaudRate`	Baud rate
`Bit_Count_T BitCount`	Data bits
`Stop_Bit_t StopBit`	Stop bit count
`Parity_T Parity`	Parity
`int InBuffSize`	Input buffer size

Returns

MANGOERROR_SUCCESS - ok.

MANGOERROR_INVALID_PARAMETERS - UART index, baud rate or data bit count invalid.

MANGOERROR_ERR_NOT_IMPLEMENTED -this board does not support UART.

MANGOERROR_INSUFFICIENT_RESOURCES - out of memory during initialization.

Remarks

It is safe to call this function repeatedly in order to change parameters. There is no need to call MangoAV_HW_Close_UART (see page 8) between calls.

2.2.9 MangoAV_HW_E2PROM_Read Function

Reads data from E2PROM.

C++

```
MANGOERROR_error_t MangoAV_HW_E2PROM_Read(Uint16 Address, Uint8 * data, Uint16 numBytes);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`Uint16 Address`	E2PROM address.
`Uint8 * data`	Data buffer.
`Uint16 numBytes`	Amount (in bytes) to read.

Returns

MANGOERROR_SUCCESS - ok.

MANGOERROR_FAILURE - I2C bus not properly initialized. MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support E2PROM.

Remarks

Raven-HX104, Raven-PH and Raven-D Mark III only.

2.2.10 MangoAV_HW_E2PROM_Write Function

Writes data to E2PROM.

C++

```
MANGOERROR_error_t MangoAV_HW_E2PROM_Write(Uint16 Address, Uint8 * data, Uint16 numBytes);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`Uint16 Address`	E2PROM address.
`Uint8 * data`	Data buffer.
`Uint16 numBytes`	Amount (in bytes) to write.

Returns

MANGOERROR_SUCCESS - ok.

MANGOERROR_FAILURE - I2C bus not properly initialized. MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support E2PROM.

Remarks

Raven-HX104, Raven-PH and Raven-D Mark III only.

2.2.11 MangoAV_HW_get_audio_frame_from_mic Function

Gets audio data from audio input channel.

C++

```
MANGOERROR_error_t MangoAV_HW_get_audio_frame_from_mic(short* in_data, int audio_channel,
int * size, Time_tag_T* time_tag);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
short* in_data	Pointer for output data.
int audio_channel	Audio channel to read.
int * size	Size of incoming data buffer.
%PAR4%	Time tag representing time frame was received.

Returns

MANGOERROR_SUCCESS - ok.

Remarks

Audio channel must have been opened with MangoAV_HW_open_mic_input (see page 15).

The audio data buffer must be aligned on a cache line size boundary, and must be a multiple of the cache line size. This value is given by the constant CACHE_L2_LINESIZE, and is equal to 0x80 on C64x processors.

The Time Tag is a 64-bit value separated into two unsigned ints (t_msb and t_lsb for most and least significant 32 bits, respectively). Its resolution is identical to that of CLK_gethtime(). The msb is obtained by incrementing it whenever the lsb has wrapped around.

2.2.12 MangoAV_HW_get_cam_input_param Function

Get video input parameter

C++

```
MANGOERROR_error_t MangoAV_HW_get_cam_input_param(const int cam_num, const Attrib_type_T
attr, unsigned char * val);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
const int cam_num	Video input port number
const Attrib_type_T attr	Attribute to set. May be one of: ATTRIB_BRIGHTNESS (0x00 - 0xff, 0x80 default) ATTRIB_CONTRAST (0x00 - 0x7f, 0x44 default) ATTRIB_COLOR (0x00 - 0x7f, 0x40 default)
unsigned char * val	Value to get.

Returns

MANGOERROR_INVALID_PARAMETERS - If any parameter was invalid.

MANGOERROR_SUCCESS - ok.

Description

Get brightness, contrast or color by accessing appropriate register in video decoder chip.

2.2.13 MangoAV_HW_get_img_from_cam Function

Get image from video in

C++

```
MANGOERROR_error_t MangoAV_HW_get_img_from_cam(unsigned char* yuv_buf, Image_Sizes_T
img_size, int cam_num, Image_Formats_T img_format, Time_tag_T* time_tag, void*
isram_scratch_buffer, SEM_Handle isram_buffer_protect);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`unsigned char* yuv_buf`	Buffer to place new image
`Image_Sizes_T img_size`	Requested size of image
`int cam_num`	Video input port number
`Image_Formats_T img_format`	Image output format.
`Time_tag_T* time_tag`	Time tag representing time frame was received.
`void* isram_scratch_buffer`	Internal memory scratch buffer (C64 only).
`SEM_Handle isram_buffer_protect`	Optional protection semaphore for scratch buffer (C64 only).

Returns

Return values depend on timeout value given to MangoAV_HW_open_cam_input (see page 14).

The following codes can only be returned if timeout is **not** MANGOBIOS_FOREVER:

MANGOERROR_TIMEOUT - Timeout occurred, no image received in the time allotted

MANGOERROR_FAILURE - Corrupt image received, perhaps due to bad connection or if video was disconnected and reconnected. User may close and reopen stream as a remedy.

These return codes may always be returned:

MANGOERROR_INSUFFICIENT_RESOURCES - Fatal I/O error

MANGOERROR_INVALID_PARAMETERS - Some parameter was incorrect. Note that not all image formats are supported on all platforms.

MANGOERROR_SUCCESS - Image received ok.

Description

Receives an image, guaranteed to be correct and usable, from analog video input, in 4:2:0 or 4:2:2 Y/Cr/Cb planar format of requested size. Raw format is also supported on some platforms (in this mode, no conversion takes place and only a 32-bit pointer to the raw buffer is placed in yuv_buf).

Remarks

The video input must have previously been enabled with MangoAV_HW_open_cam_input (see page 14).

The buffer for the incoming image must be aligned on a cache line size boundary, and must be a multiple of the cache line size. This value is given by the constant CACHE_L2_LINESIZE, and is equal to 0x80 on C64x processors.

The scratch buffer must be allocated in on-chip memory. The function rounds up the given pointer to an 8-byte alignment and uses exactly 16,000 (0x3E80) bytes. Set to NULL on DM642 based systems.

The protection semaphore, if supplied (not NULL), will be pended on before accessing scratch memory. It will be posted

upon exit from the function.

"Raw" mode is only supported in the C64 based libraries. Note that in this mode, on the Seagull PMC and Lark cards, the received buffer is only guaranteed to be valid for a short period of time, after which it will be overwritten by new data. The validity period may be up to 80ms, but as low as a few ms if processing has been lagging. On other boards, the buffer is guaranteed to be untouched until the next call to this function.

The Time Tag is a 64-bit value separated into two unsigned ints (t_msb and t_lsb for most and least significant 32 bits, respectively). Its resolution is identical to that of CLK_gethtime(). The msb is obtained by incrementing it whenever the lsb has wrapped around.

2.2.14 MangoAV_HW_get_ttl_in Function

Read status of TTL inputs (on boards supporting TTL).

C++

```
unsigned char MangoAV_HW_get_ttl_in();
```

File

MangoAV_HWExp.h (see page 26)

Returns

8-bit value, where each bit represents the status of the corresponding TTL-in channels.

2.2.15 MangoAV_HW_get_ttl_out Function

Get current status of TTL outputs (on boards supporting TTL).

C++

```
unsigned char MangoAV_HW_get_ttl_out();
```

File

MangoAV_HWExp.h (see page 26)

Returns

Each bit represents the current state of the corresponding output TTL.

2.2.16 MangoAV_HW_Get_Version Function

Returns lib version.

C++

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

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
MANGOBIOS_version_t * version	Pointer for returned version number.

Returns

- MANGOERROR_SUCCESS - successful.

Remarks

Version is returned as an unsigned int containing major and minor version numbers in this format: ((major << 16) | minor)

2.2.17 MangoAV_HW_get_vid_status Function

Check video input status

C++

```
MANGOERROR_error_t MangoAV_HW_get_vid_status(int chip_id);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int chip_id`	Chip number to check.

Returns

MANGOERROR_SUCCESS - Video is being received.

MANGOERROR_FAILURE - Video is not received.

MANGOERROR_RESOURCE_UNAVAILABLE - Hardware error when checking status, or invalid request for this hardware.

Description

Accesses status register on Philips video decoder chips and checks whether video is being received.

Remarks

For C64 based cards: Chip must have previously been configured using MangoAV_HW_C64_i2c_load (see page 6).

For DM642 based cards: Video input must be open (with MangoAV_HW_open_cam_input (see page 14)) when using this function. Also note that there are two camera inputs connected to each chip, so:

- (1) this will only return the status of the currently opened camera (behavior is undefined if both inputs are opened at once);
- (2) the parameter for this function will be the camera number divided by 2.

2.2.18 MangoAV_HW_init Function

Initialize hardware and library

C++

```
MANGOERROR_error_t MangoAV_HW_init(MANGOCARD_card_t card_type, int dsp_num, GPIO_Handle
gpio_handle, void* card_handle, int* seg_isram, int* seg_sdram);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`MANGOCARD_card_t card_type`	The hardware platform you are using, see MangoShared for a list of possible values.
`int dsp_num`	Number of DSP you are using. Not necessary on most platforms, 0 may be used.
`GPIO_Handle gpio_handle`	GPIO handle to be used by the library, obtained either by <card_name>_Open() (platform dependent) or by opening it yourself with a call such as: GPIO_open(GPIO_DEV0, GPIO_OPEN_RESET);
`void* card_handle`	A card handle suitable for your card, obtained using <card_name>_Open(). On the Raven-X and Half-X use NULL here.
`int* seg_isram`	A pointer to an external integer containing an internal- memory segment identifier for use with MEM_alloc. The contents of this integer must not necessarily be valid at the time of this call (this applies for Seagull PMC, Lark). See the user manual for details.
`int* seg_sdram`	As the previous parameter, but this must be an external memory heap.

Returns

MANGOERROR_INVALID_PARAMETERS - if any parameter is invalid.

MANGOERROR_SUCCESS - success.

Description

Initializes various system parameters, including EMIF registers. This function must be called as early as possible. The recommended method is to have a function named SYSTEM_Init, set in the CDB in System/Global Settings/User Init Function, or preferably as the init function for the relevant device (such as FPGA stream, if applicable) under Input/Output/Device Drivers/User-Defined Devices.

In SYSTEM_Init, first call <card_name>_Open(), and then this function. Please refer to example code specific to your hardware for more information.

2.2.19 MangoAV_HW_init_main Function

Perform further initializations that can only be called from main().

C++

```
MANGOERROR_error_t MangoAV_HW_init_main();
```

File

MangoAV_HWExp.h (see page 26)

Returns

MANGOERROR_FAILURE - memory allocation failure. MANGOERROR_SUCCESS - success.

Description

This function performs certain startup procedures that cannot be performed before main(). It must be called from your main() function.

2.2.20 MangoAV_HW_open_cam_input Function

Opens camera (video in) stream.

C++

```
MANGOERROR_error_t MangoAV_HW_open_cam_input(Video_Standard_T std,
Video_Analog_Connection_Format_T conn, int cam_num, int num_bufs, int timeout);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`Video_Standard_T std`	Video standard of source (PAL/NTSC)
`Video_Analog_Connection_Format_T conn`	Connection type, Composite or S-Video (not relevant for all platforms).
`int cam_num`	Video input port number
`int num_bufs`	Number of input buffers to allocate.
`int timeout`	Timeout for getting images: MANGOBIOS_FOREVER - Function will not return until a valid image is received. MANGOBIOS_POLL - Function will always return immediately. other n > 0 - Function will timeout after 'n' PRD ticks (usually milliseconds).

Returns

MANGOERROR_INSUFFICIENT_RESOURCES - Allocation or I/O error.

MANGOERROR_SUCCESS - ok.

Remarks

Here are the allowed values for the num_bufs parameter:

- 6412: A value of 1 is allowed but will likely cause frame loss. The highest value is only limited by the amount of available SDRAM.
- DM: Values allowed are 2 to 10 (again, limited by SDRAM).
- 6415: This value is currently ignored and is fixed at 2.

2.2.21 MangoAV_HW_open_mic_input Function

Opens audio input channel

C++

```
MANGOERROR_error_t MangoAV_HW_open_mic_input(int audio_channel, int is_stereo, int
sample_rate_divider, Audio_Connector_T input_type);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int audio_channel`	Audio channel to open
`int is_stereo`	0 for Mono, 1 for Stereo. Sample rate divider
`int sample_rate_divider`	Sets input gain for Line-in or Microphone.

Returns

MANGOERROR_SUCCESS - ok

Remarks

For the sample rate divider parameter, divide 96000 by the desired sample rate. For example, if you want 48000 Hz, input 2.

Input Gain parameter is effective on Raven-D Mark III only.

2.2.22 MangoAV_HW_open_spk_output Function

Opens audio output channel

C++

```
MANGOERROR_error_t MangoAV_HW_open_spk_output(int audio_channel, int is_stereo, int
sample_rate_divider);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int audio_channel`	Audio channel to open
`int is_stereo`	0 for Mono, 1 for Stereo.
`int sample_rate_divider`	Sample rate divider

Returns

MANGOERROR_SUCCESS - ok

Remarks

For the sample rate divider parameter, divide 96000 by the desired sample rate. For example, if you want 48000 Hz, input 2.

2.2.23 MangoAV_HW_open_tv_output Function

Open analog monitor (TV) output

C++

```
MANGOERROR_error_t MangoAV_HW_open_tv_output(Video_Standard_T std, int num_bufs, int
timeout);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`Video_Standard_T std`	TV standard, PAL/NTSC
`int num_bufs`	Number of output buffers to allocate.
`int timeout`	Timeout to use when sending frame.

Returns

MANGOERROR_INSUFFICIENT_RESOURCES - Allocation or I/O error.

MANGOERROR_SUCCESS - ok

Description

Performs initializations for outputting an analog video stream.

Remarks

Here are the allowed values for the num_bufs parameter:

- 6412: The lowest allowed value is 2. The highest value is only limited by the amount of available SDRAM.
- DM: Values allowed are 2 to 10 (again, limited by SDRAM).

- 6415: Both timeout and num_bufs are ignored. There is only one buffer and regardless of the timeout value MangoAV_HW_send_img_to_monitor (see page 20) will wait (calling TSK_yield) for the appropriate moment to update it in such a way that the transition will not be visible on the TV screen.

2.2.24 MangoAV_HW_query_cam Function

Test whether image is ready to be received from video input.

C++

```
MANGOERROR_error_t MangoAV_HW_query_cam(int cam_num);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
int cam_num	Video input port number

Returns

MANGOERROR_INVALID_PARAMETERS - Camera not opened. MANGOERROR_TIMEOUT - No image ready in timeout period MANGOERROR_SUCCESS - New frame waiting.

Description

Checks whether there is a full buffer ready from the video input. A success return value guarantees that a subsequent call to MangoAV_HW_get_img_from_cam (see page 11) will not block or return a timeout, but immediately process the incoming image.

2.2.25 MangoAV_HW_query_monitor Function

Check whether monitor is ready for output

C++

```
MANGOERROR_error_t MangoAV_HW_query_monitor();
```

File

MangoAV_HWExp.h (see page 26)

Returns

MANGOERROR_FAILURE - Monitor output not open. MANGOERROR_TIMEOUT - No buffer ready (if timeout exists). MANGOERROR_SUCCESS - Buffer ready.

Description

Checks whether there is a free buffer ready for TV output. A success return value guarantees that a subsequent call to MangoAV_HW_send_img_to_monitor (see page 20) will not block or return a timeout, but immediately accept the image.

Remarks

Monitor output must have been opened with MangoAV_HW_open_tv_output (see page 16).

2.2.26 MangoAV_HW_Read_UART Function

Reads a requested byte count from the UART input buffer

C++

```
MANGOERROR_error_t MangoAV_HW_Read_UART(int iUartIndex, char * pBuff, int * iSize, unsigned int iTimeout);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int iUartIndex`	Physical UART index (0/1)
`char * pBuff`	Data buffer
`int * iSize`	Requested number of bytes. When function returns, contains number of bytes actually read.
`unsigned int iTimeout`	Request timeout [mSec]

Returns

MANGOERROR_SUCCESS - ok.

MANGOERROR_INVALID_PARAMETERS - UART index invalid.

MANGOERROR_ERR_NOT_IMPLEMENTED -this board does not support UART.

MANGOERROR_TIMEOUT - UART timed out before data was received. Check value of iSize for number of bytes read prior to timeout.

Remarks

The time-out is per byte.

2.2.27 MangoAV_HW_resetBoard Function

Resets the board, equivalent to pressing the board's Reset button. Flash-based boards will reboot.

C++

```
MANGOERROR_error_t MangoAV_HW_resetBoard();
```

File

MangoAV_HWExp.h (see page 26)

Returns

MANGOERROR_FAILURE - Tried to reset but failed. MANGOERROR_ERR_NOT_IMPLEMENTED - Reset not implemented for this board.

Remarks

In case of success, this function will not return.

2.2.28 MangoAV_HW_RTC_Get_Status Function

Reads status from Real Time Clock chip.

C++

```
MANGOERROR_error_t MangoAV_HW_RTC_Get_Status(Uint8* status);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`Uint8* status`	Pointer for return status byte.

Returns

MANGOERROR_SUCCESS - ok. MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support RTC.

Description

Reads status from Real Time Clock chip.

Remarks

Bit 7 of the status register indicates power loss. If this bit is set (1), it is an indication of failure of the backup battery. Reset the RTC chip using MangoAV_HW_RTC_Reset (see page 19) and, if applicable, notify the user of the failed battery.

2.2.29 MangoAV_HW_RTC_Get_Time Function

Reads time and date from Real Time Clock chip.

C++

```
MANGOERROR_error_t MangoAV_HW_RTC_Get_Time(TimeOfDay_T* time);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`TimeOfDay_T* time`	Pointer for return time data.

Returns

MANGOERROR_SUCCESS - ok. MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support RTC.

Description

Reads time and date from Real Time Clock chip.

2.2.30 MangoAV_HW_RTC_Reset Function

Resets the Real Time Clock chip.

C++

```
MANGOERROR_error_t MangoAV_HW_RTC_Reset();
```

File

MangoAV_HWExp.h (see page 26)

Returns

MANGOERROR_SUCCESS - ok. MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support RTC.

Description

Resets the Real Time Clock chip.

2.2.31 MangoAV_HW_RTC_Set_Time Function

Sets time and date on Real Time Clock chip.

C++

```
MANGOERROR_error_t MangoAV_HW_RTC_Set_Time(const TimeOfDay_T* time);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`const TimeOfDay_T* time`	Struct containint time data.

Returns

MANGOERROR_SUCCESS - ok. MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support RTC.

Description

Sets time and date on Real Time Clock chip.

2.2.32 MangoAV_HW_send_audio_frame_to_spk Function

Sends audio data to audio output channel.

C++

```
MANGOERROR_error_t MangoAV_HW_send_audio_frame_to_spk(short* out_data, int audio_channel,
int size);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`short* out_data`	Pointer for output data.
`int audio_channel`	Audio channel to send to.
`int size`	Size of data buffer in bytes.

Returns

MANGOERROR_SUCCESS - ok.

Remarks

Audio channel must have been opened with MangoAV_HW_open_spk_output (see page 16).

The audio data buffer must be aligned on a cache line size boundary, and must be a multiple of the cache line size. This value is given by the constant CACHE_L2_LINESIZE, and is equal to 0x80 on C64x processors.

2.2.33 MangoAV_HW_send_img_to_monitor Function

Sends image to monitor (TV) output

C++

```
MANGOERROR_error_t MangoAV_HW_send_img_to_monitor(unsigned char* img, Image_Sizes_T
img_size, void* isram_scratch_buffer, SEM_Handle isram_buffer_protect);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`unsigned char* img`	Pointer to image data.
`Image_Sizes_T img_size`	Size of image.
`void* isram_scratch_buffer`	Internal memory scratch buffer (C64 only).
`SEM_Handle isram_buffer_protect`	Optional protection semaphore for scratch buffer (C64 only, see MangoAV_HW_get_img_from_cam (see page 11) for information.)

Returns

MANGOERROR_INSUFFICIENT_RESOURCES - I/O error MANGOERROR_TIMEOUT - The output buffers are full (if timeout exists). MANGOERROR_SUCCESS - ok.

Description

Accepts 4:2:0 Y/Cr/Cb Planar image buffer of several supported sizes, converts it to proper output format and size for monitor output and displays image on the analog monitor.

Remarks

Monitor output must have been opened with MangoAV_HW_open_tv_output (see page 16).

The buffer for the outgoing image must be aligned on a cache line size boundary, and must be a multiple of the cache line size. This value is given by the constant CACHE_L2_LINESIZE, and is equal to 0x80 on C64x processors.

The scratch buffer must be allocated in on-chip memory. The function rounds up the given pointer to an 8-byte alignment and uses 16928 (0x4220) bytes, or 28576 (0x6FA0) if CIF size is used. Set to NULL on DM642 based systems.

2.2.34 MangoAV_HW_set_cam_input_param Function

Set video input parameter

C++

```
MANGOERROR_error_t MangoAV_HW_set_cam_input_param(int cam_num, Attrib_type_T attr, int val);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int cam_num`	Video input port number
`Attrib_type_T attr`	Attribute to set. May be one of: ATTRIB_BRIGHTNESS (0x00 - 0xff, 0x80 default) ATTRIB_CONTRAST (0x00 - 0x7f, 0x44 default) ATTRIB_COLOR (0x00 - 0x7f, 0x40 default)
`int val`	Value to set.

Returns

MANGOERROR_INVALID_PARAMETERS - If any parameter was invalid.

MANGOERROR_SUCCESS - ok.

Description

Modify brightness, contrast or color by changing appropriate register in video decoder chip.

2.2.35 MangoAV_HW_set_led Function

Sets user-definable LEDs

C++

```
MANGOERROR_error_t MangoAV_HW_set_led(unsigned char leds);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`unsigned char leds`	LEDS to light

Returns

MANGOERROR_SUCCESS - ok. MANGOERROR_FAILURE - this board does not support LEDs.

Remarks

Raven-D only. Seven LEDs can be controlled, one remaining LED is a power indicator and cannot be user controlled. Setting 1 turns the LED on (lit), 0 turns it off.

On new (Mark III) boards: Seven bits control seven LEDs. All seven LEDs can be controlled by either DSP. See MangoAV_HW_set_led_control (see page 22).

On older boards: DSP 0 can set six LEDs, DSP 1 can set one (this is the least significant bit).

2.2.36 MangoAV_HW_set_led_control Function

Determines which DSP controls each LED.

C++

```
MANGOERROR_error_t MangoAV_HW_set_led_control(unsigned char leds);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`unsigned char leds`	Bitmask for LEDs. 0 - controlled by DSP 0, 1 - controlled by DSP 1.

Returns

MANGOERROR_SUCCESS - ok. MANGOERROR_FAILURE - board does not support this command.

Remarks

Raven-D Mark III only. Command must be called by DSP 0.

2.2.37 MangoAV_HW_set_ttl_out Function

Set status of TTL outputs (on boards supporting TTL).

C++

```
unsigned char MangoAV_HW_set_ttl_out(unsigned char data);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`unsigned char data`	Each bit sets a single output TTL.

Returns

The previous setting of the TTL out is returned.

2.2.38 MangoAV_HW_set_video_loopback Function

Enables hardware video loopback on video input chip. The video input port number to use is the same as in MangoAV_HW_open_cam_input (see page 14). The loopback output will echo the selected input.

C++

```
MANGOERROR_error_t MangoAV_HW_set_video_loopback(const int cam_num, const int isLoopback);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`const int cam_num`	Video input port number.
`const int isLoopback`	1 - activate loopback, 0 - disable.

Returns

- MANGOERROR_SUCCESS - successful.
- MANGOERROR_ERR_NOT_IMPLEMENTED - this board does not support video loopback.

Remarks

Raven-HX104 only.

This feature is recommended to be used for testing purposes only, as it may cause degradation of the input signal.

2.2.39 MangoAV_HW_Write_UART Function

Writes data to the UART.

C++

```
MANGOERROR_error_t MangoAV_HW_Write_UART(int iUartIndex, char * pBuff, int iSize);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`int iUartIndex`	Physical UART index (0/1)
`char * pBuff`	Data buffer
`int iSize`	Buffer length

Returns

MANGOERROR_SUCCESS - ok.

MANGOERROR_INVALID_PARAMETERS - UART index invalid.

MANGOERROR_ERR_NOT_IMPLEMENTED -this board does not support UART.

Remarks

Function will block untill all the data has been sent.

2.2.40 qdma_memcpy Function

Auxiliary QDMA memory copy function.

C++

```
void qdma_memcpy(void* dst, void* src, int cnt);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`void* dst`	Destination address.
`void* src`	Source address.
`int cnt`	Amount (in bytes) to copy.

Returns

nothing.

Description

This function enables a quick memory copy using QDMA.

Remarks

All addresses and counts are assumed to be 4-byte aligned.

Priority for transfer is EDMA_OPT_PRI_LOW.

Requests longer than 0x40000 bytes will be split into smaller requests.

Transfers using this function bypass the L2 Cache mechanism.

Function returns immediately and does not wait for transfer to complete. See qdma_memcpy_wait (see page 25).

Up to four QDMA requests may be issued sequentially; Additional requests may stall the CPU or be lost. It is recommended to use qdma_memcpy_wait (see page 25) to ensure that the queue is empty before submitting additional requests.

2.2.41 qdma_memcpy_array Function

Auxiliary QDMA 2D memory copy function.

C++

```
void qdma_memcpy_array(void* dst, void* src, unsigned short rows, unsigned short cols,
unsigned short row_inc, int inc_src, int inc_dst);
```

File

MangoAV_HWExp.h (see page 26)

Parameters

Parameters	Description
`void* dst`	Destination address.
`void* src`	Source address.
`unsigned short rows`	Number of rows.
`unsigned short cols`	Number of bytes in each row.
`unsigned short row_inc`	Distance (in bytes) to skip between two consecutive rows.
`int inc_src`	Whether source is consecutive (0) or array (1).
`int inc_dst`	Whether destination is consecutive (0) or array (1).

Returns

nothing.

Description

This function enables a quick two-dimensional memory copy using QDMA.

Remarks

All addresses and counts are assumed to be 4-byte aligned except for number of rows.

Priority for transfer is EDMA_OPT_PRI_LOW.

Note that transfer parameters are limited to 16-bit (short) values.

Transfers using this function bypass the L2 Cache mechanism.

Function returns immediately and does not wait for transfer to complete. See qdma_memcpy_wait (see page 25).

Up to four QDMA requests may be issued sequentially; Additional requests may stall the CPU or get lost. It is recommended to use qdma_memcpy_wait (see page 25) to ensure that the queue is empty before submitting additional requests.

2.2.42 qdma_memcpy_wait Function

Waits (spins) until all pending qdma_memcpy (see page 24) requests complete.

C++

```
void qdma_memcpy_wait();
```

File

MangoAV_HWExp.h (see page 26)

Returns

nothing.

Description

Code waits until EDMA_OPT_PRI_LOW priority queue is empty, therefore waiting until all previous qdma_memcpy (see page 24) requests are satisfied.

Remarks

Note that this function may be confused if you use low-priority EDMAs elsewhere in your project.

2.3 Files

The following table lists files in this documentation.

Files

File	Description
MangoAV_HWExp.h (see page 26)	This is file MangoAV_HWExp.h.

2.3.1 MangoAV_HWExp.h

This is file MangoAV_HWExp.h.

Enumerations

Enumeration	Description
Baud_Rate_T (see page 2)	This is type Baud_Rate_T.
Bit_Count_T (see page 2)	This is type Bit_Count_T.
I2C_Command_T (see page 3)	Commands for MangoAV_HW_C64_i2c_load (see page 6)
Image_Formats_T (see page 3)	Image formats for MangoAV_HW_get_img_from_cam (see page 11)
Parity_T (see page 3)	This is type Parity_T.
Stop_Bit_t (see page 4)	This is type Stop_Bit_t.

Functions

Function	Description
MangoAV_HW_C64_get_gpStreams (see page 5)	Returns SIO stream handles for inter-DSP general-purpose communications.
MangoAV_HW_C64_i2c_load (see page 6)	Setup video chips (C64 based devices only)
MangoAV_HW_close_cam_input (see page 6)	Closes video input and frees resources.
MangoAV_HW_close_mic_input (see page 7)	Closes audio channel and frees resources.
MangoAV_HW_close_spk_output (see page 7)	Closes output audio channel and frees resources.
MangoAV_HW_close_tv_output (see page 7)	Closes TV output and frees resources.
MangoAV_HW_Close_UART (see page 8)	Closes a UART port
MangoAV_HW_Config_UART (see page 8)	Initializes and configures a UART port
MangoAV_HW_E2PROM_Read (see page 9)	Reads data from E2PROM.
MangoAV_HW_E2PROM_Write (see page 9)	Writes data to E2PROM.
MangoAV_HW_get_audio_frame_from_mic (see page 10)	Gets audio data from audio input channel.
MangoAV_HW_get_cam_input_param (see page 10)	Get video input parameter
MangoAV_HW_get_img_from_cam (see page 11)	Get image from video in
MangoAV_HW_get_ttl_in (see page 12)	Read status of TTL inputs (on boards supporting TTL).
MangoAV_HW_get_ttl_out (see page 12)	Get current status of TTL outputs (on boards supporting TTL).
MangoAV_HW_Get_Version (see page 12)	Returns lib version.
MangoAV_HW_get_vid_status (see page 13)	Check video input status
MangoAV_HW_init (see page 13)	Initialize hardware and library
MangoAV_HW_init_main (see page 14)	Perform further initializations that can only be called from main().
MangoAV_HW_open_cam_input (see page 14)	Opens camera (video in) stream.
MangoAV_HW_open_mic_input (see page 15)	Opens audio input channel
MangoAV_HW_open_spk_output (see page 16)	Opens audio output channel
MangoAV_HW_open_tv_output (see page 16)	Open analog monitor (TV) output
MangoAV_HW_query_cam (see page 17)	Test whether image is ready to be received from video input.
MangoAV_HW_query_monitor (see page 17)	Check whether monitor is ready for output
MangoAV_HW_Read_UART (see page 17)	Reads a requested byte count from the UART input buffer
MangoAV_HW_resetBoard (see page 18)	Resets the board, equivalent to pressing the board's Reset button. Flash-based boards will reboot.
MangoAV_HW_RTC_Get_Status (see page 18)	Reads status from Real Time Clock chip.
MangoAV_HW_RTC_Get_Time (see page 19)	Reads time and date from Real Time Clock chip.
MangoAV_HW_RTC_Reset (see page 19)	Resets the Real Time Clock chip.
MangoAV_HW_RTC_Set_Time (see page 20)	Sets time and date on Real Time Clock chip.
MangoAV_HW_send_audio_frame_to_spk (see page 20)	Sends audio data to audio output channel.
MangoAV_HW_send_img_to_monitor (see page 20)	Sends image to monitor (TV) output
MangoAV_HW_set_cam_input_param (see page 21)	Set video input parameter
MangoAV_HW_set_led (see page 22)	Sets user-definable LEDs
MangoAV_HW_set_led_control (see page 22)	Determines which DSP controls each LED.

MangoAV_HW_set_ttl_out (see page 22)	Set status of TTL outputs (on boards supporting TTL).
MangoAV_HW_set_video_loopback (see page 23)	Enables hardware video loopback on video input chip. The video input port number to use is the same as in MangoAV_HW_open_cam_input (see page 14). The loopback output will echo the selected input.
MangoAV_HW_Write_UART (see page 23)	Writes data to the UART.
qdma_memcpy (see page 24)	Auxiliary QDMA memory copy function.
qdma_memcpy_array (see page 24)	Auxiliary QDMA 2D memory copy function.
qdma_memcpy_wait (see page 25)	Waits (spins) until all pending qdma_memcpy (see page 24) requests complete.

Legend

	Method

Structures

Structure	Description
TimeOfDay_T (see page 4)	This is type TimeOfDay_T.

Index