Host Libraries API Documentation

Generated by Doxygen 1.7.6.1

Mon Aug 18 2014 21:20:57

Contents

Chapter 1

Host Libraries

Typical Configuration During Configuration

Typical Function Call Sequence for Auto-Configureing a System

1. apdm_ctx_allocate_new_context
2. apdm_open_all_access_points
3. apdm_autoconfigure_devices_and_accesspoint2
4. apdm_ctx_disconnect
5. apdm_ctx_free_context

Typical Configuration During Data Straming

Typical Function Call Sequence for Streaming Data From an Already Configured System

1. apdm_ctx_allocate_new_context
2. apdm_open_all_access_points
3. apdm_set_max_sample_delay_ms
4. apdm_get_device_id_list
5. apdm_sync_record_list_head
6. apdm_get_next_access_point_record_list (called many times in a loop)

7. apdm_extract_data_by_device_id (called to retrieve per-device data from the last list retrieved)
8. apdm_ctx_disconnect
9. apdm_ctx_free_context

Chapter 2

Deprecated List

Member adpm_ap_set_max_latency_value (apdm_ap_handle_t ap_handle, const uint32_t max_latency_ms)

This has been replaced by adpm_ap_set_max_latency_value_seconds(). This function will be removed after Jan 2011.

Member apdm_ap_get_gpio_value (apdm_ap_handle_t ap_handle, const apdm_ap_gpio_pin_t gpio_pin, bool *output_value)

replaced with apdm_ap_get_io_value() for more general purpose IO features. Will be removed after May 2013.

Member apdm_ap_set_gpio_value (apdm_ap_handle_t ap_handle, const apdm_ap_gpio_pin_t gpio_pin, const bool output_value)

replaced with apdm_ap_set_io_value() for more general purpose IO features. Will be removed after May 2013.

Member apdm_autoconfigure_devices_and_accesspoint (apdm_ctx_t context, const uint8_t wireless_channel_number)

use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

Member apdm_autoconfigure_devices_and_accesspoint2 (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card)

use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

Member apdm_autoconfigure_devices_and_accesspoint3 (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card, const bool erase_sd_card)

use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

Member apdm_ctx_ap_get_gpio_value (apdm_ctx_t context, const uint32_t ap_-id, const apdm_ap_gpio_pin_t gpio_pin, bool *output_value)

replaced with apdm_ctx_ap_get_io_value() for more general IO. Will be removed after May 2013.

Member apdm_ctx_ap_set_gpio_value (apdm_ctx_t context, const uint32_t ap_-id, const apdm_ap_gpio_pin_t gpio_pin, const bool output_value)

replaced with apdm_ctx_ap_get_io_value() for more general IO. Will be removed after May 2013.

Member apdm_sensor_list_attached_sensors (uint32_t *serial_number_buffer, const uint32_t buffer_length)

non-standard function semantics, see apdm_sensor_list_attached_sensors3(). Will be removed after March 2011.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Module Documentation

5.1 Setup

Functions

- APDM_EXPORT int apdm_ctx_set_correlation_fifo_temp_directory (const char *directory)
- APDM_EXPORT int apdm_calibration_override_minimum_supported_version (const uint32_t new_version)
- APDM_DEPRECATED APDM_EXPORT int apdm_ctx_ap_get_gpio_value (apdm_ctx_t context, const uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, bool *output_value)
- APDM_EXPORT int apdm_ctx_ap_get_io_value (apdm_ctx_t context, const uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, uint32_t *output_value)
- APDM_EXPORT int apdm_ctx_ap_set_gpio_value (apdm_ctx_t context, const uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, const bool output_value)
- APDM_EXPORT int apdm_ctx_ap_set_io_value (apdm_ctx_t context, const uint32_t ap_id, const apdm_ap_gpio_pin_t gpio_pin, const uint32_t output_value)
- APDM_EXPORT int apdm_ctx_set_minimum_sync_value (apdm_ctx_t context, const uint64_t minimum_sync_value)
- APDM_EXPORT int apdm_ctx_open_all_access_points (apdm_ctx_t context)
- APDM_EXPORT int apdm_init_access_point_wireless (apdm_ap_handle_t ap_handle, const uint8_t wireless_channel_1, const uint8_t wireless_channel_2, const uint32_t device_rx_address_high_order_bytes_A, const uint32_t device_rx_address_high_order_bytes_B, const uint8_t radio1_pipe_count, const uint8_t radio2_pipe_count)
- APDM_EXPORT int apdm_exit (void)

- APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint4 (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag)
- APDM_EXPORT int apdm_ctx_autoconfigure_devices_and_accesspoint5 (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag, const apdm_monitor_decimation_rate_t decimation_rate)
- APDM_EXPORT int apdm_init_streaming_config (apdm_streaming_config_t *streaming_config)
- APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint_streaming (apdm_ctx_t context, apdm_streaming_config_t *streaming_config)
- APDM_EXPORT int apdm_apply_autoconfigure_sensor_config (apdm_ctx_t context, apdm_device_handle_t ds_handle)
- APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint_wireless (apdm_ctx_t context, const uint8_t wireless_channel_number)
- APDM_EXPORT int apdm_autoconfigure_mesh_sync (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag)
- APDM_EXPORT int apdm_autoconfigure_mesh_sync2 (apdm_ctx_t context, const uint8_t wireless_channel_number)
- APDM_EXPORT int apdm_configure_all_attached_sensors (apdm_ctx_t context, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag)
- APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint (apdm_ctx_t context, const uint8_t wireless_channel_number)
- APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint2 (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card)
- APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint3 (apdm_ctx_t context, const uint8_t wireless_channel_number, const bool enable_sd_card, const bool erase_sd_card)
- int apdm_configure_all_attached_sensors_mesh (apdm_ctx_t context, const uint32_t wireless_channel, const bool enable_sd_card, const bool erase_sd_card, const bool accel_full_scale_mode, const bool enable_accel, const bool enable_gyro, const bool enable_mag)

5.1.1 Function Documentation

5.1.1.1 APDM_EXPORT int apdm_apply_autoconfigure_sensor_config (apdm_ctx_t *context,* apdm_device_handle_t *ds_handle*)

When apdm_autoconfigure_devices_and_accesspoint_streaming() is called with set_configuration_on_device set to false, this function can be called after the fact to apply the configuration of the monitor to the respective monitor that is on the docking station.

Parameters

context	
ds_handle	A docking handle, that has been opened, and has a monitor present in it.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ds_get_docked_module_id(), apdm_log_debug(), and apdm_log_error().

5.1.1.2 APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint (apdm_ctx_t *context,* const uint8_t *wireless_channel_number*)

Deprecated use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

See also

apdm_autoconfigure_devices_and_accesspoint4()

References apdm_autoconfigure_devices_and_accesspoint4().

5.1.1.3 APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint2 (apdm_ctx_t *context,* const uint8_t *wireless_channel_number,* const bool *enable_sd_card*)

Deprecated use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

See also

apdm_autoconfigure_devices_and_accesspoint4()

References apdm_autoconfigure_devices_and_accesspoint4().

5.1.1.4 APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint3 (apdm_ctx_t *context*, const uint8_t *wireless_channel_number*, const bool *enable_sd_card*, const bool *erase_sd_card*)

Deprecated use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

See also

apdm_autoconfigure_devices_and_accesspoint4()

References apdm_autoconfigure_devices_and_accesspoint4().

5.1.1.5 APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint4 (apdm_ctx_t *context*, const uint8_t *wireless_channel_number*, const bool *enable_sd_card*, const bool *erase_sd_card*, const bool *accel_full_scale_mode*, const bool *enable_accel*, const bool *enable_gyro*, const bool *enable_mag*)

This function will automatically configure all attached access points and devices in such a way that data can be streamed from the the system.

Parameters

context	
wireless_channel_number	The base wireless channel to transmit data on, 0-100.
enable_sd_card	Boolean indicating weather or not data should be logged to the SD card on the device.
erase_sd_card	Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.
accel_full_scale_mode	If true, then accelerometers will be in +/- 6g mode, if false, then they will be in +/- 2g mode
enable_accel	Enable the accelerometers
enable_gyro	Enable the gyros
enable_mag	Enable the magnetometers

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_autoconfigure_devices_and_accesspoint(), apdm_autoconfigure_devices_and_accesspoint2(), and apdm_autoconfigure_devices_and_accesspoint3().

5.1.1.6 APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint_-streaming (apdm_ctx_t *context,* apdm_streaming_config_t * *streaming_config*)

Used to autoconfigure accesspoints and sensors based on contents of apdm_-streaming_config_t data structure, replacement for the numeric variations of apdm-_autoconfigure_devices_and_accesspoint###() functions.

Parameters

context	
**streaming-_config*	The configuration to be applied to the system.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.1.1.7 APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint_wireless (apdm_ctx_t *context,* const uint8_t *wireless_channel_number*)

This function is similar to apdm_autoconfigure_devices_and_accesspoint4(), except that it doesn't override whatever device settings are already present on the attached devices.

Parameters

context	
wireless_-channel_-number	The base wireless channel to transmit data on, 0-100.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.1.1.8 APDM_EXPORT int apdm_autoconfigure_mesh_sync (apdm_ctx_t *context,* const uint8_t *wireless_channel_number,* const bool *enable_sd_card,* const bool *erase_sd_card,* const bool *accel_full_scale_mode,* const bool *enable_accel,* const bool *enable_gyro,* const bool *enable_mag*)

This function is used to configure all Motion Monitors currently attached to the host in synchronized logging mode, maximum of 32 devices.

Parameters

context	
wireless_channel_number	The wireless channel used to synchronize time between the motion monitors in the mesh time sync group.
enable_sd_card	Boolean indicating weather or not data should be logged to the SD card on the device.
erase_sd_card	Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.
accel_full_scale_mode	If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode
enable_accel	Enable the accelerometers
enable_gyro	Enable the gyros
enable_mag	Enable the magnitometers

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_configure_all_attached_sensors_mesh(), and apdm_log_info().

5.1.1.9 APDM_EXPORT int apdm_autoconfigure_mesh_sync2 (apdm_ctx_t *context*, const uint8_t *wireless_channel_number*)

Similar to apdm_autoconfigure_mesh_sync(), however this will configure all attached monitors into synchronized logging mode without modifying the pre-existing sensor settings on the monitors.

Parameters

context	
wireless_channel_number	The wireless channel used to synchronize time between the motion monitors in the mesh time sync group.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_info().

5.1.1.10 APDM_EXPORT int apdm_calibration_override_minimum-_supported_version (const uint32_t *new_version*)

Allows you to override the minimum calibration version number used to validate calibration versions on motion sensors.

Parameters

new_version	Version number, e.g. 4 Set this to zero to use library default version number.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.1.1.11 APDM_EXPORT int apdm_configure_all_attached_sensors (apdm_ctx_t *context,* const bool *enable_sd_card,* const bool *erase_sd_card,* const bool *accel_full_scale_mode,* const bool *enable_accel,* const bool *enable_gyro,* const bool *enable_mag*)

This function will automatically configure all attached access points and devices in such a way that data can be streamed from the the system.

Parameters

context	
enable_sd_-card	Boolean indicating weather or not data should be logged to the SD card on the device.
erase_sd_-card	Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.
accel_full_-scale_mode	If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode
enable_-accel	Enable the accelerometers
enable_gyro	Enable the gyros
enable_mag	Enable the magnitometers

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_streaming_config_t::accel_full_scale_mode, apdm_init_streaming_-config(), apdm_streaming_config_t::enable_accel, apdm_streaming_config_t::enable_-

gyro, apdm_streaming_config_t::enable_mag, apdm_streaming_config_t::enable_sd_-card, and apdm_streaming_config_t::erase_sd_card.

5.1.1.12 int apdm_configure_all_attached_sensors_mesh (apdm_ctx_t *context,* const uint32_t *wireless_channel,* const bool *enable_sd_card,* const bool *erase_sd_card,* const bool *accel_full_scale_mode,* const bool *enable_accel,* const bool *enable_gyro,* const bool *enable_mag*)

This function is used to configure all opals currently attached to the host in mesh time synchronization and data logging mode.

Parameters

context	
wireless_-channel	The wireless channel used to synchronize time between the opals in the mesh time sync group.
enable_sd_-card	Boolean indicating weather or not data should be logged to the SD card on the device.
erase_sd_-card	Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.
accel_full_-scale_mode	If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode
enable_-accel	Enable the accelerometers
enable_gyro	Enable the gyros
enable_mag	Enable the magnitometers

Referenced by apdm_autoconfigure_mesh_sync().

5.1.1.13 APDM_DEPRECATED APDM_EXPORT int apdm_ctx_ap_get_gpio_value (apdm_ctx_t *context,* const uint32_t *ap_id,* const apdm_ap_gpio_pin_t *gpio_pin,* bool * *output_value*)

Parameters

context	The context of communications.
ap_id	The ID number of the AP to manipulate the GPIO on.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_-types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or AP-DM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
*output_-value	Destination into which to store the current value of the GPIO pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Deprecated replaced with apdm_ctx_ap_get_io_value() for more general IO. Will be removed after May 2013.

References apdm_ctx_ap_get_io_value().

5.1.1.14 APDM_EXPORT int apdm_ctx_ap_get_io_value (apdm_ctx_t *context*, const uint32_t *ap_id*, const apdm_ap_gpio_pin_t *gpio_pin*, uint32_t * *output_value*)

Parameters

context	The context of communications.
ap_id	The ID number of the AP to manipulate the GPIO on.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
*output_value	Destination into which to store the current value of the GPIO pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_get_io_value(), and apdm_log_error().

Referenced by apdm_ctx_ap_get_gpio_value().

5.1.1.15 APDM_EXPORT int apdm_ctx_ap_set_gpio_value (apdm_ctx_t *context*, const uint32_t *ap_id*, const apdm_ap_gpio_pin_t *gpio_pin*, const bool *output_value*)

Parameters

context	The context of communications.
ap_id	The ID number of the AP to manipulate the GPIO on.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
output_value	New value to set on a GPIO pin that has been configured as an output pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Deprecated replaced with apdm_ctx_ap_get_io_value() for more general IO. Will be removed after May 2013.

References apdm_ctx_ap_set_io_value().

5.1.1.16 APDM_EXPORT int apdm_ctx_ap_set_io_value (apdm_ctx_t *context*, const uint32_t *ap_id*, const apdm_ap_gpio_pin_t *gpio_pin*, const uint32_t *output_value*)

Parameters

context	The context of communications.
ap_id	The ID number of the AP to manipulate the GPIO on.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
output_value	New value to set on a GPIO pin that has been configured as an output pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_set_io_value(), and apdm_log_error().

Referenced by apdm_ctx_ap_set_gpio_value().

5.1.1.17 APDM_EXPORT int apdm_ctx_autoconfigure_devices_and_accesspoint5 (apdm_ctx_t *context*, const uint8_t *wireless_channel_number*, const bool *enable_sd_card*, const bool *erase_sd_card*, const bool *accel_full_scale_mode*, const bool *enable_accel*, const bool *enable_gyro*, const bool *enable_mag*, const apdm_monitor_decimation_rate_t *decimation_rate*)

Same as apdm_autoconfigure_devices_and_accesspoint4(), but extra parameter allows you to set the decimation rate.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.1.1.18 APDM_EXPORT int apdm_ctx_open_all_access_points (apdm_ctx_t *context*)

Will cause all access points connected to the host to be opened and associated with the passed handle. Note: Accesspoints can only be opened by one application at a time. If there are other applications, such as Motion Studio running that have already open the attached Accesspoints, then this will fail to open them.

Parameters

context	The handle for which to associate all opened access points.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_connect(), apdm_ap_disconnect(), apdm_ap_get_case_id(), apdm_ap_get_id_and_board_version(), apdm_ap_get_num_access_points_on_host1(), apdm_ctx_initialize_context(), apdm_get_time_ms_64(), apdm_log_context(), apdm_log_debug(), apdm_log_error(), apdm_log_info(), and apdm_strerror().

5.1.1.19 APDM_EXPORT int apdm_ctx_set_correlation_fifo_temp_directory (const char * *directory*)

Only relevant to windows. Sets the directory name into which correlation fifo temp files should be located.

Parameters

**directory*	Directory into which fifo files should be placed, with trailing slash.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.1.1.20 APDM_EXPORT int apdm_ctx_set_minimum_sync_value (apdm_ctx_t *context,* const uint64_t *minimum_sync_value*)

Parameters

context	The context of communications.
minimum_sync_value	The minimum sync value that you want sensors to send out. This is useful for skipping ahead in a data stream.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References adpm_ap_set_minimum_sync_value().

5.1.1.21 APDM_EXPORT int apdm_exit (void)

This function clears out any kernel event handlers or callbacks. Before unloading the DLL/SO/DYLIB and program termination, this function should be called. This function should be called just before you program exits (do not call this function if you intended to continue using the APDM library, wait until your completely done.)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.1.1.22 APDM_EXPORT int apdm_init_access_point_wireless (apdm_ap_handle_t *ap_handle*, const uint8_t *wireless_channel_1*, const uint8_t *wireless_channel_2*, const uint32_t *device_rx_address_high_order_bytes_A*, const uint32_t *device_rx_address_high_order_bytes_B*, const uint8_t *radio1_pipe_count*, const uint8_t *radio2_pipe_count*)

Parameters

ap_handle	The access point handle to be configured
wireless_channel_1	Wireless channel number to use on the first radio
wireless_channel_2	Wireless channel number to use on the second radio
device_rx_address_high_order_bytes_A	The high-order block ID used for data filtering/matching by the radio (has bit-sequencing constraints for proper hardware behavior)
deviceRX-Address-HighOrder-BytesB	The high-order block ID used for data filtering/matching by the radio (has bit-sequencing constraints for proper hardware behavior)
radio1_pipe_count	Number of pipes to enable
radio2_pipe_count	Number of pipes to enable

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_configure_accesspoint().

5.1.1.23 APDM_EXPORT int apdm_init_streaming_config (apdm_streaming_config_t * *streaming_config*)

Initializes streaming_configuration data structure to default values.

wireless_channel_number = 80; enable_sd_card = true; erase_sd_card = false; accel_full_scale_mode = true; enable_accel = true; enable_gyro = true; enable_mag = true; apply_new_sensor_modes = true; decimation_rate = APDM_DECIMATE_5x2; output_select_rate = APDM_OUTPUT_SELECT_RATE_128; button_enable = false;

Parameters

**streaming_config*	Pointer to apdm_streaming_config_t structure to be initialized

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_streaming_config_t::accel_full_scale_mode, apdm_streaming_config_t::apply_new_sensor_modes, apdm_streaming_config_t::button_enable, apdm_streaming_config_t::decimation_rate, apdm_streaming_config_t::enable_accel, apdm_streaming_config_t::enable_gyro, apdm_streaming_config_t::enable_mag, apdm_streaming_config_t::enable_sd_card, apdm_streaming_config_t::erase_sd_card, and apdm_streaming_config_t::set_configuration_on_device.

Referenced by apdm_configure_all_attached_sensors().

5.2 Context

Functions

- APDM_EXPORT int apdm_ctx_get_expected_number_of_sensors2 (apdm_ctx_t context, uint32_t *dest)
- APDM_EXPORT enum APDM_Status apdm_ctx_set_error_handling_mode (apdm_ctx_t context, enum APDMErrorHandlingBehavior new_mode)
- APDM_EXPORT int apdm_ctx_get_sensor_compensation_data (apdm_ctx_t context, apdm_sensor_compensation_t *dest_comp_data, const int32_t sensor_index)
- APDM_EXPORT int apdm_ctx_set_sensor_compensation_data (apdm_ctx_t context, const apdm_sensor_compensation_t *src_comp_data, const int32_t sensor_index)
- APDM_EXPORT int apdm_ctx_get_expected_sync_delta (apdm_ctx_t context, uint16_t *dest_expected_sync_delta)
- APDM_EXPORT int apdm_ctx_set_metadeta_uint32 (apdm_ctx_t context, const uint32_t device_id, const uint32_t value)
- APDM_EXPORT int apdm_ctx_set_metadata_string (apdm_ctx_t context, const uint32_t device_id, const char *str)
- APDM_EXPORT uint32_t apdm_ctx_get_metadata_uint32 (apdm_ctx_t context, const uint32_t device_id)
- APDM_EXPORT int apdm_ctx_get_wireless_configuration_mode (apdm_ctx_t context, int *dest)
- APDM_EXPORT int apdm_ctx_get_device_info (apdm_ctx_t context, const uint32_t device_id, apdm_device_info_t *dest)
- APDM_EXPORT int apdm_ctx_get_num_access_points_found (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_get_ap_id_for_ap_index (apdm_ctx_t context, const int ap_index, uint32_t *dest)
- APDM_EXPORT uint32_t apdm_ctx_get_num_sample_lists_collected (apdm_ctx_t context)
- APDM_EXPORT uint32_t apdm_ctx_get_num_samples_collected (apdm_ctx_t context)
- APDM_EXPORT uint32_t apdm_ctx_get_num_samples_collected_from_device (apdm_ctx_t context, const uint32_t device_id)
- APDM_EXPORT uint32_t apdm_ctx_get_total_omitted_sample_sets (apdm_ctx_t context)
- APDM_EXPORT uint32_t apdm_ctx_get_num_omitted_sample_sets (apdm_ctx_t context)
- APDM_EXPORT uint32_t apdm_ctx_get_num_omitted_samples (apdm_ctx_t context)
- APDM_EXPORT uint32_t apdm_ctx_get_total_omitted_samples (apdm_ctx_t context)

- APDM_EXPORT int apdm_ctx_get_sampling_frequency (apdm_ctx_t context, uint32_t *dest)
- APDM_EXPORT int apdm_ctx_extract_data_by_device_id (apdm_ctx_t context, const uint32_t device_id, apdm_record_t *dest)
- APDM_EXPORT int apdm_ctx_get_next_access_point_record (apdm_ctx_t context, apdm_record_t *data, const int ap_index_number, const bool allow_ap_transfer_flag)
- APDM_EXPORT int apdm_ctx_sync_record_list_head (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_get_next_access_point_record_list (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_purge_older_samples (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_flush_ap_fifos (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_extract_next_sample (apdm_ctx_t context, apdm_record_t *dest_record)
- APDM_EXPORT int apdm_ctx_get_next_synchronization_event (apdm_ctx_t context, apdm_external_sync_data_t *dest)
- APDM_EXPORT int apdm_ctx_populate_buffers (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_get_next_record (apdm_ctx_t context, apdm_record_t *dest)
- APDM_EXPORT int apdm_ctx_get_next_record2 (apdm_ctx_t context, apdm_record_t *dest, const bool allow_ap_transfer_flag)
- APDM_EXPORT int32_t apdm_ctx_get_device_id_by_index (apdm_ctx_t context, const uint32_t sensor_index)
- APDM_EXPORT int **apdm_ctx_set_requested_device_states** (apdm_ctx_t context, const enum RequestedDeviceState state)
- APDM_EXPORT int **apdm_ctx_set_requested_device_state** (apdm_ctx_t context, const enum RequestedDeviceState state, const int ap_index_number)
- APDM_EXPORT int apdm_ctx_get_device_index_by_id3 (apdm_ctx_t context, const uint32_t id, uint32_t *dest_index)
- APDM_EXPORT int apdm_ctx_get_device_id_list (apdm_ctx_t context, uint32_t *dest, const uint32_t destSize)
- APDM_EXPORT int apdm_ctx_initialize_context (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_disconnect (apdm_ctx_t context)
- APDM_EXPORT apdm_ctx_t apdm_ctx_allocate_new_context (void)
- APDM_EXPORT int apdm_ctx_free_context (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_persist_context_to_disk (apdm_ctx_t context, const char *filepath)
- APDM_EXPORT int apdm_ctx_restore_context_from_disk (apdm_ctx_t context, const char *filepath)
- APDM_EXPORT int apdm_ctx_disable_accesspoint_wireless (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_re_enable_accesspoint_wireless (apdm_ctx_t context)

- APDM_EXPORT int apdm_ctx_is_more_data_immediately_available (apdm_ctx_t context)
- APDM_EXPORT int apdm_ctx_avg_retry_count_for_device (apdm_ctx_t context, const uint32_t device_id)
- APDM_EXPORT int apdm_ctx_get_wireless_reliability_value (apdm_ctx_t context, const uint32_t device_id)
- APDM_EXPORT int apdm_ctx_get_wireless_streaming_status (apdm_ctx_t context, uint32_t *dest)
- APDM_EXPORT time_t apdm_ctx_get_last_received_timestamp_for_device (apdm_ctx_t context, const uint32_t device_id)
- APDM_EXPORT int apdm_ctx_set_max_sample_delay_seconds (apdm_ctx_t context, const uint16_t max_data_delay_seconds)
- APDM_EXPORT int apdm_ctx_set_orientation_model (apdm_ctx_t context, const apdm_orientation_model_t orientation_model)
- APDM_EXPORT int apdm_ctx_get_monitor_latency (apdm_ctx_t context, const uint32_t monitor_id, int64_t *dest)
- APDM_EXPORT int apdm_get_max_sample_delay_seconds (apdm_ctx_t context, uint16_t *dest)
- APDM_EXPORT int apdm_ctx_reset_num_samples_from_ap (apdm_ctx_t context)
- APDM_EXPORT int apdm_get_num_samples_from_ap (apdm_ctx_t context)
- APDM_EXPORT uint64_t apdm_ctx_estimate_now_sync_value (apdm_ctx_t context)

5.2.1 Function Documentation

5.2.1.1 APDM_EXPORT apdm_ctx_t apdm_ctx_allocate_new_context (void)

Allocates memory a handle to be used by the apdm libraries.

Returns

Non-zero on success, zero otherwise

References apdm_log_error(), and apdm_log_info().

Referenced by apdm_ctx_restore_context_from_disk().

5.2.1.2 APDM_EXPORT int apdm_ctx_avg_retry_count_for_device (apdm_ctx_t *context*, const uint32_t *device_id*)

Returns the average number of retries for samples coming from the given device, useful as a wireless reliability indicator for the device, (only accurate while actively streaming data thru the host libraries).

Parameters

context	
device_id	The device ID (this is different then the Case ID on the back of the monitor)

Returns

Negative error code on error, zero or higher with average number of retries per second for device id specified over previous 3 seconds.

Referenced by apdm_ctx_get_wireless_reliability_value().

5.2.1.3 APDM_EXPORT int apdm_ctx_disable_accesspoint_wireless (apdm_ctx_t *context*)

This function will disable the wireless radios and protocol on all the access points in the context, causing them to no longer transmit sync packets, nor be able to RX data from monitors.

Parameters

context	

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.4 APDM_EXPORT int apdm_ctx_disconnect (apdm_ctx_t *context*)

Disconnects from access points that are currently attached (USB bus handle disconnect)

Parameters

context	The handle to be disconnected

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_disconnect(), and apdm_sensor_close().

Referenced by apdm_ctx_restore_context_from_disk().

5.2.1.5 APDM_EXPORT uint64_t apdm_ctx_estimate_now_sync_value (apdm_ctx_t *context*)

This function will estimate the current sync value of the system. This should be good to within about 50ms, and is dependant on the timing latency of the USB bus on the host and the clock drift rate delta between the AP and the hose computer.

Parameters

context	The context

Returns

An estimate of the current sync value.

References apdm_get_now_sync_value_host().

Referenced by apdm_ctx_get_wireless_reliability_value(), and apdm_extract_next_sample_set().

5.2.1.6 APDM_EXPORT int apdm_ctx_extract_data_by_device_id (apdm_ctx_t *context,* const uint32_t *device_id,* apdm_record_t * *dest*)

Gets data for a particular device id from the most record list.

Parameters

context	
device_id	The device id for which you want to retrieve data for (this is different then the Case ID on the back of the monitor)
**dest*	The destination into which to put data.

Returns

APDM_OK on success, APDM_NO_MORE_DATA if no more data, error code otherwise.

References apdm_record_t::device_info_serial_number.

5.2.1.7 APDM_EXPORT int apdm_ctx_extract_next_sample (apdm_ctx_t *context,* apdm_record_t * *dest_record*)

Extracts the next single sample from the set of AP's used in the context

Parameters

context	The apdm handle
∗dest_record	The record into which the sample data is to be stored.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ctx_get_expected_number_of_sensors2(), and apdm_log_error().

5.2.1.8 APDM_EXPORT int apdm_ctx_flush_ap_fifos (apdm_ctx_t *context*)

This function is used to flush any data buffers or samples stored in RAM on the AP.

Parameters

context	The apdm handle

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.9 APDM_EXPORT int apdm_ctx_free_context (apdm_ctx_t *context*)

De-allocates memory used for the APDM handle context

Parameters

context	The handle to be deallocated.

References apdm_log_error().

Referenced by apdm_ctx_restore_context_from_disk().

5.2.1.10 APDM_EXPORT int apdm_ctx_get_ap_id_for_ap_index (apdm_ctx_t *context,* const int *ap_index,* uint32_t ∗ *dest*)

Parameters

context	
ap_index	The AP index number for which you want AP ID. This should be >= 0 and less than the number of APs configured.
∗dest_id	Destionation address into which the AP ID should be stored.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.2.1.11 APDM_EXPORT int32_t apdm_ctx_get_device_id_by_index (apdm_ctx_t *context,* const uint32_t *sensor_index*)

Parameters

context	The context for which you want the device id
sensor_index	The index of the device id for which you want

Returns

negative error code on error, zero if device is not found at the specified index, device ID $>$ 0 on success, only relevant after auto_configure has been called.

5.2.1.12 APDM_EXPORT int apdm_ctx_get_device_id_list (apdm_ctx_t *context,* uint32_t $*$ *dest,* const uint32_t *destSize*)

Parameters

context	The context for which you want the device id
$*$*dest*	Destination array of uint32_t's into which you want to store devices IDs, the (last+1) element will have a device id of zero.
destSize	The number of elements in the destination array.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.13 APDM_EXPORT int apdm_ctx_get_device_index_by_id3 (apdm_ctx_t *context,* const uint32_t *id,* uint32_t $*$ *dest_index*)

Parameters

context	The context for which you want the device id
id	The motion monitor ID for which you want the index of.
$*$*dest_index*	The index of the specified device id

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

Referenced by apdm_ctx_get_device_info(), apdm_ctx_get_next_access_point_record(), and apdm_ctx_get_wireless_streaming_status().

5.2.1.14 APDM_EXPORT int apdm_ctx_get_device_info (apdm_ctx_t *context*, const uint32_t *device_id*, apdm_device_info_t * *dest*)

Gets device detailed information about the device id passed in.

Parameters

context	The apdm handle
device_id	The ID of the device for which you'd like to get data (this is different then the Case ID on the back of the monitor)
*dest	The destination structure into which you'd like to store the data

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ctx_get_device_index_by_id3().

5.2.1.15 APDM_EXPORT int apdm_ctx_get_expected_number_of_sensors2 (apdm_ctx_t *context*, uint32_t * *dest*)

Returns the number of sensors that are configured in the context. This is with respect to an already-configured context. It is not necessarily the number of sensors attached to the system.

Parameters

context	The context of communications.

Returns

The number of sensors configured in the context.

Referenced by apdm_ctx_extract_next_sample(), apdm_ctx_get_next_record2(), apdm_ctx_get_num_samples_collected(), apdm_ctx_get_num_samples_collected_from_device(), apdm_ctx_get_sampling_frequency(), apdm_ctx_get_sensor_-

compensation_data(), apdm_ctx_get_wireless_streaming_status(), apdm_ctx_is_more_data_immediately_available(), apdm_ctx_purge_older_samples(), apdm_ctx_set_max_sample_delay_seconds(), apdm_ctx_set_sensor_compensation_data(), apdm_ctx_sync_record_list_head(), and apdm_extract_next_sample_set().

5.2.1.16 APDM_EXPORT int apdm_ctx_get_expected_sync_delta (apdm_ctx_t *context,* uint16_t * *dest_expected_sync_delta*)

Depending on the output rate (e.g. 128 samples per second, 80 samples per second etc), this will return the expected sync delta between any two samples.

Parameters

context	The apdm context
**dest_expected_sync_delta*	The destination into which to store the expected sync delta between any two samples

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.17 APDM_EXPORT time_t apdm_ctx_get_last_received_timestamp_for_device (apdm_ctx_t *context,* const uint32_t *device_id*)

Gets the unix epoch time of when the last time a sample was received for the specified device ID. If you find that it's been a "long" time since a sample has been received from a device, you may check the device is powered and within range of an access point, (only accurate while actively streaming data thru the host libraries).

Parameters

context	
device_id	The device ID (this is different then the Case ID on the back of the monitor)

Returns

Zero on error of if no data has been received, non-zero with the epoch time otherwise.

5.2.1.18 APDM_EXPORT uint32_t apdm_ctx_get_metadata_uint32 (apdm_ctx_t *context,* const uint32_t *device_id*)

Allows for the retrieval of metadata for a device id.

Parameters

context	The apdm handle
device_id	The device_id for which you want metadata (this is different then the Case ID on the back of the monitor)

Returns

The data associated with the device id, or zero if an error or no data having been set.

5.2.1.19 APDM_EXPORT int apdm_ctx_get_monitor_latency (apdm_ctx_t *context,* const uint32_t *monitor_id,* int64_t * *dest*)

Retrieves the latency of an individual monitor from the given context.

Parameters

context	The apdm context
monitor_id	The ID of the monitor for which you want to know the latency.
**dest*	The destination into which to store the latency value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_get_monitor_latency(), and apdm_log_error().

5.2.1.20 APDM_EXPORT int apdm_ctx_get_next_access_point_record (apdm_ctx_t *context,* apdm_record_t * *data,* const int *ap_index_number,* const bool *allow_ap_transfer_flag*)

Gets the next record from the access point indicated

Parameters

context	
**data*	Destination into which to place date. This can be null, if you want to only trigger transfers from the AP, but will always return APDM_NO_MORE_DATA.

ap_index_number	The index number of the AP on the host for which to retrieve data.
allow_ap_transfer_flag	Allow for the initiation of a new usb data transfer from the AP.

Returns

APDM_OK if data was retrieved, APDM_NO_MORE_DATA if no more data, error code otherwise.

References apdm_ctx_get_device_index_by_id3(), apdm_log_debug(), apdm_log_error(), apdm_log_warning(), apdm_strerror(), apdm_device_info_t::decimation_factor, apdm_record_t::device_info_serial_number, apdm_record_t::num_retrys, apdm_record_t::opt_select, apdm_device_info_t::protocol_version, apdm_record_t::source_ap_index, and apdm_record_t::sync_val32_low.

5.2.1.21 APDM_EXPORT int apdm_ctx_get_next_access_point_record_list (apdm_ctx_t *context*)

This function populates an list of records internal to the handle with a set of samples all corresponding to the same sync value (point in time). Depending on the error handling mode set, there may be some samples that are not populated or some partial sample sets that are skipped over.

Parameters

context	The apdm handle

Returns

APDM_OK If it was able to get a sample set according to the error handling mode, APDM_NO_MORE_DATA if there is no more data ready (just wait longer for more data to come in), or another code indicating what error occurred.

References apdm_extract_next_sample_set(), apdm_log_error(), and apdm_strerror().

5.2.1.22 APDM_EXPORT int apdm_ctx_get_next_record (apdm_ctx_t *context,* apdm_record_t * *dest*)

This function will retrieve the oldest sample currently in the library buffers. In the case of multiple samples having the same age, it will return one of the oldest. This function will provide good realtime responsiveness to the caller, however, you may experience duplicates in the data stream or samples coming slightly out of order as samples are

emitted as soon as it's available. This function takes a 2-5 milliseconds to execute, so avoid using it in tight data processing loops.

Note: this function does not necessarily return a record every time. If a record is available, it will be returned, but depending on timing, wireless conditions, and many other variables, a record may not be available.

Parameters

context	The apdm handle
∗dest	The record into which the data is stored.

Returns

APDM_OK upon success, APDM_NO_MORE_DATA if no data is available, error code otherwise.

References apdm_ctx_get_next_record2().

5.2.1.23 APDM_EXPORT int apdm_ctx_get_next_record2 (apdm_ctx_t *context,* apdm_record_t ∗ *dest,* const bool *allow_ap_transfer_flag*)

Same as apdm_ctx_get_next_record(), except it allows you to disable AP transfers. Note: this function does not necessarily return a record every time. If a record is available, it will be returned, but depending on timing, wireless conditions, and many other variables, a record may not be available.

Parameters

context	The apdm handle
∗dest	The record into which the data is stored.
allow_ap_transfer_flag	Flag to allow you to disable USB transfers from the AP's.

Returns

APDM_OK upon success, APDM_NO_MORE_DATA if no data is available, error code otherwise.

References apdm_ctx_get_expected_number_of_sensors2(), apdm_log_debug(), and apdm_log_error().

Referenced by apdm_ctx_get_next_record().

5.2.1.24 APDM_EXPORT int apdm_ctx_get_next_synchronization_event (apdm_ctx_t *context,* apdm_external_sync_data_t * *dest*)

This function is used to gather external synchronization I/O data events. For GPIO inputs, input signals are debounced over a 1/2560 second period of time, and the sync value tagged on the synchronization sample will be that of the sync value of the time of the rising edge of the signal. You must be streaming data at the time you call this function, as synchronization events are passed from the AP to the libraries at the time that data is received from the AP.

Parameters

context	The apdm handle
**dest*	Destination into which synchronization data is to be stored.

Returns

APDM_OK if dest was populated with data, APDM_NO_MORE_DATA if there is no synchronization event data available, error code otherwise.

5.2.1.25 APDM_EXPORT int apdm_ctx_get_num_access_points_found (apdm_ctx_t *context*)

Parameters

context	

Returns

The number of access points attached to the host

5.2.1.26 APDM_EXPORT uint32_t apdm_ctx_get_num_omitted_sample_sets (apdm_ctx_t *context*)

Returns

The number of omitted sample sets since the most recently requested sample set and the previously retrieved sample set.

5.2.1.27 APDM_EXPORT uint32_t apdm_ctx_get_num_omitted_samples (apdm_ctx_t *context*)

Returns

The number of omitted samples between the most recently requested sample set and the previously retrieved sample set.

5.2.1.28 APDM_EXPORT uint32_t apdm_ctx_get_num_sample_lists_collected (apdm_ctx_t *context*)

Returns

The total number of sample lists collected since the handle was initialized.

5.2.1.29 APDM_EXPORT uint32_t apdm_ctx_get_num_samples_collected (apdm_ctx_t *context*)

Returns

The total number of samples collected since the handle was initialized.

References apdm_ctx_get_expected_number_of_sensors2(), apdm_log_error(), and apdm_strerror().

5.2.1.30 APDM_EXPORT uint32_t apdm_ctx_get_num_samples_collected_from_device (apdm_ctx_t *context,* const uint32_t *device_id*)

Parameters

device_id	Device ID to get the number of samples for (this is different then the Case ID on the back of the monitor)

Returns

The total number of samples collected since the handle was initialized for the device id specified

References apdm_ctx_get_expected_number_of_sensors2(), apdm_log_error(), and apdm_strerror().

5.2.1.31 APDM_EXPORT int apdm_ctx_get_sampling_frequency (apdm_ctx_t *context,* uint32_t * *dest*)

Parameters

context	
∗dest	Destination into which to store the sampling frequency

Returns

Returns the sampling frequency that the devices are running at

References apdm_ctx_get_expected_number_of_sensors2().

5.2.1.32 APDM_EXPORT int apdm_ctx_get_sensor_compensation_data (apdm_ctx_t *context,* apdm_sensor_compensation_t ∗ *dest_comp_data,* const int32_t *sensor_index*)

Parameters

context	The apdm context
∗dest_comp_data	The destination into which to store compensation data for the sensor of index sensor_index.
sensor_index	The index of the sensor for which you want to retrieve compensation data.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ctx_get_expected_number_of_sensors2(), and apdm_log_error().

5.2.1.33 APDM_EXPORT uint32_t apdm_ctx_get_total_omitted_sample_sets (apdm_ctx_t *context*)

Returns

The number of omitted sample sets since the last time the context was initialized or since the last time apdm_sync_record_head_list() was called.

5.2.1.34 APDM_EXPORT uint32_t apdm_ctx_get_total_omitted_samples (apdm_ctx_t *context*)

Returns

The number of omitted samples since the context was initialized, or since the last time apdm_sync_record_head_list() was called.

5.2.1.35 APDM_EXPORT int apdm_ctx_get_wireless_configuration_mode (apdm_ctx_t *context*, int * *dest*)

Parameters

context	The apdm handle
*dest	The destination into which to store the configured wireless mode. The value will be from the enum apdm_wireless_mode_t

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.36 APDM_EXPORT int apdm_ctx_get_wireless_reliability_value (apdm_ctx_t *context*, const uint32_t *device_id*)

Used to get a number between 0 and 100 on how reliable the wireless connection is for a given device, (only accurate while actively streaming data thru the host libraries).

Parameters

context	
device_id	The device ID (this is different then the Case ID on the back of the monitor)

Returns

Negative error code on error, Zero to 100 on success, 100 being the best signal, zero being the worst (or no).

References apdm_calculate_sync_value_age(), apdm_ctx_avg_retry_count_for_device(), and apdm_ctx_estimate_now_sync_value().

5.2.1.37 APDM_EXPORT int apdm_ctx_get_wireless_streaming_status (apdm_ctx_t *context*, uint32_t * *dest*)

Parameters

context	
*dest	Destination into which to store the wireless streaming status by AP, of type apdm_ap_wireless_streaming_status_t.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ctx_get_device_index_by_id3(), and apdm_ctx_get_expected_-number_of_sensors2().

5.2.1.38 APDM_EXPORT int apdm_ctx_initialize_context (apdm_ctx_t *context*)

Used to initialize a handle context

Parameters

context	The handle to be initialized

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_init_handle().

Referenced by apdm_ctx_open_all_access_points().

5.2.1.39 APDM_EXPORT int apdm_ctx_is_more_data_immediately_available (apdm_ctx_t *context*)

Checks to see if more data is available in the host-resident sample buffers and if a subsequent call to get data would return without doing a USB bus transfer

Parameters

context	The apdm context to check data on

Returns

Zero if there is no more data, non-zero if there is more data available.

References apdm_ctx_get_expected_number_of_sensors2(), and apdm_log_error().

5.2.1.40 APDM_EXPORT int apdm_ctx_persist_context_to_disk (apdm_ctx_t *context,* const char * *filepath*)

This function will take a context, and persist all the configuration information to disk thus allowing it to be restored at a later time with apdm_ctx_restore_context_from_disk() and avoid the need to re-configure the system.

Parameters

context	The handle to be deallocated.
filepath	The file to save the context to

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.2.1.41 APDM_EXPORT int apdm_ctx_populate_buffers (apdm_ctx_t *context*)

This function will force a transmission of any samples in the AP's from the AP to the host and populate the internal buffers. The internal buffers of the library are where data is temporary stored immediately after a USB transfer. They are used when correlating groups of samples from multiple sensors and as a staging area to store data prior to emission from the libraries.

Parameters

context	The apdm handle

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

5.2.1.42 APDM_EXPORT int apdm_ctx_purge_older_samples (apdm_ctx_t *context*)

This function will purge all but the newest samples from the internal buffers. This is useful in a larger context, when you only want to get the most recent sample(s) for each monitor.

E.G. apdm_ctx_populate_buffers() apdm_ctx_purge_older_samples(); apdm_ctx_get_next_record2(allow_ap_transfer_flag=false) until it returns no more data, making sure to pass allow_ap_transfer_flag=false

Parameters

context	The apdm handle

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ctx_get_expected_number_of_sensors2().

5.2.1.43 APDM_EXPORT int apdm_ctx_re_enable_accesspoint_wireless (apdm_ctx_t *context*)

After wireless has been disabled on an AP using the apdm_ctx_disable_accesspoint_-wireless() function, in can be re-enabled using this function

Parameters

context	

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.44 APDM_EXPORT int apdm_ctx_reset_num_samples_from_ap (apdm_ctx_t *context*)

Used to reset the counter which tracks the number of samples received from the access point by the host libraries.

Parameters

context	The context

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.45 APDM_EXPORT int apdm_ctx_restore_context_from_disk (apdm_ctx_t *context,* const char * *filepath*)

Restore a context from disk to memory thus allowing you to start streaming without re-configuring the system. Make sure to re-sync the record head list prior to streaming data.

This functionality will only work if identical access points are used, and all of the original monitors are in use.

Parameters

context	Destination context into which to restore context information.
filepath	The file to restore the context from.

Returns

APDM_OK on success, APDM_UNEXPECTED_STRUCTURE_VALUE if the version of the libraries that saved the file is different then the version of libraries that is restoring the context, error code otherwise.

References apdm_ap_connect(), apdm_ap_disconnect(), apdm_ap_get_id(), apdm_ap_init_handle(), apdm_ctx_allocate_new_context(), apdm_ctx_disconnect(), apdm_ctx_free_context(), apdm_log_context(), apdm_log_error(), and apdm_log_info().

5.2.1.46 APDM_EXPORT enum APDM_Status apdm_ctx_set_error_handling_mode (apdm_ctx_t *context,* enum APDMErrorHandlingBehavior *new_mode*)

Sets the error handling behavior of the underlying APDM libraries. Particularly affects when errors, partial records or full records are returned from a call to getting a record list.

Parameters

context	The apdm context
newMode	An enum APDMErrorHandlingBehavior indicating what error handling mode to set.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.47 APDM_EXPORT int apdm_ctx_set_max_sample_delay_seconds (apdm_ctx_t *context,* const uint16_t *max_data_delay_seconds*)

This function sets the maximum amount of delay allowable for data returned from the host libraries. The default is 5ms. The max is 15 minutes.

Parameters

context	The apdm handle to check data on
max_data_delay_seconds	The maximum age of returned packets from the library, set APDM_INFINITE_MAX_LATENCY for infinity

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References adpm_ap_set_max_latency_value_seconds(), apdm_ctx_get_expected_-number_of_sensors2(), and apdm_log_error().

5.2.1.48 APDM_EXPORT int apdm_ctx_set_metadata_string (apdm_ctx_t *context,* const uint32_t *device_id,* const char * *str*)

Metadata can be stored in the context with respect to a given device id, and later retrieved.

Parameters

context	The apdm handle
device_id	The device ID for which you want the metadata string (this is different then the Case ID on the back of the monitor).
str	The char* type meta data to be stored, maximum length is USER_META_DATA_STRING_SIZE(64)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.49 APDM_EXPORT int apdm_ctx_set_metadeta_uint32 (apdm_ctx_t *context,* const uint32_t *device_id,* const uint32_t *value*)

Metadata can be stored in the context with respect to a given device id, and later retrieved.

Parameters

context	The apdm handle
device_id	The device ID to set the meta data for (this is different then the Case ID on the back of the monitor).
value	The uint32_t type meta data to be stored. You can optionally associate meta-data with a device ID in a context. It's an arbitrary number for which the meaning is defined by the application using the library. E.G. You could use this to specify what limb of a persons body each motion monitor is attached to.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.50 APDM_EXPORT int apdm_ctx_set_orientation_model (apdm_ctx_t *context,* const apdm_orientation_model_t *orientation_model*)

This function sets the orientation model used for computing orientation estimates. The default is APDM_ORIENTATION_MODEL_ALL if all sensors are enabled, or APDM_ORIENTATION_MODEL_NO_MAG if the magnetometer is disabled.

Parameters

context	The apdm handle to set the orientation model on
orientation_model	The orientation model to use: APDM_ORIENTATION_MODEL_ALL, APDM_ORIENTATION_MODEL_UNDISTURBED_MAG, APDM_ORIENTATION_MODEL_NO_MAG

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.51 APDM_EXPORT int apdm_ctx_set_sensor_compensation_data (apdm_ctx_t *context,* const apdm_sensor_compensation_t * *src_comp_data,* const int32_t *sensor_index*)

Parameters

context	The apdm context
*src_comp_data	The source of compensation data which to store into the context for the sensor of index sensor_index.
sensor_index	The index of the sensor for which you want to retrieve compensation data.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ctx_get_expected_number_of_sensors2(), and apdm_log_error().

5.2.1.52 APDM_EXPORT int apdm_ctx_sync_record_list_head (apdm_ctx_t *context*)

This function will drop all data stored in the library correlation FIFOs and start reading data from the attached access points until it is able to get a full set of data from all

sensors with the same sync value.

This function may return failure if one (or more) monitors is catching up it's data stream, you should wait until the AP's are blinking green. If one or more AP's is blinking green-red this function will likely fail.

It will also reset the total omitted samples counter.

Parameters

context	

Returns

APDM_OK if successful, APDM_UNABLE_TO_SYNC_RECORD_HEAD_LIST_ERROR if it can't sync the list due to lack of data (usually because motion monitors are still docked, or out of range of the access point), error code otherwise

References apdm_ctx_get_expected_number_of_sensors2(), apdm_get_time_ms_64(), apdm_log_context(), apdm_log_debug(), apdm_log_error(), and apdm_usleep().

5.2.1.53 APDM_EXPORT int apdm_get_max_sample_delay_seconds (apdm_ctx_t *context*, uint16_t * *dest*)

Gets the current max-sample-delay setting, in seconds (aka max latency)

Parameters

context	The apdm handle to check data on
**dest*	Destination into which the setting should be stored

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.2.1.54 APDM_EXPORT int apdm_get_num_samples_from_ap (apdm_ctx_t *context*)

Returns the number of sensor samples that have been transfered from all the attached AP's to the host. This number includes all samples that are currently in the correlation FIFO's of the library. It is useful if you want to verify data transfer from device->access point->host, but where configured library processing policys might be causing delays in data returned by the libraries.

Parameters

context	The context

Returns

If zero of positive, the number of samples that have been transfered from the A-P to the host libraries since the last time aptm_reset_num_samples_from_ap was called, negative error code otherwise.

5.3 AccessPoint

Functions

- APDM_EXPORT int apdm_ap_get_num_access_points_on_host1 (uint32_t *dest)
- APDM_EXPORT int apdm_ap_connect (apdm_ap_handle_t ap_handle, const int indexNumber)
- APDM_EXPORT int apdm_ap_disconnect (apdm_ap_handle_t ap_handle)
- APDM_EXPORT int apdm_ap_init_handle (apdm_ap_handle_t ap_handle)
- APDM_EXPORT int apdm_ap_get_monitor_latency (apdm_ap_handle_t ap_handle, const uint32_t monitor_id, int64_t *dest)
- APDM_EXPORT int apdm_ap_set_warning_blink_threshold (apdm_ap_handle_t ap_handle, const uint32_t delta_threshold)
- APDM_EXPORT int apdm_ap_set_error_blink_threshold (apdm_ap_handle_t ap_handle, const uint32_t delta_threshold)
- APDM_EXPORT int apdm_ap_get_wireless_streaming_led_status (apdm_ap_handle_t ap_handle, uint32_t *dest)
- APDM_EXPORT const char * apdm_ap_wireless_streaming_status_t_str (const apdm_ap_wireless_streaming_status_t streaming_status)
- APDM_EXPORT int adpm_ap_set_max_latency_value_seconds (apdm_ap_handle_t ap_handle, const uint16_t max_latency_seconds)
- APDM_EXPORT int adpm_ap_set_minimum_sync_value (apdm_ap_handle_t ap_handle, const uint64_t minimum_sync_value)
- APDM_EXPORT int adpm_ap_get_minimum_sync_value (apdm_ap_handle_t ap_handle, uint64_t *minimum_sync_value)
- APDM_DEPRECATED APDM_EXPORT int apdm_ap_get_gpio_value (apdm_ap_handle_t ap_handle, const apdm_ap_gpio_pin_t gpio_pin, bool *output_value)
- APDM_EXPORT int apdm_ap_get_io_value (apdm_ap_handle_t ap_handle, const int gpio_pin, uint32_t *output_value)
- APDM_DEPRECATED APDM_EXPORT int apdm_ap_set_gpio_value (apdm_ap_handle_t ap_handle, const apdm_ap_gpio_pin_t gpio_pin, const bool output_value)
- APDM_EXPORT int apdm_ap_set_io_value (apdm_ap_handle_t ap_handle, const int gpio_pin, const uint32_t output_value)
- APDM_EXPORT int apdm_send_accesspoint_cmd (apdm_ap_handle_t ap_handle, const char *cmdToSend, char *BYTE_ARRAY, const uint32_t outputBufferLength, const uint32_t numLinesToRead, const uint32_t timeoutMilliseconds)
- APDM_EXPORT int apdm_ap_get_version_string (apdm_ap_handle_t ap_handle, char *BYTE_ARRAY, const int destLength)
- APDM_EXPORT int apdm_ap_get_version (apdm_ap_handle_t ap_handle, uint64_t *dest)

- APDM_EXPORT int apdm_ap_get_board_version_string (apdm_ap_handle_t ap_handle, char *BYTE_ARRAY, const int destLength)
- APDM_EXPORT int apdm_ap_get_id_and_board_version (apdm_ap_handle_t ap_handle, uint32_t *dest_id, uint32_t *dest_board_version)
- APDM_EXPORT int apdm_ap_verify_supported_version (apdm_ap_handle_t ap_handle)
- APDM_EXPORT int apdm_ap_override_minimum_supported_version (const uint64_t new_version)
- APDM_EXPORT int apdm_ap_get_id (apdm_ap_handle_t ap_handle, uint32_t *dest)
- APDM_EXPORT int apdm_ap_get_case_id (apdm_ap_handle_t ap_handle, char *BYTE_ARRAY, const int dest_buffer_length)
- APDM_EXPORT int apdm_ap_reset_into_bootloader (apdm_ap_handle_t ap_handle)
- APDM_EXPORT int apdm_ap_reset_into_firmware (apdm_ap_handle_t ap_handle)
- APDM_EXPORT int apdm_free_ap_handle (apdm_ap_handle_t ap_handle)
- APDM_EXPORT apdm_ap_handle_t apdm_ap_allocate_handle (void)
- APDM_EXPORT int apdm_ap_get_mode (apdm_ap_handle_t ap_handle)
- APDM_EXPORT int apdm_ap_get_protocol_subversion (apdm_ap_handle_t ap_handle, int64_t *dest_protocol_subversion)
- APDM_EXPORT int apdm_configure_accesspoint (apdm_ap_handle_t ap_handle, const uint8_t radio1_pipe_count, const uint8_t radio2_pipe_count)
- APDM_EXPORT int apdm_ctx_get_all_ap_debug_info (apdm_ctx_t context)
- APDM_DEPRECATED APDM_EXPORT int adpm_ap_set_max_latency_value (apdm_ap_handle_t ap_handle, const uint32_t max_latency_ms)

5.3.1 Function Documentation

5.3.1.1 APDM_EXPORT int adpm_ap_get_minimum_sync_value (apdm_ap_handle_t *ap_handle*, uint64_t * *minimum_sync_value*)

Parameters

ap_handle	The AP handle for which this value is to be set.
**minimum_sync_value*	This will be populated with the miniumum sync values that sensors are supposed to be sending out.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.3.1.2 APDM_DEPRECATED APDM_EXPORT int adpm_ap_set_max_latency_value (apdm_ap_handle_t *ap_handle,* const uint32_t *max_latency_ms*)

Sets the maximum latency of packets that should be coming from devices to the access point. If set to zero, greater then (1000 ∗ 65535 ∗ 24 / 128)=12287812.5ms then not latency constraint will be applied by the device. Default is 15,000ms, max is 60,000ms.

Deprecated This has been replaced by adpm_ap_set_max_latency_value_seconds(). This function will be removed after Jan 2011.

Parameters

ap_handle	The AP handle for which this value is to be set.
max_latency_ms	The maximum delay, in mS, which a device should send buffered packets to the AP.

Returns

APDM_OK on success, error code otherwise.

References adpm_ap_set_max_latency_value_seconds().

5.3.1.3 APDM_EXPORT int adpm_ap_set_max_latency_value_seconds (apdm_ap_handle_t *ap_handle,* const uint16_t *max_latency_seconds*)

Sets the maximum latency of packets that should be coming from devices to the access point.

Parameters

ap_handle	The AP handle for which this value is to be set.
max_latency_seconds	The maximum delay, in seconds, which a device should send buffered packets to the AP. A value of APDM_INFINITE_MAX_LATENCY implies infinity.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by adpm_ap_set_max_latency_value(), and apdm_ctx_set_max_sample_delay_seconds().

5.3.1.4 APDM_EXPORT int adpm_ap_set_minimum_sync_value (apdm_ap_handle_t *ap_handle,* const uint64_t *minimum_sync_value*)

Parameters

ap_handle	The AP handle for which this value is to be set.
minimum_sync_value	The minimum sync value that you want sensors to send out. This is useful for skipping ahead in a data stream.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

Referenced by apdm_ctx_set_minimum_sync_value().

5.3.1.5 APDM_EXPORT apdm_ap_handle_t apdm_ap_allocate_handle (void)

Allocates memory for an access point handle.

Returns

NULL on failure, non-NULL on success.

5.3.1.6 APDM_EXPORT int apdm_ap_connect (apdm_ap_handle_t *ap_handle,* const int *indexNumber*)

Used to connect to an access point.

Parameters

ap_handle	An un-configured access point handle.
indexNumber	The index number, starting at zero, of the AP on the host to which to connect.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_get_protocol_subversion(), apdm_ap_get_version(), apdm_ap_get_version_string(), apdm_get_time_ms_64(), apdm_log_debug(), apdm_log_warning(), and apdm_strerror().

Referenced by apdm_ctx_open_all_access_points(), and apdm_ctx_restore_context_from_disk().

5.3.1.7 APDM_EXPORT int apdm_ap_disconnect (apdm_ap_handle_t *ap_handle*)

Disconnects the access point handle from the underlying OS binding

Parameters

ap_handle	The handle to be disconnected.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ctx_disconnect(), apdm_ctx_open_all_access_points(), and apdm_ctx_restore_context_from_disk().

5.3.1.8 APDM_EXPORT int apdm_ap_get_board_version_string (apdm_ap_handle_t *ap_handle,* char * *BYTE_ARRAY,* const int *destLength*)

Returns the board hardware version string from the access point.

Parameters

ap_handle	The handle with respect to what version string you want.
**BYTE_ARRAY*	The destination buffer into which you want the version string copied into, must not be NULL.
destLength	The maximum length of the destination string, must be >0

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_send_accesspoint_cmd().

5.3.1.9 APDM_EXPORT int apdm_ap_get_case_id (apdm_ap_handle_t *ap_handle,* char * *BYTE_ARRAY,* const int *dest_buffer_length*)

Retrieves the case ID of the AP.

Parameters

ap_handle	The AP handle
**BYTE_ARRAY*	The destination buffer into which to store the case ID string.
dest_buffer_length	The max length of the destination buffer

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error(), and apdm_log_warning().

Referenced by apdm_ctx_open_all_access_points().

5.3.1.10 APDM_DEPRECATED APDM_EXPORT int apdm_ap_get_gpio_value (apdm_ap_handle_t *ap_handle,* const apdm_ap_gpio_pin_t *gpio_pin,* bool * *output_value*)

Parameters

ap_handle	The AP handle.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
**output_value*	Destination into which to store the current value of the GPIO pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Deprecated replaced with apdm_ap_get_io_value() for more general purpose IO features. Will be removed after May 2013.

References apdm_ap_get_io_value().

5.3.1.11 APDM_EXPORT int apdm_ap_get_id (apdm_ap_handle_t *ap_handle,* uint32_t * *dest*)

This returns the serial number of the access point

Parameters

ap_handle	The AP handle associated with the AP for which you want the serial number.
**dest*	The destination into which the serial number is to be stored.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ap_get_id_and_board_version(), apdm_log_debug(), and apdm_send_accesspoint_cmd().

Referenced by apdm_ctx_restore_context_from_disk().

5.3.1.12 APDM_EXPORT int apdm_ap_get_id_and_board_version (apdm_ap_handle_t *ap_handle*, uint32_t * *dest_id*, uint32_t * *dest_board_version*)

Parameters

*ap_handle	The access point handle
*dest_id	The destination into which to store the ID of the access point
*dest_board_version	The destination into which to store the printed circuit board version

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ap_get_id(), and apdm_ctx_open_all_access_points().

5.3.1.13 APDM_EXPORT int apdm_ap_get_io_value (apdm_ap_handle_t *ap_handle*, const int *gpio_pin*, uint32_t * *output_value*)

Parameters

ap_handle	The AP handle.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
*output_value	Destination into which to store the current value of the GPIO pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

Referenced by apdm_ap_get_gpio_value(), and apdm_ctx_ap_get_io_value().

5.3.1.14 APDM_EXPORT int apdm_ap_get_mode (apdm_ap_handle_t *ap_handle*)

Parameters

∗*ap_handle*	A handle that is already connected to an AP via usb.

Returns

APM_FIRMWARE if the AP is in firmware mode, APM_BOOTLOADER is it's in bootloader, APM_UNKNOWN if the mode cannot be determined,

References apdm_log_error(), and apdm_send_accesspoint_cmd().

5.3.1.15 APDM_EXPORT int apdm_ap_get_monitor_latency (apdm_ap_handle_t *ap_handle,* const uint32_t *monitor_id,* int64_t ∗ *dest*)

Retrieves the latency of an individual monitor from the given AP.

Parameters

ap_handle	The ap handle which is to be queried.
monitor_id	The ID of the monitor for which you want to know the latency.
∗*dest*	The destination into which to store the latency value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

Referenced by apdm_ctx_get_monitor_latency().

5.3.1.16 APDM_EXPORT int apdm_ap_get_num_access_points_on_host1 (uint32_t ∗ *dest*)

Parameters

∗*dest*	Destination into which to store the number of USB access points attached to the host based on the VID/PID listing from the OS.

Returns

APDM_OK on success, error code otherwise

Referenced by apdm_ctx_open_all_access_points().

5.3.1.17 APDM_EXPORT int apdm_ap_get_protocol_subversion (apdm_ap_handle_t *ap_handle*, int64_t * *dest_protocol_subversion*)

Parameters

ap_handle	The AP handle
**dest_protocol_subversion*	The destination into which to store the protocol version

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ap_connect().

5.3.1.18 APDM_EXPORT int apdm_ap_get_version (apdm_ap_handle_t *ap_handle*, uint64_t * *dest*)

Parameters

ap_handle	The access point handle for which you want the numeric representation of the version
**dest*	The destination into which to store the ap firmware version number

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ap_connect(), and apdm_ap_verify_supported_version().

5.3.1.19 APDM_EXPORT int apdm_ap_get_version_string (apdm_ap_handle_t *ap_handle*, char * *BYTE_ARRAY*, const int *destLength*)

Returns the firmware version string from the access point.

Parameters

ap_handle	The handle with respect to what version string you want.
**BYTE_ARRAY*	The destination buffer into which you want the version string copied into, must not be NULL.
destLength	The maximum length of the destination string, must be greater then zero.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_get_time_ms_64(), apdm_log_debug(), and apdm_send_accesspoint_cmd().

Referenced by apdm_ap_connect(), and apdm_ap_verify_supported_version().

5.3.1.20 APDM_EXPORT int apdm_ap_get_wireless_streaming_led_status (apdm_ap_handle_t *ap_handle,* uint32_t * *dest*)

Parameters

ap_handle	The AP handle
**dest*	The destination into which to store the LED status, of type apdm_ap_wireless_streaming_status_t

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.3.1.21 APDM_EXPORT int apdm_ap_init_handle (apdm_ap_handle_t *ap_handle*)

Initializes an access point handle

Parameters

ap_handle	Pointer to the handle to be initialized

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ctx_initialize_context(), and apdm_ctx_restore_context_from_disk().

5.3.1.22 APDM_EXPORT int apdm_ap_override_minimum_supported_version (const uint64_t *new_version*)

Allows you to override the minimum access point station version number used to validate AP versions.

Parameters

new_version	Version number, e.g. 20100902170629 Set this to zero to use library default version number.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.3.1.23 APDM_EXPORT int apdm_ap_reset_into_bootloader (apdm_ap_handle_t *ap_handle*)

This function will reset the given access point into bootloader

Parameters

∗*ap_handle*	A handle that is already connected to an AP via usb.

Returns

APDM_OK on success, and if successful, the handle will have been DISCONNECTED and you must re-connect to the AP.

5.3.1.24 APDM_EXPORT int apdm_ap_reset_into_firmware (apdm_ap_handle_t *ap_handle*)

This function will reset the given access point into firmware

Parameters

∗*ap_handle*	A handle that is already connected to an AP via usb.

Returns

APDM_OK on success, and if successful, the handle will have been DISCONNECTED and you must re-connect to the AP.

5.3.1.25 APDM_EXPORT int apdm_ap_set_error_blink_threshold (apdm_ap_handle_t *ap_handle*, const uint32_t *delta_threshold*)

Parameters

ap_handle	The access point handle
delta_threshold	The threshold, in milliseconds, for the AP to start blinking green/red if one (or more) monitors are falling behind in their transmission (e.g. out of range).

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.3.1.26 APDM_DEPRECATED APDM_EXPORT int apdm_ap_set_gpio_value (apdm_ap_handle_t *ap_handle,* const apdm_ap_gpio_pin_t *gpio_pin,* const bool *output_value*)

Parameters

ap_handle	The AP handle.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.
output_value	New value to set on a GPIO pin that has been configured as an output pin.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Deprecated replaced with apdm_ap_set_io_value() for more general purpose IO features. Will be removed after May 2013.

References apdm_ap_set_io_value().

5.3.1.27 APDM_EXPORT int apdm_ap_set_io_value (apdm_ap_handle_t *ap_handle,* const int *gpio_pin,* const uint32_t *output_value*)

Parameters

ap_handle	The AP handle.
gpio_pin	The pin in question (see the apdm_ap_gpio_pin_t enum in apdm_types.h). Use APDM_AP_GPIO_0 to control the digital input or output pins on the DIN-6 connector. Use APDM_AP_ANALOG_OUT_0 or APDM_AP_ANALOG_IN_0 to control or read the analog input/output pins on the DIN-4 connector.

output_value	New value to set on a GPIO pin that has been configured as an output pin. When setting an analog output value, this will be a number between 0 and 1023, that gets set on the DAC and puts out between 0 and 5 volts.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

Referenced by apdm_ap_set_gpio_value(), and apdm_ctx_ap_set_io_value().

5.3.1.28 APDM_EXPORT int apdm_ap_set_warning_blink_threshold (apdm_ap_handle_t *ap_handle*, const uint32_t *delta_threshold*)

Parameters

ap_handle	The access point handle
delta_threshold	The threshold, in milliseconds, for the AP to start blinking green/blue if one (or more) monitors are falling behind in their transmission (e.g. out of range).

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.3.1.29 APDM_EXPORT int apdm_ap_verify_supported_version (apdm_ap_handle_t *ap_handle*)

this function is used to verify that the given access point has a version of firmware that is supported by the libraries.

Parameters

ap_handle	The access point handle

Returns

APDM_OK if the version is OK, respective error code otherwise.

References apdm_ap_get_version(), apdm_ap_get_version_string(), and apdm_log_error().

5.3.1.30 APDM_EXPORT const char* apdm_ap_wireless_streaming_status_t_str (const apdm_ap_wireless_streaming_status_t *streaming_status*)

Parameters

streaming_status	Streaming status, of type apdm_ap_wireless_streaming_status_t, for which you want the string representation.

Returns

Pointer to a string for the given status type

5.3.1.31 APDM_EXPORT int apdm_configure_accesspoint (apdm_ap_handle_t *ap_handle,* const uint8_t *radio1_pipe_count,* const uint8_t *radio2_pipe_count*)

Internal function to configure a single access point with a given pipe count and assign wireless channels based on whats configured in the AP handle data structure.

Parameters

ap_handle	The handle for the AP to be configured.
radio1_pipe_count	The number of pipes that will be used for radio 1 of the AP (first three devices usually)
radio2_pipe_count	The number of pipes that will be used for radio 2 of the AP (first three devices usually)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_init_access_point_wireless(), apdm_log_error(), and apdm_log_info().

5.3.1.32 APDM_EXPORT int apdm_ctx_get_all_ap_debug_info (apdm_ctx_t *context*)

The access point tracks some internal debugging stats and numbers. This function will retrieve those debugging statistic and print to the debug logging subsystem.

Parameters

context	

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.3.1.33 APDM_EXPORT int apdm_free_ap_handle (apdm_ap_handle_t *ap_handle*)

Frees memory for the given access point handle

Parameters

ap_handle	The handle to be freed

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_warning().

5.3.1.34 APDM_EXPORT int apdm_send_accesspoint_cmd (apdm_ap_handle_t *ap_handle,* const char * *cmdToSend,* char * *BYTE_ARRAY,* const uint32_t *outputBufferLength,* const uint32_t *numLinesToRead,* const uint32_t *timeoutMilliseconds*)

Sends a string-command to the access point, mostly used for debugging and non-standard functionality.

Parameters

ap_handle	The handle for the AP to which the command is to be sent
*cmdToSend	The command to send
*BYTE_ARRAY	The destination into which the response from the AP is to be placed.
*outputBufferLength	The length of the outputStringBuffer.
*numLinesToRead	The number of lines that are expected in response to the command being sent.
*timeoutMilliseconds	Maximum time length to wait for the response.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_get_time_ms_64(), apdm_log_debug(), and apdm_log_error().

Referenced by apdm_ap_get_board_version_string(), apdm_ap_get_id(), apdm_ap_get_mode(), and apdm_ap_get_version_string().

5.4 DataFiles

Functions

- APDM_EXPORT int apdm_read_raw_file_info (const char *filename, apdm_recording_info_t *recording_info)
- APDM_EXPORT int apdm_process_raw (char **file_in, char **calibration_file, int nFiles, const char *file_out, const bool store_raw, const bool store_si, const bool format_hdf, const bool compress, char csv_delimiter, apdm_progress_t *progress)
- APDM_EXPORT int apdm_process_raw2 (char **file_in, char **calibration_file, int nFiles, const char *file_out, const bool store_raw, const bool store_si, const bool store_filtered, const bool format_hdf, const bool compress, char csv_delimiter, apdm_progress_t *progress)
- APDM_EXPORT int apdm_process_raw3 (apdm_file_conversion_parameter_t *params)
- APDM_EXPORT int apdm_convert_h5_to_csv (char *h5file, char *csvfile, char delimiter)
- APDM_EXPORT int apdm_release_conversion_parameters (apdm_file_conversion_parameter_t *params)
- APDM_EXPORT int apdm_find_first_and_last_common_samples (char **files_in, int nFiles, uint64_t *first_sample, uint64_t *last_sample, int use_sync_lock)
- APDM_EXPORT int apdm_find_button_transition (const char *file, apdm_recording_info_t *recording_info, uint64_t *time_of_button_transition, uint32_t *start_block, uint8_t mode)
- APDM_EXPORT int apdm_initialize_file_conversion_parameters (apdm_file_conversion_parameter_t *params)
- APDM_EXPORT hid_t apdm_create_file_hdf (const char *filename, apdm_device_info_t *device_info, int nMonitors)
- APDM_EXPORT int apdm_close_file_hdf (hid_t file)
- APDM_EXPORT apdm_csv_t apdm_create_file_csv (char *filename)
- APDM_EXPORT int apdm_close_file_csv (apdm_csv_t file_handle)
- APDM_EXPORT int apdm_write_record_hdf (hid_t file, apdm_device_info_t *info, apdm_record_t *records, int sampleNumber, int nDevices, bool store_raw, bool store_si, bool compress)
- APDM_EXPORT int apdm_write_record_hdf2 (hid_t file, apdm_device_info_t *info, apdm_record_t *records, int sampleNumber, int nDevices, bool store_raw, bool store_si, bool store_filtered, bool compress)
- APDM_EXPORT int apdm_write_record_csv (apdm_csv_t file, apdm_device_info_t *info, apdm_record_t *records, int sampleNumber, int nDevices, bool store_raw, bool store_si, char delimiter)
- APDM_EXPORT int apdm_write_annotation (hid_t file, apdm_annotation_t *annotation)

- APDM_EXPORT int apdm_read_hdf_dataset (const char *file, char *monitor_id, const char *datasetName, double *data, int ndims, const int *start_index, const int *shape, const int *strideLength)
- APDM_EXPORT int apdm_read_hdf_timestamps (char *file, char *monitor_id, char *datasetName, uint64_t *data, int start_index, int nSamples, int stride-Length)
- APDM_EXPORT int apdm_get_hdf_dataset_shape (char *file, char *monitor_id, char *datasetName, int *shape, int *ndims)
- APDM_EXPORT int apdm_read_hdf_calibration_data (char *file, char *case_id, apdm_sensor_compensation_t *sensor_comp)
- APDM_EXPORT int apdm_get_hdf_device_list (char *file, char **monitor_ids, int *nDevices)
- APDM_EXPORT int apdm_get_hdf_device_list_swig (char *file, apdm_case_id-_t *monitor_ids, int *nDevices)
- APDM_EXPORT int apdm_get_hdf_label_list (char *file, char **monitor_labels, int *nDevices)
- APDM_EXPORT int apdm_get_hdf_label_list_swig (char *file, apdm_monitor_-label_t *monitor_labels, int *nDevices)

5.4.1 Function Documentation

5.4.1.1 APDM_EXPORT int apdm_close_file_csv (apdm_csv_t *file_handle*)

Close a file opened with apdm_create_file_csv.

Parameters

file_handle	The file handle returned from apdm_create_file_csv

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_convert_h5_to_csv().

5.4.1.2 APDM_EXPORT int apdm_close_file_hdf (hid_t *file*)

Closes a file previously opened with apdm_create_file_hdf.

Parameters

file	The HDF5 file handle returned from apdm_create_file_hd

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_process_raw3().

5.4.1.3 APDM_EXPORT int apdm_convert_h5_to_csv (char * *h5file*, char * *csvfile*, char *delimiter*)

Reads a .h5 file, and converts it to a .csv file.

Parameters

h5file	The path to the h5 file to convert
csvfile	The path to the csv file to create
delimiter	Delimiter to use between columns in the csv file

Returns

APDM_OK on success, error code otherwise

References apdm_record_t::accl_isPopulated, apdm_record_t::accl_x_axis, apdm_record_t::accl_y_axis, apdm_record_t::accl_y_axis_si, apdm_record_t::accl_z_axis, apdm_record_t::accl_z_axis_si, apdm_close_file_csv(), apdm_create_file_csv(), apdm_get_hdf_dataset_shape(), apdm_get_hdf_device_list(), apdm_get_hdf_label_list(), apdm_log_debug(), apdm_log_error(), apdm_read_hdf_timestamps(), apdm_write_record_csv(), apdm_record_t::button_status, apdm_device_info_t::decimation_factor, apdm_record_t::gyro_isPopulated, apdm_record_t::gyro_x_axis, apdm_record_t::gyro_x_axis_si, apdm_record_t::gyro_y_axis, apdm_record_t::gyro_y_axis_si, apdm_record_t::gyro_z_axis, apdm_record_t::gyro_z_axis_si, apdm_record_t::mag_common_axis, apdm_record_t::mag_isPopulated, apdm_record_t::mag_x_axis, apdm_record_t::mag_x_axis_si, apdm_record_t::mag_y_axis, apdm_record_t::mag_y_axis_si, apdm_record_t::mag_z_axis, and apdm_record_t::mag_z_axis_si.

5.4.1.4 APDM_EXPORT apdm_csv_t apdm_create_file_csv (char * *filename*)

Creates a new file and returns a file pointer.

Parameters

filename	Name of the file to create

Returns

apdm_csv_t (actually a FILE ∗) file handle for the new file

Referenced by apdm_convert_h5_to_csv().

5.4.1.5 APDM_EXPORT hid_t apdm_create_file_hdf (const char ∗ *filename*, apdm_device_info_t ∗ *device_info*, int *nMonitors*)

Creates a new APDM HDF5 file and opens it for writing. Used with apdm_write_record_hdf, apdm_write_annotation, and apdm_close_file_hdf to stream to a data file.

Parameters

filename	The filename to create. Should have the ".h5" extension.
device_info	The configuration information for each monitor. Used to write various metadata. This should be a pointer to an array of apdm_device_info_t structures.
nMonitors	The number of monitors in the device_info array.

Returns

hid_t HDF5 file handle, always > 0 on success.

Referenced by apdm_process_raw3().

5.4.1.6 APDM_EXPORT int apdm_find_button_transition (const char ∗ *file*, apdm_recording_info_t ∗ *recording_info*, uint64_t ∗ *time_of_button_transition*, uint32_t ∗ *start_block*, uint8_t *mode*)

Iteratively finds button transitions (0->1) within a raw .apdm input file.

Parameters

file	A string indicating the full file path to a .apdm file.
recording_info	The recording_info structure for the file. This is populated by first calling the apdm_read_file_info() method.
time_of_button_transition	An In-Out parameter indicating the start sync_val to look for (In) and the sync_val of the found transition. If no transition is found, 0 is returned. Should be '1' the first iteration of the function call.
start_block	An In-Out parameter indicating the start block to look in (In) and the last block searched (out)
mode	1: Look for button presses, 0: Look for button releases

Returns

APDM_OK on success, error code otherwise

5.4.1.7 APDM_EXPORT int apdm_find_first_and_last_common_samples (char ** *files_in,* int *nFiles,* uint64_t * *first_sample,* uint64_t * *last_sample,* int *use_sync_lock*)

Finds the first and last common sample within the raw apdm files passed in.

Parameters

file_in	An array of .apdm input file names from unique monitors.
nFiles	The number of input files present in file_in.
first	Output parameter indicating the first common sample. Default = 1
sample	Output parameter indicating the last common sample. Default = 1e15
use_sync_lock	Input parameter indicating whether sync_lock bit should be used to determine start and stop times

Returns

APDM_OK on success, error code otherwise

References apdm_log_debug().

5.4.1.8 APDM_EXPORT int apdm_get_hdf_dataset_shape (char * *file,* char * *monitor_id,* char * *datasetName,* int * *shape,* int * *ndims*)

Helper function for working with HDF5 files. Gets the size of a specified dataset.

Parameters

file	The .h5 file to inspect.
monitor_id	The group name for the monitor (returned by apdm_get_hdf_device_list). For v1 files, this is the 'Opal_xx' where xx is the monitor id. For v2 files, this is the case id.
datasetName	The name of the dataset to inspect. Must be "Accelerometers", "Gyroscopes", "Magnetometers", "Temperature",)
shape	Output array containing the size in each dimension of the datset. If shape is NULL, then only ndims will be set.
ndims	Output containing the number of dimensions in the dataset.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_convert_h5_to_csv(), and apdm_recalibrate_gyroscopes_from_h5().

5.4.1.9 APDM_EXPORT int apdm_get_hdf_device_list (char * *file*, char ** *monitor_ids*, int * *nDevices*)

Helper function for working with HDF5 files. Gets the list of monitor IDs stored in the file.

Parameters

file	The .h5 file to inspect.
***monitor_ids*	Output array containing the group name for each monitor in the file. If NULL, only nDevices will be set.
**nDevices*	Output number of monitors in the file.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_convert_h5_to_csv(), apdm_get_hdf_device_list_swig(), and apdm_recalibrate_gyroscopes_from_h5().

5.4.1.10 APDM_EXPORT int apdm_get_hdf_device_list_swig (char * *file*, apdm_case_id_t * *monitor_ids*, int * *nDevices*)

Helper function for working with HDF5 files that is compatible with SWIG based Java bindings. Gets the list of monitor IDs stored in the file.

Parameters

file	The .h5 file to inspect.
**monitor_ids*	Output array containing the group name for each monitor in the file. If NULL, only nDevices will be set.
**nDevices*	Output number of monitors in the file.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_get_hdf_device_list().

5.4.1.11 APDM_EXPORT int apdm_get_hdf_label_list (char * *file*, char ** *monitor_labels*, int * *nDevices*)

Helper function for working with HDF5 files. Gets the list of monitor labels stored in the file. This list is in the same order as the list returned by apdm_get_hdf_device_list().

Parameters

file	The .h5 file to inspect.
monitor_labels	Output array containing the user specified label for each monitor in the file. If NULL, only nDevices will be set.
nDevices	Output number of monitors in the file.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_convert_h5_to_csv(), and apdm_get_hdf_label_list_swig().

5.4.1.12 APDM_EXPORT int apdm_get_hdf_label_list_swig (char * *file*, apdm_monitor_label_t * *monitor_labels*, int * *nDevices*)

Helper function for working with HDF5 files that is compatible with SWIG based Java bindings. Gets the list of monitor labels stored in the file. This list is in the same order as the list returned by apdm_get_hdf_device_list().

Parameters

file	The .h5 file to inspect.
monitor_labels	Output array containing the user specified label for each monitor in the file. If NULL, only nDevices will be set.
nDevices	Output number of monitors in the file.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_get_hdf_label_list().

5.4.1.13 APDM_EXPORT int apdm_initialize_file_conversion_parameters (apdm_file_conversion_parameter_t * *params*)

Initializes an apdm_file_conversion_paraemter_t structure with default values. Further modifications can be made to the parameters

Parameters

params	The structure to initialize

Returns

APDM_OK on success, error code otherwise

References apdm_file_conversion_parameter_t::calibration_files, apdm_file_conversion_parameter_t::compress, apdm_file_conversion_parameter_t::csv_delimiter, apdm_file_conversion_parameter_t::dechop_raw_magnetometer, apdm_file_conversion_parameter_t::file_out, apdm_file_conversion_parameter_t::files_to_convert, apdm_file_conversion_parameter_t::format_hdf, apdm_file_conversion_parameter_t::nFiles, apdm_file_conversion_parameter_t::progress, apdm_file_conversion_parameter_t::store_filtered, apdm_file_conversion_parameter_t::store_raw, apdm_file_conversion_parameter_t::store_si, apdm_file_conversion_parameter_t::sync_end, apdm_file_conversion_parameter_t::sync_start, and apdm_file_conversion_parameter_t::timezone_string.

Referenced by apdm_process_raw(), and apdm_process_raw2().

5.4.1.14 APDM_EXPORT int apdm_process_raw (char ** *file_in*, char ** *calibration_file*, int *nFiles*, const char * *file_out*, const bool *store_raw*, const bool *store_si*, const bool *format_hdf*, const bool *compress*, char *csv_delimiter*, apdm_progress_t * *progress*)

Reads a .apdm file, and writes either raw, calibrated, or both, to either a .csv or a .h5 file.

Parameters

file_in	An array of .apdm input file names from an unique monitors.
calibration_file	array of optional files containing .hex calibration data for the monitors. If NULL, the calibration data in the .apdm file is used.
nFiles	The number of input files present in file_in.
file_out	The output filename (should end in .csv or .h5). If NULL, .csv format is forced and output is written to stdout.
store_raw	If true, raw data is stored.
store_si	If true, calibrated data (in SI units) is stored.
format_hdf	If true, file_out will be written to in HDF5 format. If false, file_out will be written to in .csv format. If multiple input files are present, HDF output must be selected.
csv_delimiter	Column delimiter to use if format_hdf is false (writing in csv format).
progress	If not null, this is an apdm_progress_t structure allocated by the caller and modified as this function runs.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_initialize_file_conversion_parameters(), apdm_process_raw3(), apdm_file_conversion_parameter_t::calibration_files, apdm_file_conversion_parameter_t::compress, apdm_file_conversion_parameter_t::csv_delimiter, apdm_file_conversion_parameter_t::file_out, apdm_file_conversion_parameter_t::files_to_convert, apdm_file_conversion_parameter_t::format_hdf, apdm_file_conversion_parameter_t::nFiles, apdm_file_conversion_parameter_t::progress, apdm_file_conversion_parameter_t::store_raw, apdm_file_conversion_parameter_t::store_si, and apdm_file_conversion_parameter_t::timezone_string.

5.4.1.15 APDM_EXPORT int apdm_process_raw2 (char ** *file_in*, char ** *calibration_file*, int *nFiles*, const char * *file_out*, const bool *store_raw*, const bool *store_si*, const bool *store_filtered*, const bool *format_hdf*, const bool *compress*, char *csv_delimiter*, apdm_progress_t * *progress*)

Reads a .apdm file, and writes either raw, calibrated, or both, to either a .csv or a .h5 file.

Parameters

file_in	An array of .apdm input file names from unique monitors.
calibration_file	array of optional files containing .hex calibration data for the monitors. If NULL, the calibration data in the .apdm file is used.
nFiles	The number of input files present in file_in.
file_out	The output filename (should end in .csv or .h5). If NULL, .csv format is forced and output is written to stdout.
store_raw	If true, raw data is stored.
store_si	If true, calibrated data (in SI units) is stored.
store_filtered	If true, filtered data (in SI units) is stored.
format_hdf	If true, file_out will be written to in HDF5 format. If false, file_out will be written to in .csv format. If multiple input files are present, HDF output must be selected.
compress	Compress data in HDF5 file output.
csv_delimiter	Column delimiter to use if format_hdf is false (writing in csv format).
progress	If not null, this is an apdm_progress_t structure allocated by the caller and modified as this function runs.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_initialize_file_conversion_parameters(), apdm_process_raw3(), apdm_file_conversion_parameter_t::calibration_files, apdm_file_conversion_parameter_t::compress, apdm_file_conversion_parameter_t::csv_delimiter, apdm_file_conversion_parameter_t::file_out, apdm_file_conversion_parameter_t::files_to_convert, apdm_file_conversion_parameter_t::format_hdf, apdm_file_conversion_parameter_t::nFiles, apdm_file_conversion_parameter_t::progress, apdm_file_conversion_parameter_t::store_raw, apdm_file_conversion_parameter_t::store_si, and apdm_file_conversion_parameter_t::timezone_string.

5.4.1.16 APDM_EXPORT int apdm_process_raw3 (apdm_file_conversion_parameter_t * *params*)

Reads a .apdm file, and writes either raw, calibrated, or both, to either a .csv or a .h5 file.

Parameters

params	An apdm_file_conversion_parameter_t struct containing parameters for file conversion. Should be initialized with apdm_initialize_file_conversion_parameters().

Returns

APDM_OK on success, error code otherwise

References apdm_close_file_hdf(), apdm_create_file_hdf(), apdm_log_debug(), apdm_log_error(), apdm_log_warning(), apdm_read_raw_file_info(), apdm_file_conversion_parameter_t::calibration_files, apdm_file_conversion_parameter_t::compress, apdm_file_conversion_parameter_t::csv_delimiter, apdm_file_conversion_parameter_t::dechop_raw_magnetometer, apdm_file_conversion_parameter_t::file_out, apdm_file_conversion_parameter_t::files_to_convert, apdm_file_conversion_parameter_t::format_hdf, apdm_file_conversion_parameter_t::nFiles, apdm_device_info_t::orientation_model, apdm_file_conversion_parameter_t::progress, apdm_file_conversion_parameter_t::store_filtered, apdm_file_conversion_parameter_t::store_raw, apdm_file_conversion_parameter_t::store_si, apdm_file_conversion_parameter_t::sync_end, apdm_file_conversion_parameter_t::sync_start, and apdm_file_conversion_parameter_t::timezone_string.

Referenced by apdm_process_raw(), and apdm_process_raw2().

5.4.1.17 APDM_EXPORT int apdm_read_hdf_calibration_data (char * *file*, char * *case_id*, apdm_sensor_compensation_t * *sensor_comp*)

Helper function for working with HDF5 files. Gets the calibration data from a file and converts it to an apdm_sensor_compensation_t struct.

Parameters

file	The .h5 file to read.
case_id	The Case ID string of the monitor to get the calibration data for.
sensor_comp	apdm_sensor_compensation_t structure to be populated with the calibration data.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug(), and apdm_log_warning().

Referenced by apdm_recalibrate_gyroscopes_from_h5().

5.4.1.18 APDM_EXPORT int apdm_read_hdf_dataset (const char * *file*, char * *monitor_id*, const char * *datasetName*, double * *data*, int *ndims*, const int * *start_index*, const int * *shape*, const int * *strideLength*)

Helper function for working with HDF5 files. Loads a segment of data from one of the datasets in one of the monitors stored in the .h5 file.

Parameters

file	The .h5 file to load data from
monitor_id	The group name for the monitor (returned by apdm_get_hdf_device_list). For v1 files, this is the 'Opal_xx' where xx is the monitor id. For v2 files, this is the case id.
datasetName	The name of the dataset to load. Must be ("Accelerometers", "Gyroscopes", "Magnetometers", or "Temperature");
data	Output array to load data into. This array is "flattened" so that the nth sample of the mth channel is at location data[n+m*N] where N is the total number of samples for each channel in the array.
ndims	Number of dimensions in the dataset to be read. Should always be 2.
start_index	Initial index to start reading from. First element is sample number, second element is channel number.
shape	Size of the output array (data). First element is the number of samples, second is the number of channels.
strideLength	Number of data points to skip by in each dimension.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

Referenced by apdm_recalibrate_gyroscopes_from_h5().

5.4.1.19 APDM_EXPORT int apdm_read_hdf_timestamps (char * *file*, char * *monitor_id*, char * *datasetName*, uint64_t * *data*, int *start_index*, int *nSamples*, int *strideLength*)

Helper function for working with HDF5 files. Loads a segment of data from one of the time datasets in one of the monitors stored in the .h5 file.

Parameters

file	The .h5 file to load data from
monitor_id	The group name for the monitor (returned by apdm_get_hdf_device_list). For v1 files, this is the 'Opal_xx' where xx is the monitor id. For v2 files, this is the case id.
datasetName	The name of the dataset to load. Must be "Time" or "SyncValue"
data	Output array to load data into.
start_index	Initial index to start reading from
nSamples	Size of the output array (data).
strideLength	Number of data points to skip by.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

Referenced by apdm_convert_h5_to_csv().

5.4.1.20 APDM_EXPORT int apdm_read_raw_file_info (const char * *filename*, apdm_recording_info_t * *recording_info*)

Reads metadata from a .apdm file and uses it to populate a apdm_recording_info_t structure.

Parameters

filename	The filename of the .apdm file
recording_info	A pointer to the file_info structure to populate with metadata.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug(), and apdm_log_error().

Referenced by apdm_process_raw3().

5.4.1.21 APDM_EXPORT int apdm_release_conversion_parameters (apdm_file_conversion_parameter_t * *params*)

Releases memory stored in the file_in_struct conversion parameter.

Parameters

params	An apdm_file_conversion_parameter_t struct containing parameters for file conversion.

Returns

APDM_OK on success, error code otherwise

References apdm_file_conversion_parameter_t::files_to_convert, and apdm_file_conversion_parameter_t::nFiles.

5.4.1.22 APDM_EXPORT int apdm_write_annotation (hid_t *file,* apdm_annotation_t * *annotation*)

Adds an annotation to the HDF5 file.

Parameters

file	The HDF5 file handle returned by apdm_create_file_hdf
annotation	An apdm_annotation_t struct containing a monitor ID, timestamp (epoch microseconds), and string (max 2048 characters).

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.4.1.23 APDM_EXPORT int apdm_write_record_csv (apdm_csv_t *file*, apdm_device_info_t * *info*, apdm_record_t * *records*, int *sampleNumber*, int *nDevices*, bool *store_raw*, bool *store_si*, char *delimiter*)

Write an array of records to a CSV file (previously created with apdm_create_file_csv). The sample number must be tracked by the caller, and incremented for each new sample.

Parameters

file	The file handle returned from apdm_create_file_csv
info	Array of apdm_device_info_t structs containing information about each monitor.
records	Array of apdm_record_t structs containing the data for one sample set of each monitor.
sampleNumber	It is important that the first time this function is called sampleNumber is 0. This can not be used as a row index. One row is created every time this function is called, reguardless of the value of sampleNumber. The meta data written in the first column depends on sampleNumber.
nDevices	Number of monitors in the info and records arrays.
store_raw	Flag indicating whether raw data should be stored. (True: yes, False: no)
store_si	Flag indicating whether SI data should be stored. (True: yes, False: no)
delimiter	Delimiter to use to separate columns.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_convert_h5_to_csv().

5.4.1.24 APDM_EXPORT int apdm_write_record_hdf (hid_t *file*, apdm_device_info_t * *info*, apdm_record_t * *records*, int *sampleNumber*, int *nDevices*, bool *store_raw*, bool *store_si*, bool *compress*)

Write an array of records to a HDF5 file (previously created with apdm_create_file_hdf). The sample number must be tracked by the caller, and incremented for each new sample. In case of dropped data, it may be desired to increment the sampleNumber for each dropped sample.

Parameters

file	The HDF5 file handle returned from apdm_create_file_hdf
info	Array of apdm_device_info_t structs containing information about each monitor.
records	Array of apdm_record_t structs containing the data for one sample set of each monitor.
sample-Number	An index into the arrays stored in the HDF5 file. It is important that the first time this function is called sampleNumber is 0.
nDevices	Number of monitors in the info and records arrays.
store_raw	Flag indicating whether raw data should be stored. (True: yes, False: no)
store_si	Flag indicating whether SI data should be stored. (True: yes, False: no)
compress	Flag indicating whether data should be compressed. This is almost always a good idea, but some old versions of Matlab (<2008b) have been found to have difficulty reading compressed data. (True: yes, -False: no)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_write_record_hdf2().

5.4.1.25 APDM_EXPORT int apdm_write_record_hdf2 (hid_t *file*, apdm_device_info_t * *info*, apdm_record_t * *records*, int *sampleNumber*, int *nDevices*, bool *store_raw*, bool *store_si*, bool *store_filtered*, bool *compress*)

Write an array of records to a HDF5 file (previously created with apdm_create_file_-hdf). The sample number must be tracked by the caller, and incremented for each new sample. In case of dropped data, it may be desired to increment the sampleNumber for each dropped sample.

Parameters

file	The HDF5 file handle returned from apdm_create_file_hdf
info	Array of apdm_device_info_t structs containing information about each monitor.
records	Array of apdm_record_t structs containing the data for one sample set of each monitor.
sample-Number	An index into the arrays stored in the HDF5 file. It is important that the first time this function is called sampleNumber is 0.
nDevices	Number of monitors in the info and records arrays.
store_raw	Flag indicating whether raw data should be stored. (True: yes, False: no)

store_si	Flag indicating whether SI data should be stored. (True: yes, False: no)
store_filtered	Flag indicating whether filtered data should be stored. (True: yes, False: no)
compress	Flag indicating whether data should be compressed. This is almost always a good idea, but some old versions of Matlab (<2008b) have been found to have difficulty reading compressed data. (True: yes, False: no)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_write_record_hdf().

5.5 Monitor

Functions

- APDM_EXPORT int apdm_sensor_close_and_free (apdm_device_handle_t device_handle)
- APDM_EXPORT apdm_device_handle_t apdm_sensor_allocate_and_open (const uint32_t sensor_index)
- APDM_EXPORT void apdm_sensor_free_handle (apdm_device_handle_t device_handle)
- APDM_EXPORT apdm_device_handle_t apdm_sensor_allocate_handle (void)
- APDM_EXPORT int apdm_sensor_open (apdm_device_handle_t device_handle, const uint32_t device_index)
- APDM_EXPORT int apdm_sensor_list_attached_sensors3 (uint32_t *serial_number_buffer, const uint32_t buffer_length, uint32_t *dest_count)
- APDM_EXPORT int apdm_sensor_close (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_verify_supported_calibration_version (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_verify_supported_version (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_override_minimum_supported_version (const char *new_version)
- APDM_EXPORT int apdm_sensor_comm_channel_verify_supported_version (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_initialize_device_info (apdm_device_info_t *device_info)
- APDM_EXPORT int apdm_sensor_apply_configuration (apdm_device_handle_t device_handle, apdm_device_info_t *device_info)
- APDM_EXPORT int apdm_device_extract_module_id_from_case_id_string (const char *case_id, uint32_t *dest_module_id)
- APDM_DEPRECATED APDM_EXPORT int apdm_sensor_get_device_id_list (uint32_t *serial_number_buffer, const uint32_t buffer_length)
- APDM_EXPORT int apdm_sensor_get_monitor_type (const char *case_id_string, apdm_monitor_type_t *dest)
- APDM_EXPORT int apdm_halt_all_attached_sensors (void)
- APDM_EXPORT int apdm_sensor_configure_wireless (apdm_device_handle_t device_handle, const enum APDMDeviceConfig wirelessConfigType, const uint32_t value)
- APDM_EXPORT int apdm_sensor_populate_device_info (apdm_device_handle_t device_handle, apdm_device_info_t *dest)
- APDM_DEPRECATED APDM_EXPORT int apdm_sensor_list_attached_sensors (uint32_t *serial_number_buffer, const uint32_t buffer_length)

5.5.1 Function Documentation

5.5.1.1 APDM_EXPORT int apdm_device_extract_module_id_from_case_id_string (const char * *case_id*, uint32_t * *dest_module_id*)

Extracts the module ID from a monitor case ID string, such as "SI-000025" will find 25.

Parameters

**case_id*	Case ID string from the motion monitor
**dest_module_id*	Destination into which to store the module ID

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.5.1.2 APDM_EXPORT int apdm_halt_all_attached_sensors (void)

Halts all sensors that are connected to the host. Note, make sure all device handles and contexts have been closed prior to calling this function.

Returns

Zero on success, non-zero error code if failure.

References apdm_log_debug(), apdm_log_error(), apdm_log_info(), apdm_sensor_allocate_handle(), apdm_sensor_close(), apdm_sensor_cmd_halt(), apdm_sensor_free_handle(), apdm_sensor_get_num_attached_dockingstations1(), apdm_sensor_open(), and apdm_strerror().

5.5.1.3 APDM_EXPORT int apdm_initialize_device_info (apdm_device_info_t * *device_info*)

Applys default configuration settings to a apdm_device_info_t structure

Parameters

**device_info*	Pointer to the structure to be initialized

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_monitor_decimation_rate_t_to_int(), apdm_monitor_output_select-_rate_t_to_int(), apdm_device_info_t::decimation_factor, apdm_device_info_t::erase_-sd_card_after_undocking, and apdm_device_info_t::orientation_model.

5.5.1.4 APDM_EXPORT apdm_device_handle_t apdm_sensor_allocate_and_open (const uint32_t *sensor_index*)

Allocates and opens/connects to a sensor on the system (monitor should be plugged into the docking station).

Parameters

sensor_-index	The index of the sensor attached to the host. This is the position of the dock in the chain, starting from zero.

Returns

Zero handle on error, non-zero handle on success.

References apdm_sensor_allocate_handle(), apdm_sensor_free_handle(), and apdm-_sensor_open().

5.5.1.5 APDM_EXPORT apdm_device_handle_t apdm_sensor_allocate_handle (void)

Allocates memory and returns a new sensor handle.

Returns

Zero on failure, non-zero handle on success.

References apdm_log_debug(), apdm_log_error(), and apdm_sensor_free_handle().

Referenced by apdm_halt_all_attached_sensors(), and apdm_sensor_allocate_and_-open().

5.5.1.6 APDM_EXPORT int apdm_sensor_apply_configuration (apdm_device_handle_t *device_handle,* apdm_device_info_t * *device_info*)

Parameters

device_-handle	The handle to the device to apply the configuration to.
device_info	The configuration to be applied to the device.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.5.1.7 APDM_EXPORT int apdm_sensor_close (apdm_device_handle_t *device_handle*)

Closes the handle.

Parameters

device_handle	The handle to be closed

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ctx_disconnect(), apdm_halt_all_attached_sensors(), apdm_sensor_close_and_free(), and apdm_sensor_get_device_id_list().

5.5.1.8 APDM_EXPORT int apdm_sensor_close_and_free (apdm_device_handle_t *device_handle*)

Closes and de-allocates a device handle.

Parameters

device_handle	The handle to be closed and deallocated

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.5.1.9 APDM_EXPORT int apdm_sensor_comm_channel_verify_supported_version (apdm_device_handle_t *device_handle*)

this function is used to verify that the given docking station handle has a version of firmware that is supported by the libraries.

Parameters

device_handle	The device handle

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.5.1.10 APDM_EXPORT int apdm_sensor_configure_wireless (apdm_ap_handle_t *handle,* const enum APDMDeviceConfig *wirelessConfigType,* const uint32_t *value*)

Used to set wireless config parameters on a device.

Parameters

handle	the device handle for which to configure.
wirelessConfigType	Which setting to be changed.
value	The value which to assign for that config parameter.

Returns

APDM_OK on success, error code otherwise.

References apdm_sensor_cmd_config_set().

5.5.1.11 APDM_EXPORT void apdm_sensor_free_handle (apdm_device_handle_t *device_handle*)

Frees memory associated with a device handle

Parameters

device_handle	The handle to be freed.

References apdm_log_debug().

Referenced by apdm_halt_all_attached_sensors(), apdm_sensor_allocate_and_open(), apdm_sensor_allocate_handle(), and apdm_sensor_close_and_free().

5.5.1.12 APDM_DEPRECATED APDM_EXPORT int apdm_sensor_get_device_id_list (uint32_t * *serial_number_buffer,* const uint32_t *buffer_length*)

This function will be removed after Jan 2014.

Retrieves a list of device ID's attached to the host.

Parameters

∗serial_number_buffer	Pointer to an array of uint32_t into which the serial numbers should be stored.
buffer_length	The number of elements in the destination array.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_info(), apdm_sensor_close(), apdm_sensor_cmd_device_id(), apdm_sensor_open(), and apdm_strerror().

5.5.1.13 APDM_EXPORT int apdm_sensor_get_monitor_type (const char ∗ *case_id_string,* apdm_monitor_type_t ∗ *dest*)

Parses a case ID string from a motion monitor and identifies which type of monitor it is (opal, emerald, saphire)

Parameters

case_id_string	Case ID from the monitor
∗dest	Destination into which the monitor type will be stored

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.5.1.14 APDM_DEPRECATED APDM_EXPORT int apdm_sensor_list_attached_sensors (uint32_t ∗ *serial_number_buffer,* const uint32_t *buffer_length*)

Fills the buffer pointed to by serial_number_buffer with a list of Motion Monitor ID numbers.

Deprecated non-standard function semantics, see apdm_sensor_list_attached_sensors3(). Will be removed after March 2011.

Parameters

∗serial_number_buffer	Destination array into which device IDs are to be populated
buffer_length	The maximum number of entries in the serial_number_buffer buffer.

Returns

APDM_OK on success.

References apdm_sensor_list_attached_sensors3().

5.5.1.15 APDM_EXPORT int apdm_sensor_list_attached_sensors3 (uint32_t ∗ *serial_number_buffer,* const uint32_t *buffer_length,* uint32_t ∗ *dest_count*)

Fills the buffer pointed to by serial_number_buffer with a list of Motion Monitor ID numbers.

Parameters

∗serial_number_buffer	Destination array into which device IDs are to be populated
buffer_length	The maximum number of entries in the serial_number_buffer buffer.
∗dest_count	The number of devices added to the buffer list

Returns

APDM_OK on success, MONITOR_READ_TIMEOUT_ERROR if one (or more) of the docks does not have a monitor plugged in, otherwise error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_list_attached_sensors().

5.5.1.16 APDM_EXPORT int apdm_sensor_open (apdm_device_handle_t *device_handle,* const uint32_t *device_index*)

Opens a sensor by it's corresponding index number on the host computer.

Parameters

device_handle	The handle to which the corresponding sensor is to be associated with
device_index	The index of the device which is to be opened

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_halt_all_attached_sensors(), apdm_sensor_allocate_and_open(), and apdm_sensor_get_device_id_list().

5.5.1.17 APDM_EXPORT int apdm_sensor_override_minimum_supported_version (const char * *new_version*)

Allows you to override the minimum motion sensor version number used to validate motion sensor versions.

Parameters

new_version	Version number, e.g. "2010-09-02" Set this to NULL to use library default version number.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.5.1.18 APDM_EXPORT int apdm_sensor_populate_device_info (apdm_device_handle_t *device_handle,* apdm_device_info_t * *dest*)

Parameters

device_handle	The device handle.
**dest*	Destination into which to put the device info associated with the device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_get_time_ms_64(), apdm_log_debug(), apdm_log_error(), apdm_sensor_cmd_calibration_data_blob(), apdm_sensor_cmd_calibration_version(), apdm-

_sensor_cmd_case_id(), apdm_sensor_cmd_config_get(), apdm_sensor_cmd_device_id(), apdm_sensor_cmd_hw_id(), apdm_sensor_cmd_protocol_version(), apdm_sensor_cmd_user_calibration_data_blob(), apdm_sensor_cmd_version_string_1(), apdm_sensor_cmd_version_string_2(), apdm_sensor_cmd_version_string_3(), apdm_sensor_config_get_label(), apdm_strerror(), apdm_device_info_t::protocol_version, apdm_device_info_t::selected_temperature_sensor, apdm_device_info_t::timezone, apdm_device_info_t::wireless_addr_id, apdm_device_info_t::wireless_block0, apdm_device_info_t::wireless_block1, apdm_device_info_t::wireless_block2, apdm_device_info_t::wireless_block3, apdm_device_info_t::wireless_channel1, apdm_device_info_t::wireless_channel2, apdm_device_info_t::wireless_channel3, and apdm_device_info_t::wireless_timeslice.

5.5.1.19 APDM_EXPORT int apdm_sensor_verify_supported_calibration_version (apdm_device_handle_t *device_handle*)

this function is used to verify that the given device handle has a version of calibration data that is supported by the libraries.

Parameters

device_handle	The device handle

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_sensor_cmd_calibration_version().

5.5.1.20 APDM_EXPORT int apdm_sensor_verify_supported_version (apdm_device_handle_t *device_handle*)

this function is used to verify that the given device handle has a version of firmware that is supported by the libraries.

Parameters

device_handle	The device handle

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error(), and apdm_sensor_cmd_version_string_2().

5.6 MonitorCommands

Functions

- APDM_EXPORT int apdm_sensor_cmd_halt (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_resume (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_reset (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_run (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_memory_crc16 (apdm_device_handle_t device_handle, const uint32_t address, const uint16_t length, uint16_t *current_value)
- APDM_EXPORT int apdm_sensor_cmd_write_flash_block (apdm_device_handle_t device_handle, const uint32_t address, uint8_t *data, const uint32_t length)
- APDM_EXPORT int apdm_sensor_cmd_time_set (apdm_device_handle_t device_handle, uint32_t year, uint32_t month, uint32_t day, uint32_t hour, uint32_t minute, uint32_t second)
- APDM_EXPORT int apdm_sensor_cmd_time_set2 (apdm_device_handle_t device_handle, const time_t epoch_time)
- APDM_EXPORT int apdm_sensor_cmd_time_get (apdm_device_handle_t device_handle, uint32_t *year, uint32_t *month, uint32_t *day, uint32_t *hour, uint32_t *minute, uint32_t *second)
- APDM_EXPORT int apdm_sensor_cmd_flash_block_set (apdm_device_handle_t device_handle, uint32_t block)
- APDM_EXPORT int apdm_sensor_cmd_flash_block_get (apdm_device_handle_t device_handle, uint32_t *block)
- APDM_EXPORT int apdm_sensor_cmd_battery_voltage (apdm_device_handle_t device_handle, uint16_t *voltage)
- APDM_EXPORT int apdm_sensor_cmd_battery_charge_rate (apdm_device_handle_t device_handle, uint16_t rate)
- APDM_EXPORT int apdm_sensor_cmd_calibration_version (apdm_device_handle_t device_handle, uint32_t *calibration_version)
- APDM_EXPORT int apdm_sensor_cmd_memory_dump (apdm_device_handle_t device_handle, const uint32_t monitor_memory_address, const int num_bytes_to_read, char *BYTE_ARRAY, const int dest_buffer_length)
- APDM_EXPORT int apdm_sensor_cmd_peek (apdm_device_handle_t device_handle, const uint32_t address, uint8_t *current_value)
- APDM_EXPORT int apdm_sensor_cmd_peek2 (apdm_device_handle_t device_handle, const uint32_t address, uint16_t *current_value)

- APDM_EXPORT int apdm_sensor_cmd_poke (apdm_device_handle_t device_handle, const uint32_t address, uint8_t new_value)
- APDM_EXPORT int apdm_sensor_cmd_poke2 (apdm_device_handle_t device_handle, const uint32_t address, uint16_t new_value)
- APDM_EXPORT int apdm_sensor_cmd_sync_get (apdm_device_handle_t device_handle, uint64_t *current_value)
- APDM_EXPORT int apdm_sensor_cmd_sync_dock_wait (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_led_pattern (apdm_device_handle_t device_handle, uint8_t interval, uint8_t *pattern, uint8_t length)
- APDM_EXPORT int apdm_sensor_cmd_led_reset (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_off_reason (apdm_device_handle_t device_handle, uint8_t *reason)
- APDM_EXPORT int apdm_sensor_cmd_uptime_get (apdm_device_handle_t device_handle, uint32_t *uptime)
- APDM_EXPORT int apdm_sensor_cmd_uptime_reset (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_last_uptime (apdm_device_handle_t device_handle, uint32_t *uptime)
- APDM_EXPORT int apdm_sensor_cmd_last_standby_uptime (apdm_device_handle_t device_handle, uint32_t *uptime)
- APDM_EXPORT int apdm_sensor_cmd_unlock_bootloader_flash (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_enter_bootloader (apdm_device_handle_t device_handle, const char *password, const int password_length)
- APDM_EXPORT int apdm_sensor_cmd_bootloader_version (apdm_device_handle_t device_handle, uint32_t *dest_version)
- APDM_EXPORT int apdm_sensor_cmd_sample_start (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_sample_get (apdm_device_handle_t device_handle, uint8_t *dest_buffer, const int buff_length)
- APDM_EXPORT int apdm_sensor_cmd_sync_set (apdm_device_handle_t device_handle, const uint64_t new_value)
- APDM_EXPORT int apdm_sensor_cmd_sync_commit (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_config_commit (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_ping (apdm_device_handle_t device_handle, uint8_t *mode)
- APDM_EXPORT int apdm_sensor_cmd_device_id (apdm_device_handle_t device_handle, uint32_t *current_value)
- APDM_EXPORT int apdm_sensor_cmd_error_count (apdm_device_handle_t device_handle, uint32_t *error_count)

- APDM_EXPORT int apdm_sensor_cmd_error_name (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int length, uint16_t error_id)
- APDM_EXPORT int apdm_sensor_cmd_error_log_size (apdm_device_handle_t device_handle, uint16_t *error_log_size)
- APDM_EXPORT int apdm_sensor_cmd_error_log_get (apdm_device_handle_t device_handle, const uint16_t offset, uint16_t *error_id)
- APDM_EXPORT int apdm_sensor_cmd_error_stats_size (apdm_device_handle_t device_handle, uint16_t *stats_size)
- APDM_EXPORT int apdm_sensor_cmd_error_stats_get (apdm_device_handle_t device_handle, const uint16_t id, uint16_t *count)
- APDM_EXPORT int apdm_sensor_cmd_stats_size (apdm_device_handle_t device_handle, uint16_t *value)
- APDM_EXPORT int apdm_sensor_cmd_stats_max_get (apdm_device_handle_t device_handle, const uint16_t id, uint16_t *max_val)
- APDM_EXPORT int apdm_sensor_cmd_stats_min_get (apdm_device_handle_t device_handle, const uint16_t id, uint16_t *min_val)
- APDM_EXPORT int apdm_sensor_cmd_stats_count_get (apdm_device_handle_t device_handle, const uint16_t id, uint16_t *count_val)
- APDM_EXPORT int apdm_sensor_cmd_stats_sum_get (apdm_device_handle_t device_handle, const uint16_t id, uint32_t *sum_val)
- APDM_EXPORT int apdm_sensor_cmd_stats_clear (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_error_clear (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_battery_charge_status (apdm_device_handle_t device_handle, uint8_t *current_status)
- APDM_EXPORT int apdm_sensor_cmd_calibration_data_blob (apdm_device_handle_t device_handle, uint8_t *dest, const int dest_length)
- APDM_EXPORT int apdm_sensor_cmd_user_calibration_data_blob (apdm_device_handle_t dev_handle, uint8_t *dest, const int dest_length)
- APDM_EXPORT int apdm_sensor_cmd_calibration_data (apdm_device_handle_t device_handle, apdm_sensor_compensation_t *sensor_comp)
- APDM_EXPORT int apdm_sensor_cmd_user_calibration_data (apdm_device_handle_t dev_handle, apdm_sensor_compensation_t *sensor_comp)
- APDM_EXPORT int apdm_sensor_cmd_version_string_1 (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int dest_buff_length)
- APDM_EXPORT int apdm_sensor_cmd_version_string_2 (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int dest_buff_length)
- APDM_EXPORT int apdm_sensor_cmd_version_string_3 (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int dest_buff_length)
- APDM_EXPORT int apdm_sensor_cmd_dock_status (apdm_device_handle_t device_handle, uint8_t *status)
- APDM_EXPORT int apdm_sensor_cmd_config_get (apdm_device_handle_t device_handle, const enum APDMDeviceConfig config_type, uint32_t *value)

- APDM_EXPORT int apdm_sensor_cmd_config_set (apdm_device_handle_t device_handle, const enum APDMDeviceConfig config_type, const uint32_t value)
- APDM_EXPORT int apdm_sensor_config_set_label (apdm_device_handle_t device_handle, const char label_str[16], const int str_length)
- APDM_EXPORT int apdm_sensor_config_get_label (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int buff_size)
- APDM_EXPORT int apdm_sensor_cmd_config_status (apdm_device_handle_t device_handle, uint8_t *status)
- APDM_EXPORT int apdm_sensor_cmd_timer_adjust_get (apdm_device_handle_t device_handle, uint16_t *value)
- APDM_EXPORT int apdm_sensor_cmd_debug_set (apdm_device_handle_t device_handle, uint8_t id, uint32_t data)
- APDM_EXPORT int apdm_sensor_cmd_debug_get (apdm_device_handle_t device_handle, uint8_t id, uint32_t *data)
- APDM_EXPORT int apdm_sensor_cmd_dock (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_undock (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_config_check (apdm_device_handle_t device_handle, uint8_t *is_valid)
- APDM_EXPORT int apdm_sensor_cmd_flash_format (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_standby (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_cmd_case_id (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int dest_buff_length)
- APDM_EXPORT int apdm_sensor_cmd_hw_id (apdm_device_handle_t device_handle, uint32_t *current_value)
- APDM_EXPORT int apdm_sensor_cmd_protocol_version (apdm_device_handle_t h, uint32_t *protocol_version)

5.6.1 Function Documentation

5.6.1.1 APDM_EXPORT int apdm_sensor_cmd_battery_charge_rate (apdm_device_handle_t *device_handle,* uint16_t *rate*)

Sets the battery charge rate

Parameters

device_handle	The device handle.
rate	Units of milliamps, minimum = 100mA, max = 450mA;

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.2 APDM_EXPORT int apdm_sensor_cmd_battery_charge_status (apdm_device_handle_t *device_handle,* uint8_t * *current_status*)

Retrieves the battery charge status

Parameters

device_handle	The device handle.
*current_status	Destination into which to store the battery charge status, values are defined in "enum APDM_Battery_Charge_Status" in apdm_types.h.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.3 APDM_EXPORT int apdm_sensor_cmd_battery_voltage (apdm_device_handle_t *device_handle,* uint16_t * *voltage*)

Parameters

device_handle	The device handle.
*voltage	Destination into which to store the battery voltage, this is in units of the raw ADC value off the MCU.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.4 APDM_EXPORT int apdm_sensor_cmd_bootloader_version (apdm_device_handle_t *device_handle,* uint32_t * *dest_version*)

This command is only supported on v1.1 or later monitors. Previous monitor version will result in a return code of APDM_DEVICE_RESPONSE_ERROR_INVALID_COMMAND.

Parameters

device_handle	The device handle.
∗dest_version	The destination into which to store the bootloader version number

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.5 APDM_EXPORT int apdm_sensor_cmd_calibration_data (apdm_device_handle_t *device_handle,* apdm_sensor_compensation_t ∗ *sensor_comp*)

This function will retrieve the sensor calibration data from the given devices via dev_handle

Parameters

device_handle	The device handle.
∗sensor_comp	Destination into which to store sensor compensation data

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.6 APDM_EXPORT int apdm_sensor_cmd_calibration_data_blob (apdm_device_handle_t *device_handle,* uint8_t ∗ *dest,* const int *dest_length*)

Returns the packed binary representation of the motion monitor calibration data

Parameters

device_handle	The device handle.
∗dest	Destination buffer into which to store the packed cal data.
dest_length	The size of the destination buffer.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info().

5.6.1.7 APDM_EXPORT int apdm_sensor_cmd_calibration_version (apdm_device_handle_t *device_handle,* uint32_t * *calibration_version*)

Gets the version of calibration date currently on the motion monitor

Parameters

device_handle	The device handle.
calibration_version	Destination into which to store the calibration varsion number

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info(), and apdm_sensor_verify_supported_calibration_version().

5.6.1.8 APDM_EXPORT int apdm_sensor_cmd_case_id (apdm_device_handle_t *device_handle,* char * *BYTE_ARRAY,* const int *dest_buff_length*)

Retrieves the case ID from the motion monitor

Parameters

device_handle	The device handle.
BYTE_ARRAY	Destination buffer into which to store the case ID
dest_buff_length	size of *BYTE_ARRAY

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

Referenced by apdm_sensor_populate_device_info().

5.6.1.9 APDM_EXPORT int apdm_sensor_cmd_config_check (apdm_device_handle_t *device_handle,* uint8_t * *is_valid*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h - Sets what is pointed to by is_valid to 1 if the configuration is valid, sets it to 0 otherwise.

5.6.1.10 APDM_EXPORT int apdm_sensor_cmd_config_commit (apdm_device_handle_t *device_handle*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.11 APDM_EXPORT int apdm_sensor_cmd_config_get (apdm_device_handle_t *device_handle,* const enum APDMDeviceConfig *config_type,* uint32_t * *value*)

Retrieves a specified configuration parameter type.

Parameters

device_handle	The device handle.
config_type	The configuration parameter type.
*value	The destination into which to store the parameter value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_config_get_label(), and apdm_sensor_populate_device_info().

5.6.1.12 APDM_EXPORT int apdm_sensor_cmd_config_set (apdm_device_handle_t *device_handle,* const enum APDMDeviceConfig *config_type,* const uint32_t *value*)

Sets a specified configuration parameter.

Parameters

device_handle	The device handle.
config_type	The parameter which is to be set, from enum APDMDeviceConfig in apdm_types.h.
value	The value to which it is to be set. There are enums in apdm_types.h to specify valid values for various parameter types.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_config_set_label(), and apdm_sensor_configure_wireless().

5.6.1.13 APDM_EXPORT int apdm_sensor_cmd_config_status (apdm_device_handle_t *device_handle,* uint8_t * *status*)

This command returns the current status of if the configuration has been committed or not.

Parameters

device_handle	The device handle.
**status*	Destination into which to put the configuration status, 0 indicates the config has not been committed, 1 indicates that it has been committed.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.14 APDM_EXPORT int apdm_sensor_cmd_debug_get (apdm_device_handle_t *device_handle,* uint8_t *id,* uint32_t * *data*)

This command gets the debug value identified by the id parameter.

Parameters

device_handle	The device handle.
id	The debug variable ID which is to be retrieved
data	The destination into which to store the debug value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.15 APDM_EXPORT int apdm_sensor_cmd_debug_set (apdm_device_handle_t *device_handle*, uint8_t *id*, uint32_t *data*)

This command sets the debug value identified by the id parameter.

Parameters

device_handle	The device handle.
id	The debug variable ID which is to be set
data	The value to which it is to be set.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.16 APDM_EXPORT int apdm_sensor_cmd_device_id (apdm_device_handle_t *device_handle*, uint32_t * *current_value*)

Retrieves the device ID off the motion monitor

Parameters

device_handle	The device handle.
*current_value	Destination into which to store the device id.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_get_device_id_list(), and apdm_sensor_populate_device_info().

5.6.1.17 APDM_EXPORT int apdm_sensor_cmd_dock (apdm_device_handle_t *device_handle*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.18 APDM_EXPORT int apdm_sensor_cmd_dock_status (apdm_device_handle_t *device_handle,* uint8_t * *status*)

Parameters

device_handle	The device handle.
status	Values defined in enum apdm_monitor_dock_status_t

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.19 APDM_EXPORT int apdm_sensor_cmd_enter_bootloader (apdm_device_handle_t *device_handle,* const char * *password,* const int *password_length*)

Parameters

device_handle	The device handle.
password	The password, of length 8, to enter the bootloader (password differs based on monitor version)
password_length	The length of the password, must be 8 bytes long.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.6.1.20 APDM_EXPORT int apdm_sensor_cmd_error_clear (apdm_device_handle_t *device_handle*)

Clears all errors and error stats on the motion monitor.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.21 APDM_EXPORT int apdm_sensor_cmd_error_count (apdm_device_handle_t *device_handle,* uint32_t * *error_count*)

Gets the number of errors on the motion monitor.

Parameters

device_handle	The device handle.
**error_count*	Destination into which to store the error count.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.22 APDM_EXPORT int apdm_sensor_cmd_error_log_get (apdm_device_handle_t *device_handle,* const uint16_t *offset,* uint16_t * *error_id*)

Retrieves the number of times the error at offset has occurred.

Parameters

device_handle	The device handle.
offset	The error number offset to retrieve
∗*error_id*	Destination into which to store the error count for the specified offset.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.23 APDM_EXPORT int apdm_sensor_cmd_error_log_size (apdm_device_handle_t *device_handle,* uint16_t ∗ *error_log_size*)

Retrieves the size of the error log on the motion monitor

Parameters

device_handle	The device handle.
∗*error_log_size*	Destination into which to store the error log size.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.24 APDM_EXPORT int apdm_sensor_cmd_error_name (apdm_device_handle_t *device_handle,* char ∗ *BYTE_ARRAY,* const int *length,* uint16_t *error_id*)

Retrieves the name of the specified error ID.

Parameters

device_handle	The device handle.
∗*BYTE_ARRAY*	Destination string into which to store the name of the error
length	The size of the ∗BYTE_ARRAY string buffer
error_id	The ID for which you want to retrieve the error name

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.25 APDM_EXPORT int apdm_sensor_cmd_error_stats_get (apdm_device_handle_t *device_handle,* const uint16_t *id,* uint16_t * *count*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.26 APDM_EXPORT int apdm_sensor_cmd_error_stats_size (apdm_device_handle_t *device_handle,* uint16_t * *stats_size*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.27 APDM_EXPORT int apdm_sensor_cmd_flash_block_get (apdm_device_handle_t *device_handle,* uint32_t * *block*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.28 APDM_EXPORT int apdm_sensor_cmd_flash_block_set (apdm_device_handle_t *device_handle,* uint32_t *block*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.29 APDM_EXPORT int apdm_sensor_cmd_flash_format (apdm_device_handle_t *device_handle*)

Causes the motion monitor to re-format it's SD card when it is removed from the docking station or device cable.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.30 APDM_EXPORT int apdm_sensor_cmd_halt (apdm_device_handle_t *device_handle*)

When the motion monitor is removed from the dock, or disconnected from the cable, the motion monitor will halt.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_halt_all_attached_sensors().

5.6.1.31 APDM_EXPORT int apdm_sensor_cmd_hw_id (apdm_device_handle_t *device_handle*, uint32_t * *current_value*)

Retrieves the hardware ID of the motion monitor.

Parameters

device_handle	The device handle.
*current_value	The destination into which to store the hardware ID.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info().

5.6.1.32 APDM_EXPORT int apdm_sensor_cmd_last_standby_uptime (apdm_device_handle_t *device_handle*, uint32_t * *uptime*)

This command returns the last max uptime the device achieved while in standby mode. Mainly useful as a debugging command.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.33 APDM_EXPORT int apdm_sensor_cmd_last_uptime (apdm_device_handle_t *device_handle*, uint32_t * *uptime*)

This command returns the last max uptime the device achieved while running before powering off or going into standby mode. Mainly useful as a debugging command.

Parameters

device_handle	The device handle.
*uptime	Destination into which to store the last uptime.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.34 APDM_EXPORT int apdm_sensor_cmd_led_pattern (apdm_device_handle_t *device_handle,* uint8_t *interval,* uint8_t * *pattern,* uint8_t *length*)

Led pattern is sent to the device as a character string which represents the led color pattern to display.

Parameters

device_handle	The device handle.
*pattern	

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.35 APDM_EXPORT int apdm_sensor_cmd_led_reset (apdm_device_handle_t *device_handle*)

Tells the device to go back to its normal led sequence (after having been overriden by apdm_sensor_cmd_led_pattern())

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.36 APDM_EXPORT int apdm_sensor_cmd_memory_crc16 (apdm_device_handle_t *device_handle,* const uint32_t *address,* const uint16_t *length,* uint16_t * *current_value*)

Does a CRC check on the given address and length of flash in the motion monitor

Parameters

device_handle	The device handle.
address	The start address of the CRC check.
length	The number of bytes to CRC
∗current_value	Destination into which to store the CRC value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.37 APDM_EXPORT int apdm_sensor_cmd_memory_dump (apdm_device_handle_t *device_handle,* const uint32_t *monitor_memory_address,* const int *num_bytes_to_read,* char ∗ *BYTE_ARRAY,* const int *dest_buffer_length*)

Parameters

device_handle	The device handle.
monitor_memory_address	The start address in the monitors address space to start reading from
num_bytes_to_read	The number of bytes of memory to read from the device, must be $>$ 0 and less then 32768
∗BYTE_ARRAY	The destination into which the memory dump is to be stored, must be non-null
dest_buffer_length	The length of the destination buffer pointed to by ∗BYTE_ARRAY, must be $>$ 0 and $>=$ num_bytes_to_read

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.6.1.38 APDM_EXPORT int apdm_sensor_cmd_off_reason (apdm_device_handle_t *device_handle,* uint8_t ∗ *reason*)

Parameters

device_handle	The device handle.
∗reason	Destination into which to store the reason code, reason is defined in 'enum apdm_monitor_off_reason_t' in apdm_types.h

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.39 APDM_EXPORT int apdm_sensor_cmd_peek (apdm_device_handle_t *device_handle,* const uint32_t *address,* uint8_t * *current_value*)

Peeks an 8-bit value in the motion monitor address space.

Parameters

device_handle	The device handle.
address	The address to be peeked
*current_value	The destination pointer into which to store the value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.40 APDM_EXPORT int apdm_sensor_cmd_peek2 (apdm_device_handle_t *device_handle,* const uint32_t *address,* uint16_t * *current_value*)

Peeks an 16-bit value in the motion monitor address space.

Parameters

device_handle	The device handle.
address	The address to be peeked
*current_value	The destination pointer into which to store the value.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.41 APDM_EXPORT int apdm_sensor_cmd_ping (apdm_device_handle_t *device_handle,* uint8_t * *mode*)

This command queries if the device is present and what its state is in regards to the bootloader (pre-bootloader/bootloader/post-bootloader).

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.42 APDM_EXPORT int apdm_sensor_cmd_poke (apdm_device_handle_t *device_handle*, const uint32_t *address*, uint8_t *new_value*)

Writes an 8-bit value into the address space of the motion monitor, note, writing to flash address space won't work.

Parameters

device_handle	The device handle.
address	The address at which to write to
new_value	The value to be written

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.43 APDM_EXPORT int apdm_sensor_cmd_poke2 (apdm_device_handle_t *device_handle*, const uint32_t *address*, uint16_t *new_value*)

Writes an 16-bit value into the address space of the motion monitor, note, writing to flash address space won't work.

Parameters

device_handle	The device handle.
address	The address at which to write to
new_value	The value to be written

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.44 APDM_EXPORT int apdm_sensor_cmd_protocol_version (apdm_device_handle_t *h,* uint32_t * *protocol_version*)

Retrieves the protocol version number from the monitor. This represents the binary packing and semantics used during wireless transmission.

Parameters

device_handle	The device handle.
**protocol_version*	Pointer to where the protocol version is to be stored

Returns

APDM_OK on success, error code otherwise.

Referenced by apdm_sensor_populate_device_info().

5.6.1.45 APDM_EXPORT int apdm_sensor_cmd_reset (apdm_device_handle_t *device_handle*)

Causes the monitor to reset.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.46 APDM_EXPORT int apdm_sensor_cmd_resume (apdm_device_handle_t *device_handle*)

When the motion monitor has been commanded to halt, you can un-set the halt flag on the monitor.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.47 APDM_EXPORT int apdm_sensor_cmd_run (apdm_device_handle_t *device_handle*)

Commands the device to enter run mode.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

This command is used to instruct the device to into run mode.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.48 APDM_EXPORT int apdm_sensor_cmd_sample_get (apdm_device_handle_t *device_handle,* uint8_t * *dest_buffer,* const int *buff_length*)

Gets the 1 second of data that was initiated by the sample_start() command.

The format of the data returned is a set of 4x 512byte blocks with 25 sample sets each. There will be 12bytes of padding at the end of each 512byte block. This provides the host with 100 total samples. Each sample is a 16bit value with a sample set packed in the following order AX,AY,AZ,GX,GY,GZ,MX,MY,MZ,T. (T=temperature)

Parameters

device_handle	The device handle.
*dest_buffer	Destination buffer into which to store samples taken.
buff_length	The length of the buffer pointed to by *dest_buffer, must be >= 2048 bytes

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.6.1.49 APDM_EXPORT int apdm_sensor_cmd_sample_start (apdm_device_handle_t *device_handle*)

Starts a 1-second cycle of the device sampling data on its internal sensors. Using the following settings:

- output rate 128
- decimation factor 5x2
- no mag set/reset
- all sensors enabled
- temperature from gyro

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.50 APDM_EXPORT int apdm_sensor_cmd_standby (apdm_device_handle_t *device_handle*)

Standby mode on the opal allows the device to retain the correct time while not recording. Issuing the standby command to the opal will instruct it to transition to this mode the next time it is undocked. The device will appear to power off but will instead be in a low power state updating the clock once a second. The duration that the device can stay in this mode before needing to fully power off will be dependent on how much battery charge is available. This allows users to ship or otherwise store the device for between a day to a week while keeping the time correct on the device.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.51 APDM_EXPORT int apdm_sensor_cmd_stats_clear (apdm_device_handle_t *device_handle*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.52 APDM_EXPORT int apdm_sensor_cmd_stats_count_get (apdm_device_handle_t *device_handle,* const uint16_t *id,* uint16_t * *count_val*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.53 APDM_EXPORT int apdm_sensor_cmd_stats_max_get (apdm_device_handle_t *device_handle,* const uint16_t *id,* uint16_t * *max_val*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.54 APDM_EXPORT int apdm_sensor_cmd_stats_min_get (apdm_device_handle_t *device_handle,* const uint16_t *id,* uint16_t * *min_val*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.55 APDM_EXPORT int apdm_sensor_cmd_stats_size (apdm_device_handle_t *device_handle*, uint16_t * *value*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.56 APDM_EXPORT int apdm_sensor_cmd_stats_sum_get (apdm_device_handle_t *device_handle*, const uint16_t *id*, uint32_t * *sum_val*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.57 APDM_EXPORT int apdm_sensor_cmd_sync_commit (apdm_device_handle_t *device_handle*)

Commits the sync value previously set by cmd_sync_set() thus causing the change to take effect.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.58 APDM_EXPORT int apdm_sensor_cmd_sync_dock_wait (apdm_device_handle_t *device_handle*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.59 APDM_EXPORT int apdm_sensor_cmd_sync_get (apdm_device_handle_t *device_handle,* uint64_t * *current_value*)

Retrieves the sync value currently on the motion monitor.

Parameters

device_handle	The device handle.
*current_value	The destination into which to store the current sync value on the motion monitor

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.60 APDM_EXPORT int apdm_sensor_cmd_sync_set (apdm_device_handle_t *device_handle,* const uint64_t *new_value*)

Sets the sync value on the motion monitor, should call cmd_sync_commit() sometime after the sync value is set.

Parameters

device_handle	The device handle.
new_value	The new sync value to be set.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

5.6.1.61 APDM_EXPORT int apdm_sensor_cmd_time_get (apdm_device_handle_t *device_handle,* uint32_t * *year,* uint32_t * *month,* uint32_t * *day,* uint32_t * *hour,* uint32_t * *minute,* uint32_t * *second*)

Retrieves the time from the motion monitor.

Parameters

device_handle	The device handle.
*year	
*month	
*day	
*hour	
*minute	
*second	

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

5.6.1.62 APDM_EXPORT int apdm_sensor_cmd_time_set (apdm_device_handle_t *device_handle,* uint32_t *year,* uint32_t *month,* uint32_t *day,* uint32_t *hour,* uint32_t *minute,* uint32_t *second*)

Sets the time on the motion monitor

Parameters

device_handle	The device handle.
year	0-9999
month	1-12
day	1-31
hour	0-23
minute	0-59
second	0-59

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug().

Referenced by apdm_sensor_cmd_time_set2().

5.6.1.63 APDM_EXPORT int apdm_sensor_cmd_time_set2 (apdm_device_handle_t *device_handle,* const time_t *epoch_time*)

Sets the time on the Motion Monitor in terms of the epoch time (number of seconds since 1970)

Parameters

device_handle	The device handle.
epoch_time	Number of seconds since 1970.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_sensor_cmd_time_set().

5.6.1.64 APDM_EXPORT int apdm_sensor_cmd_timer_adjust_get (apdm_device_handle_t *device_handle,* uint16_t * *value*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.65 APDM_EXPORT int apdm_sensor_cmd_undock (apdm_device_handle_t *device_handle*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.66 APDM_EXPORT int apdm_sensor_cmd_unlock_bootloader_flash (apdm_device_handle_t *device_handle*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.67 APDM_EXPORT int apdm_sensor_cmd_uptime_get (apdm_device_handle_t *device_handle,* uint32_t * *uptime*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.68 APDM_EXPORT int apdm_sensor_cmd_uptime_reset (apdm_device_handle_t *device_handle*)

This command resets the uptime counter on the device. Mainly useful as a debugging command.

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.69 APDM_EXPORT int apdm_sensor_cmd_user_calibration_data (apdm_device_handle_t *dev_handle,* apdm_sensor_compensation_t * *sensor_comp*)

This function will retrieve the user-overridden sensor calibration data from the given devices via dev_handle. Requires monitor firmware versions newer then March 2011.

Parameters

device_handle	The device handle.
**sensor_comp*	Destination into which to store sensor compensation data

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.70 APDM_EXPORT int apdm_sensor_cmd_user_calibration_data_blob (apdm_device_handle_t *dev_handle,* uint8_t * *dest,* const int *dest_length*)

Retrieves the user calibration data from the sensor (from re-calibration in the field)

Parameters

dev_handle	The device handle of which you want user calibration from.
**dest*	The destination buffer into which binary data is to be stored.
dest_length	The length of the destination buffer, which is not to be overrun

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info().

5.6.1.71 APDM_EXPORT int apdm_sensor_cmd_version_string_1 (apdm_device_handle_t *device_handle,* char * *BYTE_ARRAY,* const int *dest_buff_length*)

Parameters

device_handle	The device handle.
**BYTE_ARRAY*	Destination into which to store the version string.

dest_buff_length	length of the *BYTE_ARRAY array.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info().

5.6.1.72 APDM_EXPORT int apdm_sensor_cmd_version_string_2 (apdm_device_handle_t *device_handle*, char * *BYTE_ARRAY*, const int *dest_buff_length*)

Parameters

device_handle	The device handle.
**BYTE_ARRAY*	Destination into which to store the version string.
dest_buff_length	length of the *BYTE_ARRAY array.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info(), and apdm_sensor_verify_supported_version().

5.6.1.73 APDM_EXPORT int apdm_sensor_cmd_version_string_3 (apdm_device_handle_t *device_handle*, char * *BYTE_ARRAY*, const int *dest_buff_length*)

Parameters

device_handle	The device handle.
**BYTE_ARRAY*	Destination into which to store the version string.
dest_buff_length	length of the *BYTE_ARRAY array.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_sensor_populate_device_info().

5.6.1.74 APDM_EXPORT int apdm_sensor_cmd_write_flash_block (apdm_device_handle_t *device_handle,* const uint32_t *address,* uint8_t * *data,* const uint32_t *length*)

Parameters

device_handle	The device handle.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.6.1.75 APDM_EXPORT int apdm_sensor_config_get_label (apdm_device_handle_t *device_handle,* char * *BYTE_ARRAY,* const int *buff_size*)

Wrapper function for getting the label config parameter

Parameters

device_handle	The device handle.
BYTE_ARRAY	16 character array destination to store the label.
buff_size	The size of the buffer into which to store the label, must be at least 16

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug(), apdm_log_error(), and apdm_sensor_cmd_config_get().

Referenced by apdm_sensor_populate_device_info().

5.6.1.76 APDM_EXPORT int apdm_sensor_config_set_label (apdm_device_handle_t *device_handle,* const char *label_str[16],* const int *str_length*)

Wrapper function for setting the label config parameter

Parameters

device_handle	The device handle.
label_str	16 character array source for label.
str_length	The length of the buffer pointed to by label_str

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug(), apdm_log_error(), and apdm_sensor_cmd_config_set().

5.7 DockingStation

Functions

- APDM_EXPORT int apdm_sensor_get_num_attached_dockingstations1 (uint32_t *dest_num_docks)
- APDM_EXPORT int apdm_ds_override_minimum_supported_version (const uint64_t new_version)
- APDM_EXPORT int apdm_ds_get_docked_module_id (apdm_device_handle_t device_handle, uint32_t *dest)
- APDM_EXPORT int apdm_ds_get_protocol_subversion (apdm_device_handle_t device_handle, int64_t *dest_protocol_subversion)
- APDM_EXPORT int apdm_ds_get_hardware_version (apdm_device_handle_t device_handle, uint32_t *dest)
- APDM_EXPORT int apdm_ds_get_firmware_version (apdm_device_handle_t device_handle, uint64_t *dest)
- APDM_EXPORT int apdm_ds_get_case_id (apdm_device_handle_t device_handle, char *BYTE_ARRAY, const int dest_buffer_length)
- APDM_EXPORT int apdm_ds_get_serial_number_by_index (const int docking_station_index, uint32_t *serial_number)
- APDM_EXPORT int apdm_ds_is_monitor_present (apdm_device_handle_t device_handle, uint32_t *output_flag)
- APDM_EXPORT int apdm_ds_is_monitor_data_forwarding_enabled (apdm_device_handle_t device_handle, uint32_t *output_flag)
- APDM_EXPORT int apdm_ds_get_serial (apdm_device_handle_t device_handle, uint32_t *serial_number)
- APDM_EXPORT int apdm_ds_get_index_by_serial_number (const uint32_t serial_number, uint32_t *docking_station_index)
- APDM_EXPORT int apdm_ds_set_monitor_baud_rate (apdm_device_handle_t ds_handle, const uint32_t baud_mode)

5.7.1 Function Documentation

5.7.1.1 APDM_EXPORT int apdm_ds_get_case_id (apdm_device_handle_t *device_handle*, char * *BYTE_ARRAY*, const int *dest_buffer_length*)

Parameters

device_handle	The docking station handle
BYTE_ARRAY	Destination into which to store the Case ID String
dest_buffer_length	The length of the buffer to which dest_buffer is pointing

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.7.1.2 APDM_EXPORT int apdm_ds_get_docked_module_id (apdm_device_handle_t *device_handle*, uint32_t * *dest*)

Gets the Module ID that the dock things is currently placed in the dock.

Parameters

device_handle	The docking station handle
**dest*	Destination into which you want the module stored into.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_apply_autoconfigure_sensor_config().

5.7.1.3 APDM_EXPORT int apdm_ds_get_firmware_version (apdm_device_handle_t *device_handle*, uint64_t * *dest*)

Note only works in firmware/bootloaders after Nov 8, 2010

Parameters

device_handle	The docking station handle
**dest*	The destination into which to store the dock firmware version

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.7.1.4 APDM_EXPORT int apdm_ds_get_hardware_version (apdm_device_handle_t *device_handle*, uint32_t * *dest*)

Note this only works in firmware/bootloaders after Nov 8, 2010

Parameters

device_handle	The docking station handle
∗dest	The destination into which to store the hardware revision number of the dock

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.7.1.5 APDM_EXPORT int apdm_ds_get_index_by_serial_number (const uint32_t *serial_number,* uint32_t ∗ *docking_station_index*)

This will return the index of the of the docking station with with the specified serial number.

Parameters

serial_number	The serial number of the docking station for which you want the index of.
∗docking_station_index	The destination into which you want the index to be stored.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_ds_get_serial_number_by_index(), apdm_log_error(), and apdm_sensor_get_num_attached_dockingstations1().

5.7.1.6 APDM_EXPORT int apdm_ds_get_protocol_subversion (apdm_device_handle_t *device_handle,* int64_t ∗ *dest_protocol_subversion*)

Parameters

device_handle	The device handle
∗dest_protocol_subversion	The destination into which to store the protocol version

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.7.1.7 APDM_EXPORT int apdm_ds_get_serial (apdm_device_handle_t *device_handle*, uint32_t * *serial_number*)

Parameters

device_handle	The docking station handle for which you want the serial number
serial_number	Destination into which to store the dock serial number

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_warning().

5.7.1.8 APDM_EXPORT int apdm_ds_get_serial_number_by_index (const int *docking_station_index*, uint32_t * *serial_number*)

Used to retrieve the serial number of a given docking station index number.

Parameters

docking_station_index	Index of the docking station for which you want the serial number
serial_number	Destination into which the serial number will be stored

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ds_get_index_by_serial_number().

5.7.1.9 APDM_EXPORT int apdm_ds_is_monitor_data_forwarding_enabled (apdm_device_handle_t *device_handle*, uint32_t * *output_flag*)

Parameters

device_handle	The device handle
*output_flag	Destination into which to store the current status of weather or not data forwarding is enabled, zero indicates data is not being forwarded, non-zero indicates it is being forwarded

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_error().

5.7.1.10 APDM_EXPORT int apdm_ds_is_monitor_present (apdm_device_handle_t *device_handle*, uint32_t * *output_flag*)

Parameters

device_handle	The docking station handle
*output_flag	The destination into which to store the indicator as to weather or not there is an monitor present in the dock, zero indicates dock is empty, non-zero indicates a monitor is present.

References apdm_log_error().

5.7.1.11 APDM_EXPORT int apdm_ds_override_minimum_supported_version (const uint64_t *new_version*)

Allows you to override the minimum docking station version number used to validate dock versions.

Parameters

new_version	Version number, e.g. 20100902170629 Set this to zero to use library default version number.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.7.1.12 APDM_EXPORT int apdm_ds_set_monitor_baud_rate (apdm_device_handle_t *ds_handle*, const uint32_t *baud_mode*)

Parameters

device_handle	The handle to the device to apply the configuration to.
baud_mode	0 to disable high speed baud rates (default), 57600 to enable 57600 baud rate auto negotiation

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_log_debug(), apdm_log_error(), and apdm_usleep().

5.7.1.13 APDM_EXPORT int apdm_sensor_get_num_attached_dockingstations1 (uint32_t * *dest_num_docks*)

Parameters

*dest_num_docks	Destination into which to store the number of docking stations attached to the host

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ds_get_index_by_serial_number(), and apdm_halt_all_attached_sensors().

5.8 DataHandling

Functions

- APDM_EXPORT uint64_t apdm_calculate_sync_value_age (const uint64_t sync_newer, const uint64_t sync_older)
- APDM_EXPORT uint64_t apdm_epoch_access_point_to_epoch_second (const uint64_t sync_value)
- APDM_EXPORT uint64_t apdm_epoch_access_point_to_epoch_millisecond (const uint64_t sync_value)
- APDM_EXPORT int apdm_epoch_access_point_to_epoch_microsecond (const uint64_t sync_value, struct timeval ∗dest)
- APDM_EXPORT uint64_t apdm_epoch_second_to_epoch_access_point (const uint64_t epochSecond)
- APDM_EXPORT int apdm_recalibrate_magnetometers_from_h5 (char ∗file, double local_field_magnitude, uint8_t ∗calibration_block, double ∗uncalibrated_data, double ∗calibrated_data, int32_t ∗num_samples)
- APDM_EXPORT int apdm_recalibrate_gyroscopes_from_h5 (char ∗file, uint8_t ∗calibration_block)
- int apdm_extract_next_sample_set (apdm_ctx_t context, const bool checked_all_aps)

5.8.1 Function Documentation

5.8.1.1 APDM_EXPORT uint64_t apdm_calculate_sync_value_age (const uint64_t *sync_newer,* const uint64_t *sync_older*)

Calculates the time delta in milliseconds between two sync values.

Parameters

sync_newer	The larger of the two sync values
sync_older	The smaller of the two sync values

Returns

The number of milliseconds delta between the two passed synced value.

Referenced by apdm_ctx_get_wireless_reliability_value(), and apdm_extract_next_sample_set().

5.8.1.2 APDM_EXPORT int apdm_epoch_access_point_to_epoch_microsecond (const uint64_t *sync_value,* struct timeval * *dest*)

Converts a sync value to an epoch second and microseconds (point in time, as since 1970, that the sample was taken).

Parameters

sync_value	The sync value
**dest*	Destination timeval struct into which to store the time,

Returns

The corresponding epoch second and microseconds for the passed sync value (point in time, since 1970, that the sample was taken).

5.8.1.3 APDM_EXPORT uint64_t apdm_epoch_access_point_to_epoch_millisecond (const uint64_t *sync_value*)

Converts a sync value to an epoch millisecond (point in time, as number of milliseconds since 1970, that the sample was taken).

Parameters

sync_value	The sync value

Returns

The corresponding epoch millisecond for the passed sync value (point in time, as number of milliseconds since 1970, that the sample was taken).

5.8.1.4 APDM_EXPORT uint64_t apdm_epoch_access_point_to_epoch_second (const uint64_t *sync_value*)

Converts a sync value to an epoch second.

Parameters

sync_value	The sync value

Returns

The corresponding epoch second for the passed sync value.

5.8.1.5 APDM_EXPORT uint64_t apdm_epoch_second_to_epoch_access_point (const uint64_t *epochSecond*)

Helper function to convert an epoch second to a sync-value

Parameters

epochSecond	Number of seconds since 1970, unix time.

Returns

The system sync value that represents that point int time.

5.8.1.6 int apdm_extract_next_sample_set (apdm_ctx_t *context,* const bool *checked_all_aps*)

This function will inspect the head elements of the correlation fifos and assemble a set of samples all of which have the same sync value and store that into a context-specific data structure.

The resulting list may not necessarily contain a sample from all devices, as data may have been dropped due to wireless issues, or max latency thresholds could have been exceeded while waiting for a given sensor's data.

Parameters

context	

Returns

APDM_OK on success, error code otherwise.

References apdm_record_t::accl_x_axis, apdm_calculate_sync_value_age(), apdm_ctx_estimate_now_sync_value(), apdm_ctx_get_expected_number_of_sensors2(), apdm_log_debug(), apdm_log_error(), apdm_log_warning(), apdm_strerror(), apdm_record_t::device_info_serial_number, apdm_record_t::source_ap_index, and apdm_record_t::sync_val32_low.

Referenced by apdm_ctx_get_next_access_point_record_list().

5.8.1.7 APDM_EXPORT int apdm_recalibrate_gyroscopes_from_h5 (char * *file*, uint8_t * *calibration_block*)

Recalibrates the gyroscopes for bias shifts.

Parameters

file	HDF5 file containing raw and calibrated data during a period of up to 5 minutes of sitting still on a table.
calibration_block	Byte array to be populated with the updated calibration data. Must be 2048 bytes.

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_get_hdf_dataset_shape(), apdm_get_hdf_device_list(), apdm_log_debug(), apdm_log_error(), apdm_read_hdf_calibration_data(), apdm_read_hdf_dataset(), calibration_v6_t::gyro_y_bias, and calibration_v6_t::gyro_z_bias.

5.8.1.8 APDM_EXPORT int apdm_recalibrate_magnetometers_from_h5 (char * *file*, double *local_field_magnitude*, uint8_t * *calibration_block*, double * *uncalibrated_data*, double * *calibrated_data*, int32_t * *num_samples*)

Recalibrates the magenteometers for bias shifts due to magnetization of monitor componenets.

Parameters

mag_recalibration	Structure containing the fields below:
file	HDF5 file containing raw and calibrated data during a period of up to 5 minutes of rotation covering as much of the orientation space as possible in a uniform magnetic field
local_field_magnitude	Local magnetic field strength, usually obtained through a geomagnetic model like (http://www.ngdc.noaa.gov/geomagmodels/IGRFWMM.jsp). Set to 0 to use the same value as last time the monitor was calibrated.
calibration_block	Byte array to be populated with the updated calibration data. Must be 2048 bytes.
original_calibration	Output array containing the raw data used during recalibration calibrated with the original calibration data. 3∗num_samples, column major ordering
recalibrated	Output array containing the raw data used during recalibration calibrated with the updated calibration data. 3∗num_samples, column major ordering

num_samples	Number of samples in the calibrated data arrays (acutal array is 3 times larger). Updated with the number of samples actually written to the arrays. Should be at least 38400 (5 minutes)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.9 Logging

Functions

- APDM_EXPORT int apdm_set_log_level (int log_level)
- APDM_EXPORT const char * apdm_logging_level_t_str (const apdm_logging_level_t level)
- APDM_EXPORT int apdm_set_log_file (const char *filePath)
- APDM_EXPORT int apdm_close_log_file (void)
- APDM_EXPORT int apdm_log (const char *format,...)
- APDM_EXPORT int apdm_logl (const enum APDM_Logging_Level level, const char *format,...)
- APDM_EXPORT int apdm_log_debug (const char *format,...)
- APDM_EXPORT int apdm_log_info (const char *format,...)
- APDM_EXPORT int apdm_log_warning (const char *format,...)
- APDM_EXPORT int apdm_log_error (const char *format,...)
- APDM_EXPORT int apdm_log_context (apdm_ctx_t context, const enum APDM_Logging_Level level)

5.9.1 Function Documentation

5.9.1.1 APDM_EXPORT int apdm_close_log_file (void)

Closes the apdm log file (if its open)

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_set_log_file().

5.9.1.2 APDM_EXPORT int apdm_log (const char * *format*, ...)

Adds log message to the apdm log stream at DEBUG level.

Parameters

format	printf-style format string
...	var-args for printf-style values

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.9.1.3 APDM_EXPORT int apdm_log_context (apdm_ctx_t *context,* const enum APDM_Logging_Level *level*)

Logs all the details about the passed context to the apdm logging stream

Parameters

context	The context to be logged
level	The logging severity level to log as

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_logl(), apdm_monitor_decimation_rate_t_str(), apdm_output_select-_rate_t_str(), apdm_wireless_mode_t_str(), apdm_device_info_t::decimation_factor, apdm_device_info_t::dock_id_during_configuration, apdm_device_info_t::erase_sd-_card_after_undocking, apdm_device_info_t::protocol_version, apdm_device_info-_t::selected_temperature_sensor, apdm_device_info_t::wireless_addr_id, apdm_-device_info_t::wireless_block0, apdm_device_info_t::wireless_block1, apdm_device-_info_t::wireless_block2, apdm_device_info_t::wireless_block3, apdm_device_info_-t::wireless_channel1, apdm_device_info_t::wireless_channel2, apdm_device_info_t-::wireless_channel3, and apdm_device_info_t::wireless_timeslice.

Referenced by apdm_ctx_open_all_access_points(), apdm_ctx_restore_context_from-_disk(), and apdm_ctx_sync_record_list_head().

5.9.1.4 APDM_EXPORT int apdm_log_debug (const char * *format,* ...)

Adds log message to the apdm log stream at DEBUG level.

Parameters

format	printf-style format string
...	var-args for printf-style values

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ap_connect(), apdm_ap_get_id(), apdm_ap_get_version_string(), apdm_apply_autoconfigure_sensor_config(), apdm_convert_h5_to_csv(), apdm_ctx_get_next_access_point_record(), apdm_ctx_get_next_record2(), apdm_ctx_open_all_access_points(), apdm_ctx_populate_buffers(), apdm_ctx_sync_record_list_head(), apdm_ds_set_monitor_baud_rate(), apdm_extract_next_sample_set(), apdm_find_first_and_last_common_samples(), apdm_halt_all_attached_sensors(), apdm_process_raw3(), apdm_read_hdf_calibration_data(), apdm_read_hdf_dataset(), apdm_read_hdf_timestamps(), apdm_read_raw_file_info(), apdm_recalibrate_gyroscopes_from_h5(), apdm_send_accesspoint_cmd(), apdm_sensor_allocate_handle(), apdm_sensor_cmd_case_id(), apdm_sensor_cmd_sync_set(), apdm_sensor_cmd_time_get(), apdm_sensor_cmd_time_set(), apdm_sensor_config_get_label(), apdm_sensor_config_set_label(), apdm_sensor_free_handle(), and apdm_sensor_populate_device_info().

5.9.1.5 APDM_EXPORT int apdm_log_error (const char * *format*, ...)

Adds log message to the apdm log stream at ERROR level.

Parameters

format	printf-style format string
...	var-args for printf-style values

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by adpm_ap_get_minimum_sync_value(), adpm_ap_set_minimum_sync_value(), apdm_ap_get_case_id(), apdm_ap_get_io_value(), apdm_ap_get_mode(), apdm_ap_get_monitor_latency(), apdm_ap_set_io_value(), apdm_ap_verify_supported_version(), apdm_apply_autoconfigure_sensor_config(), apdm_configure_accesspoint(), apdm_convert_h5_to_csv(), apdm_ctx_allocate_new_context(), apdm_ctx_ap_get_io_value(), apdm_ctx_ap_set_io_value(), apdm_ctx_extract_next_sample(), apdm_ctx_free_context(), apdm_ctx_get_ap_id_for_ap_index(), apdm_ctx_get_device_index_by_id3(), apdm_ctx_get_monitor_latency(), apdm_ctx_get_next_access_point_record(), apdm_ctx_get_next_access_point_record_list(), apdm_ctx_get_next_record2(), apdm_ctx_get_num_samples_collected(), apdm_ctx_get_num_samples_collected_from_device(), apdm_ctx_get_sensor_compensation_data(), apdm_ctx_is_more_data_immediately_available(), apdm_ctx_open_all_access_points(), apdm_ctx_persist_context_to_disk(), apdm_ctx_restore_context_from_disk(), apdm_ctx_set_max_sample_delay_seconds(), apdm_ctx_set_sensor_compensation_data(), apdm_ctx_sync_record_list_head(), apdm_ds_get_case_id(), apdm_ds_get-

_firmware_version(), apdm_ds_get_index_by_serial_number(), apdm_ds_is_monitor-_data_forwarding_enabled(), apdm_ds_is_monitor_present(), apdm_ds_set_monitor-_baud_rate(), apdm_extract_next_sample_set(), apdm_halt_all_attached_sensors(), apdm_process_raw3(), apdm_read_raw_file_info(), apdm_recalibrate_gyroscopes_-from_h5(), apdm_send_accesspoint_cmd(), apdm_sensor_allocate_handle(), apdm_-sensor_cmd_enter_bootloader(), apdm_sensor_cmd_memory_dump(), apdm_sensor-_cmd_sample_get(), apdm_sensor_config_get_label(), apdm_sensor_config_set_-label(), apdm_sensor_populate_device_info(), and apdm_sensor_verify_supported_-version().

5.9.1.6 APDM_EXPORT int apdm_log_info (const char * *format*, ...)

Adds log message to the apdm log stream at INFO level.

Parameters

format	printf-style format string
...	var-args for printf-style values

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_autoconfigure_mesh_sync(), apdm_autoconfigure_mesh_-sync2(), apdm_configure_accesspoint(), apdm_ctx_allocate_new_context(), apdm_-ctx_open_all_access_points(), apdm_ctx_restore_context_from_disk(), apdm_halt_all-_attached_sensors(), and apdm_sensor_get_device_id_list().

5.9.1.7 APDM_EXPORT int apdm_log_warning (const char * *format*, ...)

Adds log message to the apdm log stream at WARNING level.

Parameters

format	printf-style format string
...	var-args for printf-style values

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_ap_connect(), apdm_ap_get_case_id(), apdm_ctx_get_next-_access_point_record(), apdm_ds_get_serial(), apdm_extract_next_sample_set(), apdm_free_ap_handle(), apdm_process_raw3(), and apdm_read_hdf_calibration_-data().

5.9.1.8 APDM_EXPORT const char* apdm_logging_level_t_str (const apdm_logging_level_t *level*)

Parameters

level	The level for which you want the string representation

Returns

Pointer to string for the given log level

5.9.1.9 APDM_EXPORT int apdm_logl (const enum APDM_Logging_Level *level,* const char * *format,* ...)

Adds a log message to the adpm loc stream at the given loglevel using the format and args passed in.

Parameters

level	The log level that the message should be logged at.
format	printf-style format string
...	var-args for printf-style values

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

Referenced by apdm_log_context().

5.9.1.10 APDM_EXPORT int apdm_set_log_file (const char * *filePath*)

Sets and opens a log file to be used by APDM libraries for logging purposes

Parameters

filePath	The file to which logging data should be saved

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

References apdm_close_log_file().

5.9.1.11 APDM_EXPORT int apdm_set_log_level (int *log_level*)

Sets the current log level to be used by APDM libraries. Valid values are:

Parameters

log_level	Valid values are: APDM_LL_ALL = 0, APDM_LL_DEBUG = 1, APDM_LL_INFO = 2, APDM_LL_WARNING = 3, APDM_LL_ERROR = 4, APDM_LL_NONE = 5

Returns

APDM_OK on success, error code from 'enum APDM_Status' in apdm_types.h

5.10 Misc

Functions

- APDM_EXPORT apdm_device_info_t * apdm_streaming_config_get_device_info (apdm_streaming_config_t *streaming_config, int sensor_index)
- APDM_EXPORT const char * apdm_monitor_error_id_str (const apdm_monitor_error_id_t error_id)
- APDM_EXPORT const char * apdm_get_library_version (void)
- APDM_EXPORT const char * apdm_get_library_build_datetime (void)
- APDM_EXPORT uint64_t apdm_get_time_ms_64 (struct timeval *dest)
- APDM_EXPORT const char * apdm_strerror (const enum APDM_Status status_code)
- APDM_EXPORT const char * apdm_output_select_rate_t_str (const apdm_monitor_output_select_rate_t rate)
- APDM_EXPORT const char * apdm_monitor_decimation_rate_t_str (const apdm_monitor_decimation_rate_t rate)
- APDM_EXPORT uint32_t apdm_monitor_output_select_rate_t_to_int (const apdm_monitor_output_select_rate_t rate)
- APDM_EXPORT uint32_t apdm_monitor_get_expected_sync_delta (const apdm_monitor_output_select_rate_t rate)
- APDM_EXPORT uint32_t apdm_monitor_decimation_rate_t_to_int (const apdm_monitor_decimation_rate_t rate)
- APDM_EXPORT const char * apdm_wireless_mode_t_str (const apdm_wireless_mode_t mode)
- APDM_EXPORT enum APDM_Status_Severity apdm_error_severity (const int status)
- APDM_EXPORT void apdm_usleep (const uint64_t microseconds)
- APDM_EXPORT void apdm_msleep (const uint64_t milliseconds)
- uint64_t apdm_get_now_sync_value_host (void)

5.10.1 Function Documentation

5.10.1.1 APDM_EXPORT enum APDM_Status_Severity apdm_error_severity (const int *status*)

Helper function to get the severity level of a given apdm status code (APDM_Status)

Parameters

status	The APDM_Status status code in question

Returns

APDM_SEVERITY_ERROR, APDM_SEVERITY_WARNING or APDM_SEVERITY_INFO depending on the respective error severity.

5.10.1.2 APDM_EXPORT const char* apdm_get_library_build_datetime (void)

Returns

The date on which the libraries were built.

5.10.1.3 APDM_EXPORT const char* apdm_get_library_version (void)

Returns

The version of the host libraries currently being used

5.10.1.4 uint64_t apdm_get_now_sync_value_host (void)

Returns

the sync value for 'now', based on the host computers current clock time. Note: this does not account for clock drift errors between the host computer and the access point.

References apdm_get_time_ms_64().

Referenced by apdm_ctx_estimate_now_sync_value().

5.10.1.5 APDM_EXPORT uint64_t apdm_get_time_ms_64 (struct timeval * *dest*)

Returns

the number of milliseconds elapsed since the UNIX epoch. Works on both windows and linux.

Referenced by apdm_ap_connect(), apdm_ap_get_version_string(), apdm_ctx_open_all_access_points(), apdm_ctx_sync_record_list_head(), apdm_get_now_sync_value_host(), apdm_send_accesspoint_cmd(), and apdm_sensor_populate_device_info().

5.10.1.6 APDM_EXPORT const char* apdm_monitor_decimation_rate_t_str (const apdm_monitor_decimation_rate_t *rate*)

Parameters

rate	The apdm_monitor_decimation_rate_t for which you want the string representation.

Returns

The string representation of the rate passed in.

Referenced by apdm_log_context().

5.10.1.7 APDM_EXPORT uint32_t apdm_monitor_decimation_rate_t_to_int (const apdm_monitor_decimation_rate_t *rate*)

Parameters

rate	The apdm_monitor_decimation_rate_t for which you want the numerical decimation rate.

Returns

The decimation rate, numerical, for the specified rate, E.G. APDM_DECIMATE_5x2 maps to 10

Referenced by apdm_initialize_device_info().

5.10.1.8 APDM_EXPORT const char* apdm_monitor_error_id_str (const apdm_monitor_error_id_t *error_id*)

Parameters

error_id	The error ID, of type, apdm_motion_monitor_error_id_t, for which you want a string representation

Returns

Const char* pointing to string representation of the given error ID.

5.10.1.9 APDM_EXPORT uint32_t apdm_monitor_get_expected_sync_delta (const apdm_monitor_output_select_rate_t *rate*)

Parameters

rate	For a given output rate, determine the expected sync delta between any two samples.

Returns

The expected sync delta

References apdm_monitor_output_select_rate_t_to_int().

5.10.1.10 APDM_EXPORT uint32_t apdm_monitor_output_select_rate_t_to_int (const apdm_monitor_output_select_rate_t *rate*)

Parameters

rate	The apdm_monitor_output_select_rate_t for which you want the numerical output rate.

Returns

The output sample rate of the specified apdm_monitor_output_select_rate_t, e.g APDM_OUTPUT_SELECT_RATE_128 maps to 128

Referenced by apdm_initialize_device_info(), and apdm_monitor_get_expected_sync_delta().

5.10.1.11 APDM_EXPORT void apdm_msleep (const uint64_t *milliseconds*)

Platform independent version of msleep().

Parameters

milliseconds	Number of milliseconds to sleep for

5.10.1.12 APDM_EXPORT const char* apdm_output_select_rate_t_str (const apdm_monitor_output_select_rate_t *rate*)

Parameters

rate	The apdm_monitor_output_select_rate_t for which you want the string representation

Returns

The string representation of the rate passed in.

Referenced by apdm_log_context().

5.10.1.13 APDM_EXPORT apdm_device_info_t* apdm_streaming_config_get_device_info (apdm_streaming_config_t * *streaming_config*, int *sensor_index*)

Helper method to retreive a reference to an apdm_device_info_t structure. Useful for the Java SWIG binding.

Parameters

**streaming_config*	Pointer to apdm_streaming_config_t structure to be used
int	sensor_index The index into the array of apdm_device_info_t structures

Returns

Pointer to the corresponding apdm_device_info_t structure.

References apdm_streaming_config_t::device_info_cache.

5.10.1.14 APDM_EXPORT const char* apdm_strerror (const enum APDM_Status *status_code*)

Helper function to convert an apdm status code to a string.

Parameters

status_code	The status code for which you want the string representation.

Returns

Pointer to a char array with a string representation of the status code.

Referenced by apdm_ap_connect(), apdm_ctx_get_next_access_point_record(), apdm_ctx_get_next_access_point_record_list(), apdm_ctx_get_num_samples_collected(), apdm_ctx_get_num_samples_collected_from_device(), apdm_ctx_open_all_access_points(), apdm_extract_next_sample_set(), apdm_halt_all_attached_sensors(), apdm_sensor_get_device_id_list(), and apdm_sensor_populate_device_info().

5.10.1.15 APDM_EXPORT void apdm_usleep (const uint64_t *microseconds*)

Helper function for working with HDF5 files. Reads all of the annotations stored in the .h5 file.

Parameters

file	The .h5 file to load data from
annotations	Array of apdm_annotation_t structures containing the annotations. If NULL, only nAnnotations is set.
nAnnotations	Number of annotations in the file.

Returns

APDM_OK on success Platform independent version of usleep().

Parameters

microseconds	Number of microseconds to sleep for

Referenced by apdm_ctx_sync_record_list_head(), and apdm_ds_set_monitor_baud_rate().

5.10.1.16 APDM_EXPORT const char* apdm_wireless_mode_t_str (const apdm_wireless_mode_t *mode*)

Parameters

mode	The apdm_wireless_mode_t for which you want the string representation of.

Returns

The string representation of the specified mode.

Referenced by apdm_log_context().

Chapter 6

Class Documentation

6.1 __attribute__ Struct Reference

Public Member Functions

- union {
  int64_t **calibration_version**
  apdm_calibration_data_v4_t **v4**
  apdm_calibration_data_v5_t **v5**
  apdm_calibration_data_v6_t **v6**
  uint8_t **raw** [256]
  } **__attribute__** ((__packed__)) data

Public Attributes

- int64_t **accl_x_bias**
- int64_t **accl_y_bias**
- int64_t **accl_z_bias**
- int64_t **accl_x_bias_temp**
- int64_t **accl_y_bias_temp**
- int64_t **accl_z_bias_temp**
- int64_t **accl_x_scale**
- int64_t **accl_y_scale**
- int64_t **accl_z_scale**
- int64_t **accl_x_scale_temp**
- int64_t **accl_y_scale_temp**
- int64_t **accl_z_scale_temp**

- int64_t **accl_xy_sensitivity**
- int64_t **accl_xz_sensitivity**
- int64_t **accl_yz_sensitivity**
- int64_t **gyro_x_bias**
- int64_t **gyro_y_bias**
- int64_t **gyro_z_bias**
- int16_t **gyro_z_bias_temp** [61]
- int16_t **nothing** [3]
- int64_t **gyro_x_bias_temp**
- int64_t **gyro_x_bias_temp2**
- int64_t **gyro_y_bias_temp**
- int64_t **gyro_y_bias_temp2**
- int64_t **gyro_x_scale**
- int64_t **gyro_y_scale**
- int64_t **gyro_z_scale**
- int64_t **gyro_x_scale_temp**
- int64_t **gyro_y_scale_temp**
- int64_t **gyro_z_scale_temp**
- int64_t **gyro_xy_sensitivity**
- int64_t **gyro_xz_sensitivity**
- int64_t **gyro_yz_sensitivity**
- int64_t **gyro_accl_roll**
- int64_t **gyro_accl_pitch**
- int64_t **gyro_accl_yaw**
- int64_t **mag_xy_sensitivity**
- int64_t **mag_xz_sensitivity**
- int64_t **mag_yz_sensitivity**
- int64_t **mag_x_bias**
- int64_t **mag_y_bias**
- int64_t **mag_z_bias**
- int64_t **mag_x_scale**
- int64_t **mag_y_scale**
- int64_t **mag_z_scale**
- int64_t **mag_accl_roll**
- int64_t **mag_accl_pitch**
- int64_t **mag_accl_yaw**
- int64_t **temperature_bias**
- int64_t **temperature_scale**
- int64_t **accl_z_dtemp_scale**
- int64_t **temperature_bias_msp**
- int64_t **temperature_scale_msp**
- int64_t **cal_version**
- uint16_t **accl_x_bias** [61]

- uint16_t **accl_y_bias** [61]
- uint16_t **accl_z_bias** [61]
- int16_t **nothing0**
- uint16_t **gyro_x_bias** [61]
- uint16_t **gyro_y_bias** [61]
- uint16_t **gyro_z_bias** [61]
- int16_t **nothing1**
- int64_t **mag_x_scale_temp**
- int64_t **mag_y_scale_temp**
- int64_t **mag_z_scale_temp**
- uint16_t **mag_x_bias** [61]
- uint16_t **mag_y_bias** [61]
- uint16_t **mag_z_bias** [61]
- int16_t **nothing2**
- int64_t **mag_x_offset**
- int64_t **mag_y_offset**
- int64_t **mag_z_offset**
- int64_t **mag_conversion_gain**
- int64_t **mag_accl_x**
- int64_t **mag_accl_y**
- int64_t **mag_accl_z**
- int64_t **mag_inclination**
- uint32_t **cal_version**
- uint32_t **device_id**
- uint8_t **retrys**
- uint16_t **event_id**
- union {
  } **packet**

- uint16_t **ax**
- uint16_t **ay**
- uint16_t **az**
- uint16_t **gx**
- uint16_t **gy**
- uint16_t **gz**
- uint16_t **mx**
- uint16_t **my**
- uint16_t **mz**
- uint16_t **t**
- uint32_t **data_a**
- uint32_t **data_b**
- uint32_t **data_c**
- uint32_t **data_d**

- uint32_t **data_e**
- uint32_t **data_f**
- uint32_t **error_id**
- uint32_t **error_count**
- uint32_t **sync_low_32**
- uint32_t **sync_high_32**
- uint32_t **pre_block**
- uint32_t **post_block**
- uint32_t **samples_per_block**
- uint32_t **max_latency**
- uint64_t **pre_sync**
- uint64_t **post_sync**

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

- apdm_internal.h

6.2 apdm_access_point_configuration_t Struct Reference

Public Attributes

- uint8_t **radio1_channel**
- uint8_t **radio2_channel**
- uint32_t **address_blockA**
- uint32_t **address_blockB**
- uint32_t **radio1_pipe_count**
- uint32_t **radio2_pipe_count**
- uint64_t **sync_value_subtractor**
- uint32_t **single_ap_mode**
- uint32_t **id**
- uint32_t **board_version**
- uint32_t **sensor_group**
- uint16_t **ap_group**
- char **case_id** [64]
- uint32_t **pipe_mappings** [NUM_PIPES_IN_PIPE_MAPPING]

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

- apdm_internal.h

6.3 apdm_access_point_handle Struct Reference

`#include <apdm_internal.h>`

Public Attributes

- struct libusb_device_handle * **devh**
- uint8_t **usb_protocol_version**
- uint64_t **usb_protocol_subversion**
- bool **has_flushed_data**
- bool **has_skipped_first_sample**
- apdm_bulk_in_buffer_t **ep_in_buffer**
- apdm_bulk_in_buffer_t **ep_in_binary_buffer**
- uint8_t **sensor_group**
- uint32_t **num_remaining_samples**
- uint64_t **current_ap_sync_value_64**
- int64_t **sync_value_clock_modifier**
- uint32_t **current_ap_sample_counter**
- apdm_ap_wireless_streaming_status_t **current_led_streaming_status**
- uint32_t **sample_number**
- uint32_t **total_samples_collected**
- int32_t **data_array_start_idx**
- int32_t **data_array_end_idx**
- apdm_record_t **data_array** [DA_SIZE]
- int32_t **sync_data_array_head**
- int32_t **sync_data_array_tail**
- apdm_external_sync_data_t **sync_data_array** [SYNC_DATA_ARRAY_SIZE]
- int32_t **opal_event_data_array_head**
- int32_t **opal_event_data_array_tail**
- apdm_opal_event_packet_t **opal_event_packet_data_array** [MONITOR_EVENT_ARRAY_SIZE]
- struct timeval **last_read_time**
- uint64_t **ap_firmware_ver**
- char **ap_firmware_version** [1024]
- apdm_access_point_configuration_t **ap_configuration**

6.3.1 Detailed Description

This structure maps to the client programmer data type of: typedef void* apdm_ap_handle_t;

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

- apdm_internal.h

6.4 apdm_annotation_t Struct Reference

Public Attributes

- uint64_t **time**
- uint32_t **device_id**
- char **text** [2048]

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

- apdm_types.h

6.5 apdm_bulk_in_buffer_t Struct Reference

Public Attributes

- char **temp_buffer** [16384]
- apdm_byte_array_ring_buffer_t **ring_buffer**

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

- apdm_internal.h

6.6 apdm_byte_array_ring_buffer_t Struct Reference

Public Attributes

- uint32_t **current_head**
- uint32_t **current_tail**
- char **data** [APDM_RING_BUFFER_SIZE]

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

- apdm_ring_buffer.h

6.7 apdm_case_id_t Struct Reference

Public Attributes

- char **id** [CASE_ID_SIZE]

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

- apdm_types.h

6.8 apdm_context_t Struct Reference

`#include <apdm_internal.h>`

Public Attributes

- uint64_t **context_library_version_number**
- uint32_t **num_configured_aps**
- apdm_access_point_handle **ap_handle_list** [APDM_MAXIMUM_NUM_ACCESS_POINTS]
- uint64_t **last_correlation_fifo_population_time_ms**
- per_device_info_t **sensor_list** [APDM_MAX_NUMBER_OF_SENSORS]
- uint32_t **num_compenstation_entries**
- int32_t **temp**
- enum APDMErrorHandlingBehavior **error_handling_behavior**
- uint16_t **max_data_delay_seconds**
- uint64_t **minimum_sync_value**
- uint32_t **expected_sync_delta**
- uint32_t **total_sample_lists_collected**
- bool **has_returned_full_sample_set_flag**
- uint8_t **temp_buff** [DEFAULT_READ_SIZE ∗2]
- apdm_wireless_mode_t **wireless_configuration_mode**
- uint32_t **initial_sample_retrevial_count**
- uint32_t **num_omitted_sample_sets**
- uint32_t **total_omitted_sample_sets**
- uint32_t **num_omitted_samples**
- uint32_t **total_omitted_samples**
- bool **more_data_available_flag**
- uint64_t **last_found_sync_value**
- uint64_t **last_returned_sample_list_sync_value**
- apdm_device_sample_buffer_row_t **most_recent_list**

6.8.1 Detailed Description

This structure maps to the client programmer data type of: typedef void∗ apdm_ctx_t;

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

- apdm_internal.h

6.9 apdm_conversion_parameter_t Struct Reference

Public Attributes

- FILE * **fin**
- apdm_device_info_t * **info**
- FILE * **fout_csv**
- hid_t **fout_hdf**
- bool **store_raw**
- bool **store_si**
- bool **store_filtered**
- bool **compress**
- char **csv_delimiter**
- apdm_progress_t * **progress**
- bool **dechop_raw_magnetometer**
- uint64_t **sync_start**
- uint64_t **sync_end**
- apdm_orientation_model_t **orientation_model**
- char **timezone_string** [TIMEZONE_STRING_SIZE]

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

- apdm_internal.h

6.10 apdm_device_dtemp_filter_state_t Struct Reference

Public Attributes

- double **state** [2]
- double state_transition_matrix [4]
- double process_noise_matrix [4]
- double measurement [1]
- double measurement_matrix [2]
- double measurement_noise_matrix [1]
- double error_covariance_matrix [4]
- double filtered_measurement [1]

6.10.1 Member Data Documentation

6.10.1.1 double apdm_device_dtemp_filter_state_t::error_covariance_matrix[4]

6.10.1.2 double apdm_device_dtemp_filter_state_t::filtered_measurement[1]

6.10.1.3 double apdm_device_dtemp_filter_state_t::measurement[1]

6.10.1.4 double apdm_device_dtemp_filter_state_t::measurement_matrix[2]

initialize to [0;0;0;0;0;0;0];

6.10.1.5 double apdm_device_dtemp_filter_state_t::measurement_noise_matrix[1]

6.10.1.6 double apdm_device_dtemp_filter_state_t::process_noise_matrix[4]

initialize to [0 0; 0 1]

6.10.1.7 double apdm_device_dtemp_filter_state_t::state_transition_matrix[4]

initialize to [0 0];

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

- apdm_internal.h

6.11 apdm_device_fifo_t Struct Reference

Public Attributes

- int32_t **head**
- int32_t **tail**
- apdm_record_t **fifo_data** [MAX_SAMPLES_THAT_A_DEVICE_CAN_BUFFER]

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

- apdm_internal.h

6.12 apdm_device_info_t Struct Reference

Public Attributes

- bool **decimation_bypass_flag**
- bool **time_good_flag**
- bool **accelerometer_full_scale_flag**
- bool **accelerometer_enabled_flag**
- bool **gyroscope_enabled_flag**
- bool **magnetometer_enabled_flag**
- bool **sd_card_enabled_flag**
- bool **always_off_flag**
- bool erase_sd_card_after_undocking
- bool **enable_button**
- uint8_t **button_mode**
- apdm_monitor_spin_mode_t **spin_mode**
- uint8_t selected_temperature_sensor
- apdm_monitor_decimation_rate_t **decimation_rate**
- apdm_monitor_output_select_rate_t **output_select_rate**
- uint16_t **sample_rate**
- uint32_t decimation_factor
- int32_t timezone
- char **device_label** [DEVICE_LABEL_SIZE]
- apdm_orientation_model_t orientation_model
- uint8_t **calibration_binary_blob** [CALIBRATION_DATA_BUFFER_SIZE]
- uint32_t **calibration_version_number**
- uint8_t **user_calibration_binary_blob** [CALIBRATION_DATA_BUFFER_SIZE]
- uint32_t **user_calibration_version_number**
- uint32_t **device_id**
- uint32_t **hardware_id**
- char **sd_file_version** [9]
- char **firmware_version_string1** [VERSION_STRING_SIZE]
- char **firmware_version_string2** [VERSION_STRING_SIZE]
- int64_t **firmware_version_string2_number**
- char **firmware_version_string3** [VERSION_STRING_SIZE]
- char **case_id** [CASE_ID_SIZE]
- char **timezone_string** [TIMEZONE_STRING_SIZE]
- apdm_config_mag_set_reset_t **magnetometer_set_reset**
- apdm_monitor_recording_mode_t **recording_mode**
- apdm_monitor_data_mode_t **data_mode**
- bool **enable_wireless**
- apdm_wireless_mode_t **wireless_protocol**
- uint8_t wireless_timeslice
- uint8_t wireless_addr_id
- uint32_t protocol_version

- uint8_t **wireless_channel0**
- uint32_t wireless_block0
- uint8_t wireless_channel1
- uint32_t wireless_block1
- uint8_t wireless_channel2
- uint32_t wireless_block2
- uint8_t wireless_channel3
- uint32_t wireless_block3
- uint32_t dock_id_during_configuration
- uint32_t **dock_hardware_version_during_configuration**

6.12.1 Detailed Description

6.12.2 Member Data Documentation

6.12.2.1 uint32_t apdm_device_info_t::decimation_factor

E.G. 128, can be directly derived from output_select_rate, this should be set with results from apdm_monitor_output_select_rate_t_to_int()

Referenced by apdm_convert_h5_to_csv(), apdm_ctx_get_next_access_point_record(), apdm_initialize_device_info(), and apdm_log_context().

6.12.2.2 uint32_t apdm_device_info_t::dock_id_during_configuration

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context().

6.12.2.3 bool apdm_device_info_t::erase_sd_card_after_undocking

True if you want the monitor to erase the SD card after it undocks

Referenced by apdm_initialize_device_info(), and apdm_log_context().

6.12.2.4 apdm_orientation_model_t apdm_device_info_t::orientation_model

Orientation model to use

Referenced by apdm_initialize_device_info(), and apdm_process_raw3().

6.12.2.5 uint32_t apdm_device_info_t::protocol_version

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value. Defines the pipe-number that data will come in on for the monitor.

Referenced by apdm_ctx_get_next_access_point_record(), apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.6 uint8_t apdm_device_info_t::selected_temperature_sensor

(1:APDM_TEMP_SENSOR_GYRO or 0:APDM_TEMP_SENSOR_MSP)

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.7 int32_t apdm_device_info_t::timezone

E.G. 10, can be directly derived from decimation_rate, this should be set with results from apdm_monitor_decimation_rate_t_to_int() offset from UTC in minutes such that local time = UTC + timezone

Referenced by apdm_sensor_populate_device_info().

6.12.2.8 uint8_t apdm_device_info_t::wireless_addr_id

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.9 uint32_t apdm_device_info_t::wireless_block0

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.10 uint32_t apdm_device_info_t::wireless_block1

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.11 uint32_t apdm_device_info_t::wireless_block2

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.12 uint32_t apdm_device_info_t::wireless_block3

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.13 uint8_t apdm_device_info_t::wireless_channel1

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.14 uint8_t apdm_device_info_t::wireless_channel2

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.15 uint8_t apdm_device_info_t::wireless_channel3

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

6.12.2.16 uint8_t apdm_device_info_t::wireless_timeslice

CAUTION: modifying this can cause unpredictable behavior, allow autoconfigure() or other similar functions to set this value.

Referenced by apdm_log_context(), and apdm_sensor_populate_device_info().

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

- apdm_types.h

6.13 apdm_device_sample_buffer_row_t Struct Reference

Public Attributes

- bool **is_full_flag**
- bool **is_partially_populated_flag**
- uint64_t **sync_val_for_this_line**
- apdm_record_t **data_records** [APDM_MAX_NUMBER_OF_SENSORS]

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

- apdm_internal.h

6.14 apdm_device_state_data_t Struct Reference

Public Attributes

- uint32_t **device_id**
- time_t **last_received_data_timestamp**
- apdm_opal_event_packet_t **last_sync_event_received**
- uint64_t **last_received_data_sync_value**
- uint32_t **last_received_sample_count**
- double **battery_level**
- double **last_temperature**
- double **last_temperature_diff**
- uint64_t **last_processed_data_sync_value**
- bool **first_sample**
- double **temperature_derivative_buffer** [NUM_TEMPERATURE_READINGS_FOR_AVERAGING]
- int **temperature_derivative_buffer_index**
- uint64_t **sync_buffer** [NUM_TEMPERATURE_READINGS_FOR_AVERAGING]
- double **differentiator_buffer** [NUM_TEMPERATURE_READINGS_FOR_DIFFERENTIATOR]
- int **differentiator_buffer_index**
- double **mag_x_buffer** [14]
- double **mag_y_buffer** [14]
- double **mag_z_buffer** [14]
- int **mag_buffer_index**
- int **calibration_data_validated**
- apdm_orientation_info_t **orientation_info**
- int32_t **retry_count_history** [APDM_RETRY_HISTORY_LENGTH]

- uint32_t **retry_count_history_head_index**

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

- apdm_internal.h

6.15 apdm_device_status_t Struct Reference

`#include <apdm_types.h>`

Public Attributes

- int **result_code**
- uint8_t **gyro_recalibration_block** [CALIBRATION_DATA_BUFFER_SIZE]
- enum APDM_Status **gyro_recalibration_result**
- uint32_t sd_mbytes_total
- uint32_t sd_mbytes_used

6.15.1 Detailed Description

FIXME document this

6.15.2 Member Data Documentation

6.15.2.1 uint32_t apdm_device_status_t::sd_mbytes_total

Currently filled in by MotionStudio

6.15.2.2 uint32_t apdm_device_status_t::sd_mbytes_used

Currently filled in by MotionStudio

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

- apdm_types.h

6.16 apdm_disk_ll_t Struct Reference

Public Attributes

- FILE * **file_handle**
- int32_t **current_length**
- int32_t **tail_largest_idx**
- int32_t **head_smallest_idx**
- uint8_t **free_sample_list** [APDM_FREE_SAMPLE_LIST_ARRAY_SIZE]

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

- apdm_internal.h

6.17 apdm_external_sync_data_t Struct Reference

Public Attributes

- uint8_t **data**
- uint8_t data_type
- uint64_t sync_value
- uint32_t ap_id

6.17.1 Member Data Documentation

6.17.1.1 uint32_t apdm_external_sync_data_t::ap_id

ID of the access point that the event occurred on

6.17.1.2 uint8_t apdm_external_sync_data_t::data_type

value is from enum External_Sync_Data_Types

6.17.1.3 uint64_t apdm_external_sync_data_t::sync_value

Time at which the event occurred

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

- apdm_types.h

6.18 apdm_file_conversion_parameter_t Struct Reference

Public Attributes

- char ** files_to_convert
- int nFiles
- const char * file_out
- bool store_raw
- bool store_si
- bool store_filtered
- bool format_hdf
- bool compress
- char csv_delimiter
- apdm_progress_t * progress
- uint64_t sync_start
- uint64_t sync_end
- char timezone_string [TIMEZONE_STRING_SIZE]
- bool dechop_raw_magnetometer
- char ** calibration_files
- apdm_orientation_model_t **orientation_model**

6.18.1 Member Data Documentation

6.18.1.1 char** apdm_file_conversion_parameter_t::calibration_files

default NULL to indicate calibration parameters included in each file should be used

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.2 bool apdm_file_conversion_parameter_t::compress

true to compress HDF data, has no effect if output is CSV format

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.3 char apdm_file_conversion_parameter_t::csv_delimiter

delimiter character to use for csv files. Default is ','. Has no effect if output is HDF format

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.4 bool apdm_file_conversion_parameter_t::dechop_raw_magnetometer

default true

Referenced by apdm_initialize_file_conversion_parameters(), and apdm_process_raw3().

6.18.1.5 const char* apdm_file_conversion_parameter_t::file_out

output file path

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.6 char** apdm_file_conversion_parameter_t::files_to_convert

array of .apdm file name paths

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), apdm_process_raw3(), and apdm_release_conversion_parameters().

6.18.1.7 bool apdm_file_conversion_parameter_t::format_hdf

true to store output in HDF5 format, false to store output in CSV format

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.8 int apdm_file_conversion_parameter_t::nFiles

number of files to convert (size of files_in array)

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), apdm_process_raw3(), and apdm_release_conversion_parameters().

6.18.1.9 apdm_progress_t* apdm_file_conversion_parameter_t::progress

progress structure updated during the file conversion process that can be inspected by another thread for updating a progress bar

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.10 bool apdm_file_conversion_parameter_t::store_filtered

true to store filtered calibrated data

Referenced by apdm_initialize_file_conversion_parameters(), and apdm_process_raw3().

6.18.1.11 bool apdm_file_conversion_parameter_t::store_raw

true to store raw ADC data in the output file

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.12 bool apdm_file_conversion_parameter_t::store_si

true to store calibrated data in SI units

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

6.18.1.13 uint64_t apdm_file_conversion_parameter_t::sync_end

no data after sync_end will be included in the output file, default 0 indicates all data included

Referenced by apdm_initialize_file_conversion_parameters(), and apdm_process_raw3().

6.18.1.14 uint64_t apdm_file_conversion_parameter_t::sync_start

only data after sync_start will be included in the output file, default 0 indicates all data included

Referenced by apdm_initialize_file_conversion_parameters(), and apdm_process_raw3().

6.18.1.15 char apdm_file_conversion_parameter_t::timezone_string[TIMEZONE_STRING_SIZE]

Timezone string (eg. "America/Los_Angeles")

Referenced by apdm_initialize_file_conversion_parameters(), apdm_process_raw(), apdm_process_raw2(), and apdm_process_raw3().

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

- apdm_types.h

6.19 apdm_mag_dechop_state_t Struct Reference

Public Attributes

- double **state** [2]
- double state_transition_matrix [4]
- double process_noise_matrix [4]
- double measurement [6]
- double measurement_matrix [12]
- double measurement_noise_matrix [6 ∗6]
- double error_covariance_matrix [4]
- double filtered_measurement [6]
- double stepResponse [10]
- apdm_mag_step_response_state_t stepResponseEstimate
- int **set_reset_flag**
- int polarity
- int iSample

6.19.1 Detailed Description

6.19.2 Member Data Documentation

6.19.2.1 double apdm_mag_dechop_state_t::error_covariance_matrix[4]

6.19.2.2 double apdm_mag_dechop_state_t::filtered_measurement[6]

6.19.2.3 int apdm_mag_dechop_state_t::iSample

6.19.2.4 double apdm_mag_dechop_state_t::measurement[6]

6.19.2.5 double apdm_mag_dechop_state_t::measurement_matrix[12]

initialize to [0;0;0;0;0;0;0;0];

6.19.2.6 double apdm_mag_dechop_state_t::measurement_noise_matrix[6 ∗6]

6.19.2.7 int apdm_mag_dechop_state_t::polarity

6.19.2.8 double apdm_mag_dechop_state_t::process_noise_matrix[4]

initialize to [0 0; 0 1]

6.19.2.9 double apdm_mag_dechop_state_t::state_transition_matrix[4]

initialize to [0 0];

6.19.2.10 double apdm_mag_dechop_state_t::stepResponse[10]

6.19.2.11 apdm_mag_step_response_state_t apdm_mag_dechop_state_t::stepResponseEstimate

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

- apdm_types.h

6.20 apdm_mag_opt_data_t Struct Reference

Public Attributes

- double ∗ **samples**
- double ∗ **temperature**
- double ∗ **cal_samples**
- double ∗ **acc**
- double ∗ **qi**
- double **mean_angle**
- int **n**
- int **n_print**
- calibration_v6_t **sensor_comp**
- nlopt_opt **opt**

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

- apdm_internal.h

6.21 apdm_mag_step_response_state_t Struct Reference

Public Attributes

- double **state** [3]
- double state_transition_matrix [9]
- double process_noise_matrix [9]
- double measurement [3]
- double measurement_matrix [9]
- double measurement_noise_matrix [9]
- double error_covariance_matrix [9]
- double filtered_measurement [3]

6.21.1 Member Data Documentation

6.21.1.1 double apdm_mag_step_response_state_t::error_covariance_matrix[9]

6.21.1.2 double apdm_mag_step_response_state_t::filtered_measurement[3]

6.21.1.3 double apdm_mag_step_response_state_t::measurement[3]

6.21.1.4 double apdm_mag_step_response_state_t::measurement_matrix[9]

initialize to [0;0;0];

6.21.1.5 double apdm_mag_step_response_state_t::measurement_noise_matrix[9]

6.21.1.6 double apdm_mag_step_response_state_t::process_noise_matrix[9]

initialize to [1 0 0; 0 1 0; 0 0 1]

6.21.1.7 double apdm_mag_step_response_state_t::state_transition_matrix[9]

initialize to [0 0 0];

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

- apdm_types.h

6.22 apdm_magnetometer_recalibration_t Struct Reference

Public Attributes

- char * **file**
- double **local_field_magnitude**
- uint8_t **calibration_block** [2048]
- double **original_calibrated_data** [115200]
- double **updated_calibrated_data** [115200]
- int **num_samples**

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

- apdm_types.h

6.23 apdm_monitor_error_stat_t Struct Reference

Public Attributes

- apdm_monitor_error_id_t **error_id**
- uint32_t **error_count**
- uint64_t sync_value

6.23.1 Member Data Documentation

6.23.1.1 uint64_t apdm_monitor_error_stat_t::sync_value

A non-zero sync value means that this has valid data

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

- apdm_types.h

6.24 apdm_monitor_label_t Struct Reference

Public Attributes

- char **label** [DEVICE_LABEL_SIZE]

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

- apdm_types.h

6.25 apdm_orientation_info_t Struct Reference

Public Attributes

- double **x** [3]
- double **y** [6]
- apdm_orientation_model_t **model**
- double **state_transition_matrix** [9]
- double **process_noise_matrix** [9]
- double **error_covariance_matrix** [9]
- double **measurement_matrix** [36]
- double **measurement_covariance_matrix** [36]
- double **filtered_measurement** [6]
- int **state_dimension**
- int **measurement_dimension**
- double **acc_filt** [3]
- double **mag_filt** [3]
- double **mt** [3]
- double **acc_filt_state** [9]
- double **mag_filt_state** [9]
- double **filt_state_template** [3]
- double **sample_rate**
- double **mag_mag**
- double **acc_mag**
- double **acc_var**
- double **gyro_var**
- double **mag_var**
- double **mag_inclination**
- double **fs**
- double **filter_a** [4]
- double **filter_b** [4]
- double **gyro_window** [3 ∗26]
- double **mag_window** [3 ∗64]
- double **gyro_delay_buffer** [3 ∗13]
- double **gyro_bias_buffer** [3 ∗128 ∗5]
- double **mag_delay_buffer** [3 ∗128 ∗5]
- int **iSample**
- double **q_current** [4]
- double **q_old** [4]
- double **q_int** [4]
- double **q_int_old** [4]
- double **q_err** [4]

- double **old_qesta** [4]
- double **q_buff** [128 ∗5 ∗4]
- int **q_buff_ind**
- int **n_samples_delay**

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

- apdm_internal.h

6.26 apdm_orientation_info_ukf_t Struct Reference

Public Attributes

- apdm_orientation_ukf_fdata_t **fdata**
- apdm_orientation_ukf_hdata_t **hdata**
- apdm_ukf_state_t **ukf_state**

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

- apdm_internal.h

6.27 apdm_orientation_ukf_fdata_t Struct Reference

Public Attributes

- double **fs**
- double **gyro** [3]

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

- apdm_internal.h

6.28 apdm_orientation_ukf_hdata_t Struct Reference

Public Attributes

- int **iDelayBuffer**
- int **iSample**
- double **old_gyro** [ORIENTATION_BUFFER_LENGTH ∗3]

- double **old_acc** [ORIENTATION_BUFFER_LENGTH ∗3]
- double **old_mag** [ORIENTATION_BUFFER_LENGTH ∗3]
- apdm_orientation_model_t **model**
- double **gyro_bias** [3]
- double **gyro_scale** [3]
- double **acc_bias** [3]
- double **mag_mag**
- double **acc_mag**
- double **acc_var**
- double **gyro_var**
- double **mag_var**
- double **mag_inclination**
- double **inclination_var**
- double **fs**
- double **acc** [3]
- double **gyro** [3]
- double **mag** [3]
- double **q_current** [4]
- double **q_old** [4]

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

- apdm_internal.h

6.29 apdm_progress_t Struct Reference

Public Attributes

- char **task** [128]
- int **num_tasks**
- int **task_index**
- double percent_complete

6.29.1 Member Data Documentation

6.29.1.1 double apdm_progress_t::percent_complete

0-100 floating point for current task

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

- apdm_types.h

6.30 apdm_record_ll_disk_data_t Struct Reference

Public Attributes

- int32_t **idx**
- int32_t **prev_larger_idx**
- int32_t **next_smaller_idx**
- apdm_record_t **data**

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

- apdm_internal.h

6.31 apdm_record_t Struct Reference

```
#include <apdm_types.h>
```

Public Attributes

- uint64_t **sync_val64**
- uint32_t sync_val32_low
- uint32_t **sync_val32_high**
- uint8_t **nRF_pipe**
- uint8_t num_retrys
- int32_t source_ap_index
- uint16_t accl_x_axis
- uint16_t accl_y_axis
- uint16_t accl_z_axis
- bool accl_full_scale_mode
- bool accl_isPopulated
- uint16_t gyro_x_axis
- uint16_t gyro_y_axis
- uint16_t gyro_z_axis
- bool gyro_isPopulated
- uint16_t mag_x_axis
- uint16_t mag_y_axis
- uint16_t mag_z_axis
- uint16_t mag_common_axis
- bool mag_isPopulated
- uint8_t flag_accel_enabled
- uint8_t **flag_gyro_enabled**

- uint8_t **flag_mag_enabled**
- uint8_t **flag_full_scale_enabled**
- uint8_t **flag_sync_lock**
- uint8_t **flag_sync_reset**
- uint8_t **flag_temp_select**
- uint16_t **flags**
- bool gyro_temperature_sensor_selected
- uint32_t **optional_data**
- uint8_t opt_select
- uint32_t debug_data
- uint16_t **debug_flags**
- double **temperature**
- double **temperature_average**
- double **temperature_diff**
- bool **temperature_isPopulated**
- uint32_t **batt_voltage**
- bool batt_voltage_isPopulated
- uint32_t device_info_serial_number
- uint8_t device_info_wireless_channel_id
- uint8_t device_info_wireless_address
- bool device_info_isPopulated
- uint8_t button_status
- uint32_t **tag_data**
- bool **tag_data_isPopulated**
- double **accl_x_axis_si**
- double accl_y_axis_si
- double accl_z_axis_si
- double gyro_x_axis_si
- double gyro_y_axis_si
- double gyro_z_axis_si
- double mag_x_axis_si
- double mag_y_axis_si
- double mag_z_axis_si
- double **gyro_x_axis_filtered**
- double **gyro_y_axis_filtered**
- double **gyro_z_axis_filtered**
- double **orientation_quaternion0**
- double **orientation_quaternion1**
- double **orientation_quaternion2**
- double **orientation_quaternion3**
- double **temperature_si**
- double temperature_derivative_si
- double battery_level

6.31.1 Detailed Description

APDM Sensor Sample

6.31.2 Member Data Documentation

6.31.2.1 bool apdm_record_t::accl_full_scale_mode

raw ADC readings

6.31.2.2 bool apdm_record_t::accl_isPopulated

True indicates accelerometers are in 6G mode, false indicates 2G mode

Referenced by apdm_convert_h5_to_csv().

6.31.2.3 uint16_t apdm_record_t::accl_x_axis

Index of the AP that the sample came in on.

Referenced by apdm_convert_h5_to_csv(), and apdm_extract_next_sample_set().

6.31.2.4 uint16_t apdm_record_t::accl_y_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.5 double apdm_record_t::accl_y_axis_si

meters per second^2

Referenced by apdm_convert_h5_to_csv().

6.31.2.6 uint16_t apdm_record_t::accl_z_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.7 double apdm_record_t::accl_z_axis_si

meters per second^2

Referenced by apdm_convert_h5_to_csv().

6.31.2.8 bool apdm_record_t::batt_voltage_isPopulated

raw ADC readings

6.31.2.9 double apdm_record_t::battery_level

degrees C per sec

6.31.2.10 uint8_t apdm_record_t::button_status

Referenced by apdm_convert_h5_to_csv().

6.31.2.11 uint32_t apdm_record_t::debug_data

Indicates what type of data is in optional_data, see enum apdm_raw_opt_select_t

6.31.2.12 bool apdm_record_t::device_info_isPopulated

6.31.2.13 uint32_t apdm_record_t::device_info_serial_number

Indicates that the battery voltage data is populated

Referenced by apdm_ctx_extract_data_by_device_id(), apdm_ctx_get_next_access_point_record(), and apdm_extract_next_sample_set().

6.31.2.14 uint8_t apdm_record_t::device_info_wireless_address

6.31.2.15 uint8_t apdm_record_t::device_info_wireless_channel_id

Device ID

6.31.2.16 uint8_t apdm_record_t::flag_accel_enabled

Indicates that the mag data is populated

6.31.2.17 bool apdm_record_t::gyro_isPopulated

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.18 bool apdm_record_t::gyro_temperature_sensor_selected

Flags packed binary structure, used to derive the flag_XXXX field values.

6.31.2.19 uint16_t apdm_record_t::gyro_x_axis

Indicates that the accel data is populated

Referenced by apdm_convert_h5_to_csv().

6.31.2.20 double apdm_record_t::gyro_x_axis_si

meters per second^2

Referenced by apdm_convert_h5_to_csv().

6.31.2.21 uint16_t apdm_record_t::gyro_y_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.22 double apdm_record_t::gyro_y_axis_si

radians per second

Referenced by apdm_convert_h5_to_csv().

6.31.2.23 uint16_t apdm_record_t::gyro_z_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.24 double apdm_record_t::gyro_z_axis_si

radians per second

Referenced by apdm_convert_h5_to_csv().

6.31.2.25 uint16_t apdm_record_t::mag_common_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.26 bool apdm_record_t::mag_isPopulated

only used with device protocol version 0. raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.27 uint16_t apdm_record_t::mag_x_axis

Indicates that the gyro data is populated

Referenced by apdm_convert_h5_to_csv().

6.31.2.28 double apdm_record_t::mag_x_axis_si

radians per second a.u.

Referenced by apdm_convert_h5_to_csv().

6.31.2.29 uint16_t apdm_record_t::mag_y_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.30 double apdm_record_t::mag_y_axis_si

a.u.

Referenced by apdm_convert_h5_to_csv().

6.31.2.31 uint16_t apdm_record_t::mag_z_axis

raw ADC readings

Referenced by apdm_convert_h5_to_csv().

6.31.2.32 double apdm_record_t::mag_z_axis_si

a.u.

Referenced by apdm_convert_h5_to_csv().

6.31.2.33 uint8_t apdm_record_t::num_retrys

Internal use only

Referenced by apdm_ctx_get_next_access_point_record().

6.31.2.34 uint8_t apdm_record_t::opt_select

Optional and varying data from the monitor

Referenced by apdm_ctx_get_next_access_point_record().

6.31.2.35 int32_t apdm_record_t::source_ap_index

Number of retry before this sample was received by the AP.

Referenced by apdm_ctx_get_next_access_point_record(), and apdm_extract_next_sample_set().

6.31.2.36 uint32_t apdm_record_t::sync_val32_low

Full 64 bit sync value

Referenced by apdm_ctx_get_next_access_point_record(), and apdm_extract_next_sample_set().

6.31.2.37 double apdm_record_t::temperature_derivative_si

degrees celcius

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

- apdm_types.h

6.32 apdm_recording_info_t Struct Reference

Public Attributes

- apdm_device_info_t **device_info**
- uint64_t **start_sync_count**
- uint64_t **end_sync_count**
- int **num_samples**

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

- apdm_types.h

6.33 apdm_sensor_cmd Struct Reference

Public Member Functions

- **__attribute__** ((__packed__)) union
- payload **__attribute__** ((__packed__))

Public Attributes

- uint8_t **cmd_number**
- uint16_t **payload_size**
- uint16_t **crc16**

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

- apdm_internal.h

6.34 apdm_sensor_compensation_t Struct Reference

Public Attributes

- uint32_t **converted_calibration_version**
- uint32_t **raw_calibration_version**
- union {
  calibration_v4_t **v4**
  calibration_v5_t **v5**
  calibration_v6_t **v6**
  } **data**

6.34.1 Detailed Description

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

- apdm_types.h

6.35 apdm_sensor_device_handle_t Struct Reference

`#include <apdm_internal.h>`

Public Attributes

- bool **is_opal_attached**
- struct libusb_device_handle * **devh**
- apdm_bulk_in_buffer_t **ep_ds_opal_in_buffer**
- apdm_bulk_in_buffer_t **ep_ds_in_buffer**
- uint8_t **usb_protocol_version**
- uint64_t **usb_protocol_subversion**
- uint32_t **dock_id**
- apdm_device_info_t **device_info**
- apdm_device_status_t **offset_test_results**

6.35.1 Detailed Description

This structure maps to the client programmer data type of: typedef void* apdm_device_handle_t;

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

- apdm_internal.h

6.36 apdm_sensor_response Struct Reference

Public Attributes

- uint8_t **response_number**
- uint16_t **payload_size**

- union {
  uint8_t **in_uint8_t**
  uint16_t **in_uint16_t**
  uint32_t **in_uint32_t**
  uint64_t **in_uint64_t**
  uint8_t **ping_mode**
  uint8_t **peek_value**
  uint16_t **peek2_value**
  uint16_t **memory_crc16**
  uint16_t **status_register**
  uint32_t **bootloader_version**
  uint8_t **memory_dump** [UINT16_MAX]
  uint32_t **device_id**
  uint8_t **binary_blob** [2048]
  char **version_string_1** [1024]
  char **version_string_2** [1024]
  char **version_string_3** [1024]
  char **label_0** [DEVICE_LABEL_SIZE]
  char **label_1** [DEVICE_LABEL_SIZE]
  char **label_2** [DEVICE_LABEL_SIZE]
  char **label_3** [DEVICE_LABEL_SIZE]
  uint8_t **mode**
  uint8_t **dock_status**
  uint8_t **battery_charge_status**
  uint16_t **battery_voltage**
  uint64_t **sync_value**
  uint8_t **off_reason**
  uint32_t **uptime_get_value**
  uint32_t **last_uptime_value**
  uint32_t **last_standby_uptime_value**
  uint32_t **config_get_value**
  uint8_t **config_status**
  uint32_t **error_count**
  char **error_name** [1024]
  uint16_t **error_log_size**
  uint16_t **error_log_get_error_id**
  uint16_t **error_stats_size**
  uint16_t **error_stats_get_count**
  uint16_t **stats_size**
  uint16_t **stats_max_value**
  uint16_t **stats_min_value**
  uint16_t **stats_count_value**
  uint32_t **stats_sum_value**
  uint32_t **flash_block_get_value**
  struct {
    uint16_t **year**

uint8_t **month**
uint8_t **day**
uint8_t **hour**
uint8_t **min**
uint8_t **sec**
} **time_get**
uint32_t **calibration_version**
uint32_t **debug_get_value**
uint32_t **protocol_version**
apdm_calibration_data_t **calibration_data**
uint32_t **hw_id**
char **case_id** [16]
uint8_t **sample_get** [2048]
} **response_data**

- uint16_t **crc16**

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

- apdm_internal.h

6.37 apdm_streaming_config_t Struct Reference

Public Attributes

- uint8_t **wireless_channel_number**
- bool enable_sd_card
- bool erase_sd_card
- bool accel_full_scale_mode
- bool enable_accel
- bool enable_gyro
- bool enable_mag
- bool apply_new_sensor_modes
- bool set_configuration_on_device
- apdm_monitor_decimation_rate_t decimation_rate
- apdm_monitor_output_select_rate_t **output_select_rate**
- bool button_enable
- apdm_device_info_t device_info_cache [APDM_MAX_NUMBER_OF_SENSORS]

6.37.1 Member Data Documentation

6.37.1.1 bool apdm_streaming_config_t::accel_full_scale_mode

Boolean flag indicating that the data on the SD card should be erased as part of the initialization process.

Referenced by apdm_configure_all_attached_sensors(), and apdm_init_streaming_config().

6.37.1.2 bool apdm_streaming_config_t::apply_new_sensor_modes

Enable the magnitometers

Referenced by apdm_init_streaming_config().

6.37.1.3 bool apdm_streaming_config_t::button_enable

select output rate

Referenced by apdm_init_streaming_config().

6.37.1.4 apdm_monitor_decimation_rate_t apdm_streaming_config_t::decimation_rate

Allows you to disable the setting of the configuration on the device so that it can be done later in a threaded/concurent manor by the application.

Referenced by apdm_init_streaming_config().

6.37.1.5 apdm_device_info_t apdm_streaming_config_t::device_info_cache[APDM_MAX_NUMBER_OF_SENSORS]

Enable monitor button accessory

Referenced by apdm_streaming_config_get_device_info().

6.37.1.6 bool apdm_streaming_config_t::enable_accel

If true, then accelerometers will be in 6G mode, if false, then they will be in 2G mode

Referenced by apdm_configure_all_attached_sensors(), and apdm_init_streaming_config().

6.37.1.7 bool apdm_streaming_config_t::enable_gyro

Enable the accelerometers

Referenced by apdm_configure_all_attached_sensors(), and apdm_init_streaming_config().

6.37.1.8 bool apdm_streaming_config_t::enable_mag

Enable the gyros

Referenced by apdm_configure_all_attached_sensors(), and apdm_init_streaming_config().

6.37.1.9 bool apdm_streaming_config_t::enable_sd_card

The base wireless channel number to use

Referenced by apdm_configure_all_attached_sensors(), and apdm_init_streaming_config().

6.37.1.10 bool apdm_streaming_config_t::erase_sd_card

Boolean indicating weather or not data should be logged to the SD card on the device.

Referenced by apdm_configure_all_attached_sensors(), and apdm_init_streaming_config().

6.37.1.11 bool apdm_streaming_config_t::set_configuration_on_device

If set to true, flags are carried thru to the device_info structure

Referenced by apdm_init_streaming_config().

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

- apdm_types.h

6.38 apdm_ukf_state_t Struct Reference

Public Attributes

- int **nStates**
- int **nMeasurements**

- double * **x**
- double * **z**
- double * **zf**
- double * **Q**
- double * **R**
- double * **P**
- int(* **f**)(double *, double *, void *)
- int(* **h**)(double *, double *, void *)

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

- kalman_filter.h

6.39 apdm_usb_device_list_t Struct Reference

Public Attributes

- libusb_device ** **deviceListPtr**
- int **num_elements**
- apdm_usb_sorted_device_element_t * **sorted_element_list**

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

- apdm_usb.h

6.40 apdm_usb_sorted_device_element_t Struct Reference

Public Attributes

- uint16_t **vid**
- uint16_t **pid**
- uint32_t **bus_number**
- uint32_t **device_address**
- uint16_t **bcd_device**
- int **iSerialNumber**
- int **libusb_index_number**

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

- apdm_usb.h

6.41 calibration_v4_t Struct Reference

Public Attributes

- double **accl_x_bias**
- double accl_y_bias
- double accl_z_bias
- double accl_x_bias_temp
- double accl_y_bias_temp
- double accl_z_bias_temp
- double accl_z_bias_dtemp
- double accl_x_scale
- double accl_y_scale
- double accl_z_scale
- double accl_x_scale_temp
- double accl_y_scale_temp
- double accl_z_scale_temp
- double accl_xy_sensitivity
- double accl_xz_sensitivity
- double accl_yz_sensitivity
- double accl_error_matrix [3 ∗3]
- double **gyro_x_bias**
- double gyro_y_bias
- double gyro_z_bias
- double gyro_x_bias_temp
- double gyro_x_bias_temp2
- double gyro_y_bias_temp
- double gyro_y_bias_temp2
- double gyro_z_bias_temp [61]
- double gyro_x_scale
- double gyro_y_scale
- double gyro_z_scale
- double gyro_x_scale_temp
- double gyro_y_scale_temp
- double gyro_z_scale_temp
- double gyro_xy_sensitivity
- double gyro_xz_sensitivity
- double gyro_yz_sensitivity
- double gyro_accl_roll
- double gyro_accl_pitch
- double gyro_accl_yaw
- double gyro_error_matrix [3 ∗3]

- double **mag_x_bias**
- double mag_y_bias
- double mag_z_bias
- double mag_x_scale
- double **mag_y_scale**
- double **mag_z_scale**
- double **mag_xy_sensitivity**
- double **mag_xz_sensitivity**
- double **mag_yz_sensitivity**
- double **mag_accl_roll**
- double **mag_accl_pitch**
- double **mag_accl_yaw**
- double **mag_error_matrix** [3 ∗3]
- apdm_mag_dechop_state_t **mag_x_state**
- apdm_mag_dechop_state_t mag_y_state
- apdm_mag_dechop_state_t mag_z_state
- double temperature_bias
- double temperature_scale
- double temperature_bias_msp
- double temperature_scale_msp

6.41.1 Member Data Documentation

6.41.1.1 double calibration_v4_t::accl_error_matrix[3 ∗3]

this will be calculated by the host libraries

6.41.1.2 double calibration_v4_t::accl_x_bias_temp

6.41.1.3 double calibration_v4_t::accl_x_scale

6.41.1.4 double calibration_v4_t::accl_x_scale_temp

6.41.1.5 double calibration_v4_t::accl_xy_sensitivity

6.41.1.6 double calibration_v4_t::accl_xz_sensitivity

6.41.1.7 double calibration_v4_t::accl_y_bias

6.41.1.8 double calibration_v4_t::accl_y_bias_temp

6.41.1.9 double calibration_v4_t::accl_y_scale

6.41.1.10 double calibration_v4_t::accl_y_scale_temp

6.41.1.11 double calibration_v4_t::accl_yz_sensitivity

6.41.1.12 double calibration_v4_t::accl_z_bias

6.41.1.13 double calibration_v4_t::accl_z_bias_dtemp

6.41.1.14 double calibration_v4_t::accl_z_bias_temp

6.41.1.15 double calibration_v4_t::accl_z_scale

6.41.1.16 double calibration_v4_t::accl_z_scale_temp

6.41.1.17 double calibration_v4_t::gyro_accl_pitch

6.41.1.18 double calibration_v4_t::gyro_accl_roll

6.41.1.19 double calibration_v4_t::gyro_accl_yaw

6.41.1.20 double calibration_v4_t::gyro_error_matrix[3 ∗3]

this will be calculated by the host libraries

6.41.1.21 double calibration_v4_t::gyro_x_bias_temp

6.41.1.22 double calibration_v4_t::gyro_x_bias_temp2

6.41.1.23 double calibration_v4_t::gyro_x_scale

6.41.1.24 double calibration_v4_t::gyro_x_scale_temp

6.41.1.25 double calibration_v4_t::gyro_xy_sensitivity

6.41.1.26 double calibration_v4_t::gyro_xz_sensitivity

6.41.1.27 double calibration_v4_t::gyro_y_bias

6.41.1.28 double calibration_v4_t::gyro_y_bias_temp

6.41.1.29 double calibration_v4_t::gyro_y_bias_temp2

6.41.1.30 double calibration_v4_t::gyro_y_scale

6.41.1.31 double calibration_v4_t::gyro_y_scale_temp

6.41.1.32 double calibration_v4_t::gyro_yz_sensitivity

6.41.1.33 double calibration_v4_t::gyro_z_bias

6.41.1.34 double calibration_v4_t::gyro_z_bias_temp[61]

6.41.1.35 double calibration_v4_t::gyro_z_scale

6.41.1.36 double calibration_v4_t::gyro_z_scale_temp

6.41.1.37 double calibration_v4_t::mag_x_scale

6.41.1.38 double calibration_v4_t::mag_y_bias

6.41.1.39 apdm_mag_dechop_state_t calibration_v4_t::mag_y_state

6.41.1.40 double calibration_v4_t::mag_z_bias

6.41.1.41 apdm_mag_dechop_state_t calibration_v4_t::mag_z_state

6.41.1.42 double calibration_v4_t::temperature_bias

6.41.1.43 double calibration_v4_t::temperature_bias_msp

6.41.1.44 double calibration_v4_t::temperature_scale

6.41.1.45 double calibration_v4_t::temperature_scale_msp

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

- apdm_types.h

6.42 calibration_v5_t Struct Reference

Public Attributes

- uint16_t **accl_x_bias** [61]
- uint16_t accl_y_bias [61]
- uint16_t accl_z_bias [61]
- double accl_z_bias_dtemp
- double accl_x_scale

- double accl_y_scale
- double accl_z_scale
- double accl_x_scale_temp
- double accl_y_scale_temp
- double accl_z_scale_temp
- double accl_xy_sensitivity
- double accl_xz_sensitivity
- double accl_yz_sensitivity
- double accl_error_matrix [3 ∗3]
- uint16_t **gyro_x_bias** [61]
- uint16_t gyro_y_bias [61]
- uint16_t gyro_z_bias [61]
- double gyro_x_scale
- double gyro_y_scale
- double gyro_z_scale
- double gyro_x_scale_temp
- double gyro_y_scale_temp
- double gyro_z_scale_temp
- double gyro_xy_sensitivity
- double gyro_xz_sensitivity
- double gyro_yz_sensitivity
- double gyro_accl_roll
- double gyro_accl_pitch
- double gyro_accl_yaw
- double gyro_error_matrix [3 ∗3]
- uint16_t **mag_x_bias** [61]
- uint16_t mag_y_bias [61]
- uint16_t mag_z_bias [61]
- double mag_x_scale
- double **mag_y_scale**
- double **mag_z_scale**
- double **mag_x_scale_temp**
- double **mag_y_scale_temp**
- double **mag_z_scale_temp**
- double **mag_xy_sensitivity**
- double **mag_xz_sensitivity**
- double **mag_yz_sensitivity**
- double **mag_accl_roll**
- double **mag_accl_pitch**
- double **mag_accl_yaw**
- double mag_x_offset
- double mag_y_offset
- double mag_z_offset

- double **mag_conversion_gain**
- double **mag_error_matrix** [3 ∗3]
- apdm_mag_dechop_state_t **mag_x_state**
- apdm_mag_dechop_state_t mag_y_state
- apdm_mag_dechop_state_t mag_z_state
- double temperature_bias
- double temperature_scale
- double temperature_bias_msp
- double temperature_scale_msp

6.42.1 Member Data Documentation

6.42.1.1 double calibration_v5_t::accl_error_matrix[3 ∗3]

this will be calculated by the host libraries

6.42.1.2 double calibration_v5_t::accl_x_scale

6.42.1.3 double calibration_v5_t::accl_x_scale_temp

6.42.1.4 double calibration_v5_t::accl_xy_sensitivity

6.42.1.5 double calibration_v5_t::accl_xz_sensitivity

6.42.1.6 uint16_t calibration_v5_t::accl_y_bias[61]

Temperature dependent bias covering the range [-10,50] C

6.42.1.7 double calibration_v5_t::accl_y_scale

6.42.1.8 double calibration_v5_t::accl_y_scale_temp

6.42.1.9 double calibration_v5_t::accl_yz_sensitivity

6.42.1.10 uint16_t calibration_v5_t::accl_z_bias[61]

Temperature dependent bias covering the range [-10,50] C

6.42.1.11 double calibration_v5_t::accl_z_bias_dtemp

Temperature dependent bias covering the range [-10,50] C

6.42.1.12 double calibration_v5_t::accl_z_scale

6.42.1.13 double calibration_v5_t::accl_z_scale_temp

6.42.1.14 double calibration_v5_t::gyro_accl_pitch

6.42.1.15 double calibration_v5_t::gyro_accl_roll

6.42.1.16 double calibration_v5_t::gyro_accl_yaw

6.42.1.17 double calibration_v5_t::gyro_error_matrix[3 ∗3]

this will be calculated by the host libraries

6.42.1.18 double calibration_v5_t::gyro_x_scale

6.42.1.19 double calibration_v5_t::gyro_x_scale_temp

6.42.1.20 double calibration_v5_t::gyro_xy_sensitivity

6.42.1.21 double calibration_v5_t::gyro_xz_sensitivity

6.42.1.22 uint16_t calibration_v5_t::gyro_y_bias[61]

6.42.1.23 double calibration_v5_t::gyro_y_scale

6.42.1.24 double calibration_v5_t::gyro_y_scale_temp

6.42.1.25 double calibration_v5_t::gyro_yz_sensitivity

6.42.1.26 uint16_t calibration_v5_t::gyro_z_bias[61]

6.42.1.27 double calibration_v5_t::gyro_z_scale

6.42.1.28 double calibration_v5_t::gyro_z_scale_temp

6.42.1.29 double calibration_v5_t::mag_x_offset

Used internally for removing set/reset pulse artifacts

6.42.1.30 double calibration_v5_t::mag_x_scale

6.42.1.31 uint16_t calibration_v5_t::mag_y_bias[61]

6.42.1.32 double calibration_v5_t::mag_y_offset

Used internally for removing set/reset pulse artifacts

6.42.1.33 apdm_mag_dechop_state_t calibration_v5_t::mag_y_state

6.42.1.34 uint16_t calibration_v5_t::mag_z_bias[61]

6.42.1.35 double calibration_v5_t::mag_z_offset

Used internally for removing set/reset pulse artifacts

6.42.1.36 apdm_mag_dechop_state_t calibration_v5_t::mag_z_state

6.42.1.37 double calibration_v5_t::temperature_bias

6.42.1.38 double calibration_v5_t::temperature_bias_msp

6.42.1.39 double calibration_v5_t::temperature_scale

6.42.1.40 double calibration_v5_t::temperature_scale_msp

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

- apdm_types.h

6.43 calibration_v6_t Struct Reference

Public Attributes

- uint16_t **accl_x_bias** [61]
- uint16_t accl_y_bias [61]
- uint16_t accl_z_bias [61]
- double accl_z_bias_dtemp
- double accl_x_scale
- double accl_y_scale
- double accl_z_scale
- double accl_x_scale_temp
- double accl_y_scale_temp

- double accl_z_scale_temp
- double accl_xy_sensitivity
- double accl_xz_sensitivity
- double accl_yz_sensitivity
- double accl_error_matrix [3 ∗3]
- uint16_t **gyro_x_bias** [61]
- uint16_t gyro_y_bias [61]
- uint16_t gyro_z_bias [61]
- double gyro_x_scale
- double gyro_y_scale
- double gyro_z_scale
- double gyro_x_scale_temp
- double gyro_y_scale_temp
- double gyro_z_scale_temp
- double gyro_xy_sensitivity
- double gyro_xz_sensitivity
- double gyro_yz_sensitivity
- double gyro_accl_roll
- double gyro_accl_pitch
- double gyro_accl_yaw
- double gyro_error_matrix [3 ∗3]
- uint16_t **mag_x_bias**
- uint16_t mag_y_bias
- uint16_t mag_z_bias
- double mag_x_scale
- double **mag_y_scale**
- double **mag_z_scale**
- double **mag_xy_sensitivity**
- double **mag_xz_sensitivity**
- double **mag_yz_sensitivity**
- double **mag_accl_x**
- double **mag_accl_y**
- double **mag_accl_z**
- double mag_x_offset
- double mag_y_offset
- double mag_z_offset
- double mag_conversion_gain
- double mag_inclination
- double **mag_error_matrix** [3 ∗3]
- apdm_mag_dechop_state_t **mag_x_state**
- apdm_mag_dechop_state_t mag_y_state
- apdm_mag_dechop_state_t mag_z_state
- double temperature_bias

- double temperature_scale
- double temperature_bias_msp
- double temperature_scale_msp

6.43.1 Member Data Documentation

6.43.1.1 double calibration_v6_t::accl_error_matrix[3 ∗3]

this will be calculated by the host libraries

6.43.1.2 double calibration_v6_t::accl_x_scale

6.43.1.3 double calibration_v6_t::accl_x_scale_temp

6.43.1.4 double calibration_v6_t::accl_xy_sensitivity

6.43.1.5 double calibration_v6_t::accl_xz_sensitivity

6.43.1.6 uint16_t calibration_v6_t::accl_y_bias[61]

Temperature dependent bias covering the range [-10,50] C

6.43.1.7 double calibration_v6_t::accl_y_scale

6.43.1.8 double calibration_v6_t::accl_y_scale_temp

6.43.1.9 double calibration_v6_t::accl_yz_sensitivity

6.43.1.10 uint16_t calibration_v6_t::accl_z_bias[61]

Temperature dependent bias covering the range [-10,50] C

6.43.1.11 double calibration_v6_t::accl_z_bias_dtemp

Temperature dependent bias covering the range [-10,50] C

6.43.1.12 double calibration_v6_t::accl_z_scale

6.43.1.13 double calibration_v6_t::accl_z_scale_temp

6.43.1.14 double calibration_v6_t::gyro_accl_pitch

6.43.1.15 double calibration_v6_t::gyro_accl_roll

6.43.1.16 double calibration_v6_t::gyro_accl_yaw

6.43.1.17 double calibration_v6_t::gyro_error_matrix[3 ∗3]

this will be calculated by the host libraries

6.43.1.18 double calibration_v6_t::gyro_x_scale

6.43.1.19 double calibration_v6_t::gyro_x_scale_temp

6.43.1.20 double calibration_v6_t::gyro_xy_sensitivity

6.43.1.21 double calibration_v6_t::gyro_xz_sensitivity

6.43.1.22 uint16_t calibration_v6_t::gyro_y_bias[61]

Referenced by apdm_recalibrate_gyroscopes_from_h5().

6.43.1.23 double calibration_v6_t::gyro_y_scale

6.43.1.24 double calibration_v6_t::gyro_y_scale_temp

6.43.1.25 double calibration_v6_t::gyro_yz_sensitivity

6.43.1.26 uint16_t calibration_v6_t::gyro_z_bias[61]

Referenced by apdm_recalibrate_gyroscopes_from_h5().

6.43.1.27 double calibration_v6_t::gyro_z_scale

6.43.1.28 double calibration_v6_t::gyro_z_scale_temp

6.43.1.29 double calibration_v6_t::mag_conversion_gain

Calibrated to magnitude 1, this is to convert to uT and should be equal to the field strength during calibration

6.43.1.30 double calibration_v6_t::mag_inclination

Inclination angle in rad. pi/2 - ang(mag, -gravity)

6.43.1.31 double calibration_v6_t::mag_x_offset

Used internally for removing set/reset pulse artifacts

6.43.1.32 double calibration_v6_t::mag_x_scale

6.43.1.33 uint16_t calibration_v6_t::mag_y_bias

6.43.1.34 double calibration_v6_t::mag_y_offset

Used internally for removing set/reset pulse artifacts

6.43.1.35 apdm_mag_dechop_state_t calibration_v6_t::mag_y_state

6.43.1.36 uint16_t calibration_v6_t::mag_z_bias

6.43.1.37 double calibration_v6_t::mag_z_offset

Used internally for removing set/reset pulse artifacts

6.43.1.38 apdm_mag_dechop_state_t calibration_v6_t::mag_z_state

6.43.1.39 double calibration_v6_t::temperature_bias

6.43.1.40 double calibration_v6_t::temperature_bias_msp

6.43.1.41 double calibration_v6_t::temperature_scale

6.43.1.42 double calibration_v6_t::temperature_scale_msp

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

- apdm_types.h

6.44 per_device_info_t Struct Reference

Public Attributes

- apdm_sensor_device_handle_t **sensor_handle**
- apdm_sensor_compensation_t **compensation_data**
- uint32_t **device_last_sync_val_received**
- uint32_t **total_samples_received**
- apdm_device_state_data_t **discovered_device_id_data**
- uint32_t **user_meta_data_uint32_list**
- uint8_t **sensor_has_sync_lock**
- char **user_meta_data_strings** [USER_META_DATA_STRING_SIZE]
- apdm_monitor_error_stat_t **sensor_error_counts** [APDM_MAX_SENSOR_ERROR_COUNTERS]
- apdm_disk_ll_t * **sensor_sample_list_ptr**

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

- apdm_internal.h

6.45 tekhex_t Struct Reference

Public Attributes

- unsigned char * **data**
- unsigned char * **data_default**
- unsigned int **data_size**

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

- tekhex.h

6.46 WIRELESS_PACKET Union Reference

Public Attributes

- wp_raw_t **raw**
- wp_sync_t **sync**
- wp_data_t **data**
- wp_event_t **event**
- WP_CONFIG **config**
- WP_CONFIG_ACK **config_ack**

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

- apdm_l1_ap.h

6.47 WP_CONFIG Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **cmd**
- uint32_t **device_id**
- uint8_t **args** [26]

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

- apdm_l1_ap.h

6.48 WP_CONFIG_ACK Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **error**
- uint32_t **device_id**
- uint8_t **args** [26]

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

- apdm_l1_ap.h

6.49 WP_DATA Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **retrys**
- uint16_t **flags**
- uint16_t **mx**
- uint16_t **my**

- uint16_t **mz**
- uint16_t **mc**
- uint16_t **gx**
- uint16_t **gy**
- uint16_t **gz**
- uint16_t **ax**
- uint16_t **ay**
- uint16_t **az**
- uint32_t **opt_data**
- uint32_t **sync_val**

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

- apdm_l1_ap.h

6.50 WP_EVENT Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **retrys**
- uint16_t **event_id**
- uint32_t **data_a**
- uint32_t **data_b**
- uint32_t **data_c**
- uint32_t **data_d**
- uint32_t **data_e**
- uint32_t **data_f**

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

- apdm_l1_ap.h

6.51 WP_RAW Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **data** [31]

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

- apdm_l1_ap.h

6.52 WP_SYNC Struct Reference

Public Attributes

- uint8_t **type**
- uint8_t **id**
- uint32_t **sync_time** [2]
- uint16_t **requested_device_state**
- uint16_t **max_latency**

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

- apdm_l1_ap.h