Host Libraries API Documentation

Generated by Doxygen 1.6.3

Fri Jan 7 10:28:24 2011

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This has been replaced by adpm_ap_set_max_latency_value_seconds().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

Member apdm_autoconfigure_devices_and_accesspoint(apdm_ctx_t context, const uint8_t wire
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 wi
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 wi
use apdm_autoconfigure_devices_and_accesspoint4(), will be removed after March 2011

Member apdm_sensor_list_attached_sensors(uint32_t *serial_number_buffer, const uint32_t bu
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 APDM

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, apdm_progress_t *progress)
- 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_csv (apdm_csv_t file, apdm_device_info_t *info, apdm_record_t *records, int sampleNumber, int nDevices, bool store_raw, bool store_si)
- 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 strideLength)
- APDM_EXPORT int apdm_get_hdf_dataset_shape (char *file, char *monitor_id, char *datasetName, int *shape, int *ndims)
- 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)
- APDM_EXPORT void apdm_usleep (const uint64_t microseconds)
- APDM_EXPORT void apdm_msleep (const uint64_t milliseconds)

5.1.1 Function Documentation

5.1.1.1 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

5.1.1.2 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

Referenced by apdm_process_raw().

5.1.1.3 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

5.1.1.4 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.

nMonitors The number of monitors in the device_info array.

Returns

hid_t HDF5 file handle, always > 0 on success.

Referenced by apdm_process_raw().

5.1.1.5 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

5.1.1.6 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.

device_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

Referenced by apdm_get_hdf_device_list_swig().

5.1.1.7 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.

device_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

References apdm_get_hdf_device_list().

5.1.1.8 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

Referenced by apdm_get_hdf_label_list_swig().

5.1.1.9 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

References apdm_get_hdf_label_list().

5.1.1.10 APDM_EXPORT void apdm_msleep (const uint64_t *milliseconds*)

Platform independent version of msleep().

Parameters

milliseconds Number of milliseconds to sleep for

5.1.1.11 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*, 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.

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 otherwise

References apdm_close_file_hdf(), apdm_create_file_hdf(), apdm_log_debug(), and apdm_read_raw_file_info().

5.1.1.12 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

References apdm_log_debug().

5.1.1.13 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 "Sync-Value"

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

References apdm_log_debug().

5.1.1.14 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

file_info A pointer to the file_info structure to populate with metadata.

Returns

APDM_OK on success, error code otherwise

References apdm_log_debug().

Referenced by apdm_process_raw().

5.1.1.15 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().

5.1.1.16 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

Referenced by apdm_write_record_hdf().

5.1.1.17 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*)

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)

Returns

APDM_OK on success

5.1.1.18 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.

sampleNumber 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

References 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_log_debug(), apdm_write_annotation(), 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_x_axis, apdm_record_t::mag_x_axis_si, apdm_record_t::mag_y_axis, apdm_record_t::mag_z_axis, and apdm_record_t::temperature_derivative_si.

5.2 Setup

Functions

- APDM_EXPORT int apdm_ap_get_num_access_points_on_host1 (uint32_t *dest)
- APDM_EXPORT int apdm_ap_init_handle (apdm_ap_handle_t ap_handle)
- 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 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 deviceIndex)
- APDM_EXPORT int apdm_sensor_list_attached_sensors3 (uint32_t *serial_number_buffer, const uint32_t buffer_length, uint32_t *dest_count)
- APDM_DEPRECATED APDM_EXPORT int apdm_sensor_list_attached_sensors (uint32_t *serial_number_buffer, const uint32_t buffer_length)
- APDM_EXPORT int apdm_sensor_get_num_attached_dockingstations1 (uint32_t *dest_num_docks)
- 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_calibration_override_minimum_supported_version (const uint32_t new_version)
- 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_DEPRECATED APDM_EXPORT int adpm_ap_set_max_latency_value (apdm_ap_handle_t ap_handle, const uint32_t max_latency_ms)
- 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 adpm_ap_set_max_latency_value_seconds (apdm_ap_handle_t ap_handle, const uint16_t max_latency_seconds)
- APDM_EXPORT int apdm_send_accesspoint_cmd (apdm_ap_handle_t ap_handle, const char *cmdToSend, char *BYTE, const uint32_t outputBufferLength, const uint32_t numLinesToRead, const uint32_t timeoutMilliseconds)
- APDM_EXPORT int **apdm_init_access_point_wireless** (apdm_ap_handle_t ap_handle, const uint8_t wirelessChannel1, const uint8_t wirelessChannel2, const uint32_t deviceRXAddressHighOrderBytesA, const uint32_t deviceRXAddressHighOrderBytesB, const uint8_t radio1_pipe_count, const uint8_t radio2_pipe_count)
- APDM_EXPORT int apdm_exit (void)
- APDM_EXPORT 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)

5.2.1 Function Documentation

5.2.1.1 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().

5.2.1.2 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_INFINANT_MAX_LATENCY implies infinity.

Returns

APDM_OK on success, error code otherwise.

Referenced by adpm_ap_set_max_latency_value(), and apdm_set_max_sample_delay_seconds().

5.2.1.3 APDM_EXPORT int apdm_ap_get_num_access_points_on_host1 (uint32_t * *dest*)

Parameters

∗***num_devices*** 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_open_all_access_points().

5.2.1.4 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 if successful.

Referenced by apdm_ctx_initialize_context().

5.2.1.5 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 otherwise.

5.2.1.6 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 otherwise.

5.2.1.7 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

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

Returns

APDM_OK on success, error code otherwise

5.2.1.8 APDM_EXPORT 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

References apdm_ds_get_hardware_version(), apdm_ds_get_serial(), apdm_estimate_now_sync_value(), apdm_log_debug(), apdm_log_error(), apdm_log_info(), apdm_log_warning(), apdm_sensor_close(), apdm_-sensor_cmd_config_commit(), apdm_sensor_cmd_error_clear(), apdm_-sensor_cmd_flash_block_set(), apdm_sensor_cmd_ping(), apdm_sensor_-cmd_sync_set(), apdm_sensor_comm_channel_verify_supported_version(), apdm_sensor_open(), apdm_device_info_t::dock_id_during_configuration, apdm_device_info_t::wireless_addr_id, apdm_device_info_t::wireless_block2, apdm_device_info_t::wireless_channel2, and apdm_device_info_t::wireless_-timeslice.

Referenced by apdm_autoconfigure_mesh_sync().

5.2.1.9 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.

Returns

APDM_OK on success, error code otherwise

5.2.1.10 APDM_EXPORT int apdm_send_accesspoint_cmd (apdm_ap_handle_t *ap_handle*, const char * *cmdToSend*, char * *BYTE*, 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

outputStringBuffer The destination into which the response from the AP is to be placeed.

outputBufferLength The length of the outputStringBuffer.

numLinesToRead The number of lines that are expected in response to the command being sent.

timeoutMiliseconds Maximum time length to wait for the response.

Returns

APDM_OK on success.

References apdm_log_error().

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

5.2.1.11 APDM_EXPORT apdm_device_handle_t apdm_sensor_allocate_and_open (const uint32_t *index*)

Allocates and opens/connects to a sensor on the system

Parameters

index The index of the sensor attached to the host

Returns

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

References apdm_sensor_allocate_handle(), apdm_sensor_free_handle(), and apdm_sensor_open().

5.2.1.12 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().

Referenced by apdm_halt_all_attached_sensors(), and apdm_sensor_allocate_and_open().

5.2.1.13 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.

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

5.2.1.14 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, other code on failure.

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

Referenced by apdm_configure_all_attached_sensors_mesh().

5.2.1.16 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(), and apdm_sensor_close_and_free().

5.2.1.17 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 otherwise

5.2.1.18 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().

5.2.1.19 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.

Referenced by apdm_sensor_list_attached_sensors().

5.2.1.20 APDM_EXPORT int apdm_sensor_open (apdm_device_handle_t *device_handle*, const uint32_t *deviceIndex*)

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

Returns

APDM_OK on success.

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

5.2.1.21 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

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

Returns

APDM_OK on success, error code otherwise

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

References apdm_sensor_cmd_calibration_version().

5.3 Context

Functions

- APDM_EXPORT int apdm_ctx_set_correlation_fifo_temp_directory (const char *directory)
- APDM_EXPORT int apdm_ctx_get_expected_number_of_sensors2 (apdm_ctx_t context, uint32_t *dest)
- APDM_EXPORT int apdm_open_all_access_points (apdm_ctx_t context)
- APDM_EXPORT enum APDM_Status apdm_ctx_set_error_handling_mode (apdm_ctx_t context, enum APDMErrorHandlingBehavior newMode)
- APDM_EXPORT int apdm_ctx_get_sensor_compensation_data (apdm_ctx_t context, apdm_sensor_compensation_t *dest_comp_data, const int32_t sensorIndex)
- APDM_EXPORT int apdm_ctx_set_sensor_compensation_data (apdm_ctx_t context, const apdm_sensor_compensation_t *src_comp_data, const int32_t sensorIndex)
- 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 deviceId, const uint32_t value)
- APDM_EXPORT int apdm_ctx_set_metadata_string (apdm_ctx_t context, const uint32_t deviceId, const char *str)
- APDM_EXPORT uint32_t apdm_ctx_get_metadata_uint32 (apdm_ctx_t context, const uint32_t deviceId)
- 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 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 deviceID, apdm_record_t *dest)
- APDM_EXPORT int apdm_ctx_get_next_access_point_record (apdm_ctx_t context, apdm_record_t *data, const int apIndexNumber, const bool allowAPTransferFlag)
- 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_get_next_synchronization_event (apdm_ctx_t context, apdm_external_sync_data_t *dest)
- APDM_EXPORT int apdm_ctx_get_next_record (apdm_ctx_t context, apdm_record_t *dest)
- APDM_EXPORT int32_t apdm_ctx_get_device_id_by_index (apdm_ctx_t context, const uint32_t index)
- APDM_EXPORT int **apdm_ctx_set_requested_device_state** (apdm_ctx_t context, const enum RequestedDeviceState state, const int apIndexNumber)
- 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_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint (apdm_ctx_t context, const uint8_t wirelessChannelNumber)
- APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint2 (apdm_ctx_t context, const uint8_t wirelessChannelNumber, const bool enable_sd_card)
- APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint3 (apdm_ctx_t context, const uint8_t wirelessChannelNumber, const bool enable_sd_card, const bool erase_sd_card)
- APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint4 (apdm_ctx_t context, const uint8_t wirelessChannelNumber, 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_devices_and_accesspoint_-wireless (apdm_ctx_t context, const uint8_t wirelessChannelNumber)
- APDM_EXPORT int apdm_autoconfigure_mesh_sync (apdm_ctx_t context, const uint8_t wirelessChannelNumber, 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_disable_accesspoint_wireless (apdm_-ctx_t context)
- APDM_EXPORT int apdm_ctx_re_enable_accesspoint_wireless (apdm_ctx_t context)
- 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_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_set_max_sample_delay_seconds (apdm_-ctx_t context, const uint16_t max_data_delay_seconds)
- APDM_EXPORT int apdm_get_max_sample_delay_seconds (apdm_-ctx_t context, uint16_t *dest)
- APDM_EXPORT int apdm_reset_num_samples_from_ap (apdm_ctx_-t context)
- APDM_EXPORT int apdm_get_num_samples_from_ap (apdm_ctx_-t context)
- APDM_EXPORT uint32_t apdm_monitor_decimation_rate_t_to_int (const apdm_monitor_decimation_rate_t rate)

5.3.1 Function Documentation

5.3.1.1 APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint (apdm_ctx_t *context*, const uint8_t *wirelessChannelNumber*)

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().

5.3.1.2 APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint2 (apdm_ctx_t *context*, const uint8_t *wirelessChannelNumber*, 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().

5.3.1.3 APDM_DEPRECATED APDM_EXPORT int apdm_autoconfigure_devices_and_accesspoint3 (apdm_ctx_t *context*, const uint8_t *wirelessChannelNumber*, 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().

5.3.1.4 APDM_EXPORT int apdm_autoconfigure_devices_-and_accesspoint4 (apdm_ctx_t *context*, const uint8_t *wirelessChannelNumber*, 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

wirelessChannelNumber 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 magnitometers

Returns

APDM_OK if everything worked, something else if configuration failed.

Referenced by apdm_autoconfigure_devices_and_accesspoint(), apdm_-autoconfigure_devices_and_accesspoint2(), and apdm_autoconfigure_-devices_and_accesspoint3().

5.3.1.5 APDM_EXPORT int apdm_autoconfigure_devices_and_-accesspoint_wireless (apdm_ctx_t *context*, const uint8_t *wirelessChannelNumber*)

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

wirelessChannelNumber The base wireless channel to transmit data on, 0-100.

Returns

APDM_OK if everything worked, something else if configuration failed.

5.3.1.6 APDM_EXPORT int apdm_autoconfigure_mesh_sync (apdm_ctx_t *context*, const uint8_t *wirelessChannelNumber*, 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

wirelessChannelNumber 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

References apdm_configure_all_attached_sensors_mesh(), and apdm_log_-info().

5.3.1.7 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

wirelessChannelNumber 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 magnitometers

Returns

APDM_OK if everything worked, something else if configuration failed.

5.3.1.8 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().

5.3.1.9 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

The device ID

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().

5.3.1.10 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 if everything worked, something else if configuration failed.

5.3.1.11 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 if successful, error code otherwise.

References apdm_ap_disconnect(), and apdm_sensor_close().

5.3.1.12 APDM_EXPORT int apdm_ctx_extract_data_by_device_id (apdm_ctx_t *context*, const uint32_t *deviceID*, apdm_record_t * *dest*)

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

Parameters

context

deviceID The device id for which you want to retrieve data for

∗***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.

5.3.1.13 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().

5.3.1.14 APDM_EXPORT int32_t apdm_ctx_get_device_id_by_index (apdm_ctx_t *context*, const uint32_t *index*)

Parameters

context The context for which you want the device id

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.

5.3.1.15 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 otherwise

5.3.1.16 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.

Returns

-1 on error, otherwise the device index number for the given device ID,

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

5.3.1.17 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

∗***dest*** The destination structure into which you'd like to store the data

Returns

APDM_OK on success, error code otherwise.

References apdm_ctx_get_device_index_by_id3().

5.3.1.18 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_get_sampling_frequency(), and apdm_ctx_get_-wireless_streaming_status().

5.3.1.19 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

5.3.1.20 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

The device ID

Returns

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

5.3.1.21 APDM_EXPORT uint32_t apdm_ctx_get_metadata_uint32 (apdm_ctx_t *context*, const uint32_t *deviceId*)

Allows for the retrieval of metadata for a device id.

Parameters

context The apdm handle

deviceId The deviceId for which you want metadata

Returns

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

5.3.1.22 APDM_EXPORT int apdm_ctx_get_next_access_point_record (apdm_ctx_t *context*, apdm_record_t * *data*, const int *apIndexNumber*, const bool *allowAPTransferFlag*)

Gets the next record from the access point indicated

Parameters

context

∗***data*** Destination into which to place date

apIndexNumber The index number of the AP on the host for which to retrieve data.

allowAPTransferFlag 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_all_ap_debug_info(), apdm_ctx_get_device_-index_by_id3(), apdm_log_debug(), apdm_log_error(), apdm_device_info_-t::decimation_factor, apdm_record_t::device_info_serial_number, apdm_-record_t::num_retrys, apdm_record_t::opt_select, apdm_record_t::source_-ap_index, apdm_record_t::sync_val32_low, and apdm_record_t::temperature.

5.3.1.23 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(), and apdm_log_error().

5.3.1.24 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 emited as soon as it's 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_log_error().

5.3.1.25 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.

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.

5.3.1.26 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

5.3.1.27 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.

5.3.1.28 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.

5.3.1.29 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.

5.3.1.30 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.

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

Returns

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

5.3.1.32 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().

5.3.1.33 APDM_EXPORT int apdm_ctx_get_sensor_compensation_data (apdm_ctx_t *context*, apdm_sensor_compensation_t * *dest_comp_data*, const int32_t *sensorIndex*)

Parameters

context The apdm context

*** *dest_comp_data*** The destination into which to store compensation data for the sensor of index sensorIndex.

sensorIndex The index of the sensor for which you want to retrieve compensatoin data.

Returns

APDM_OK on success

References apdm_log_error().

5.3.1.34 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 initializedm or since the last time apdm_sync_record_head_list() was called.

5.3.1.35 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.

5.3.1.36 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 otherwise.

5.3.1.37 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

The device ID

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_estimate_now_sync_value().

5.3.1.38 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 otherwise

References apdm_ctx_get_device_index_by_id3(), apdm_ctx_get_expected_number_of_sensors2(), and apdm_log_debug().

5.3.1.39 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 if initialization was successful, error code otherwise.

References apdm_ap_init_handle().

Referenced by apdm_open_all_access_points().

5.3.1.40 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_log_error().

5.3.1.41 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 if everything worked, something else if configuration failed.

5.3.1.42 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 otherwise

5.3.1.43 APDM_EXPORT enum APDM_Status apdm_ctx_set_error_handling_mode (apdm_ctx_t *context*, enum APDMErrorHandlingBehavior *newMode*)

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

5.3.1.44 APDM_EXPORT int apdm_ctx_set_metadata_string (apdm_ctx_t *context*, const uint32_t *deviceId*, 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

value The char* type meta data to be stored, maximum length is USER_-META_DATA_STRING_SIZE(64)

Returns

APDM_OK on success

5.3.1.45 APDM_EXPORT int apdm_ctx_set_metadeta_uint32 (apdm_ctx_t *context*, const uint32_t *deviceId*, 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

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

5.3.1.46 APDM_EXPORT int apdm_ctx_set_sensor_compensation_data (apdm_ctx_t *context*, const apdm_sensor_compensation_t * *src_comp_data*, const int32_t *sensorIndex*)

Parameters

context The apdm context

__src_comp_data__ The source of compensation data which to store into the contex for the sensor of index sensorIndex.

sensorIndex The index of the sensor for which you want to retrieve compensatoin data.

Returns

APDM_OK on success

References apdm_log_error().

5.3.1.47 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.

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_get_time_ms_64(), apdm_log_debug(), apdm_log_error(), and apdm_usleep().

5.3.1.48 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 upon success, error code if failure.

5.3.1.49 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 AP to the host libraries since the last time aptm_reset_num_samples_-from_ap was called, negative error code otherwise.

5.3.1.50 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().

5.3.1.51 APDM_EXPORT int apdm_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.

Parameters

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

Returns

APDM_OK on success, error code otherwise.

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_debug(), apdm_log_error(), apdm_log_info(), and apdm_strerror().

5.3.1.52 APDM_EXPORT int apdm_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 otherwise.

5.3.1.53 APDM_EXPORT int apdm_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_INFINANT_MAX_LATENCY for infinity

Returns

APDM_OK upon success, error code if failure.

References adpm_ap_set_max_latency_value_seconds().

5.4 Misc

Functions

- APDM_EXPORT int apdm_device_extract_module_id_from_case_id_string (const char *case_id, uint32_t *dest_module_id)
- 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_calculate_sync_value_age (const uint64_t sync1, const uint64_t sync2)
- APDM_EXPORT uint64_t apdm_get_time_ms_64 (struct timeval *dest)
- 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)
- 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 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 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 uint64_t apdm_estimate_now_sync_value (apdm_ctx_t context)
- uint64_t apdm_get_now_sync_value_host (void)

5.4.1 Function Documentation

5.4.1.1 APDM_EXPORT uint64_t apdm_calculate_sync_value_age (const uint64_t *sync1*, const uint64_t *sync2*)

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().

5.4.1.2 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 moniton monitor

dest_module_id Destination into which to store the module ID

Returns

APDM_OK on success, error code otherwise

5.4.1.3 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

Returns

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

5.4.1.4 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).

5.4.1.5 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.

5.4.1.6 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.

5.4.1.7 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.

5.4.1.8 APDM_EXPORT uint64_t apdm_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_configure_all_attached_sensors_mesh(), apdm_ctx_-get_wireless_reliability_value(), and apdm_extract_next_sample_set().

5.4.1.9 APDM_EXPORT const char* apdm_get_library_build_datetime (void)

Returns

The date on which the libraries were built.

5.4.1.10 APDM_EXPORT const char* apdm_get_library_version (void)

Returns

The version of the host libraries currently being used

5.4.1.11 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_estimate_now_sync_value().

5.4.1.12 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_ctx_sync_record_list_head(), apdm_get_now_sync_value_host(), apdm_open_all_access_points(), and apdm_sensor_populate_device_info().

5.4.1.13 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.

5.4.1.14 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.

5.4.1.15 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().

5.4.1.16 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().

5.4.1.17 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.

5.4.1.18 APDM_EXPORT const char* apdm_strerror (const enum APDM_Status *status_code*)

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.

Referenced by apdm_open_all_access_points().

5.4.1.19 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.

5.5 AccessPoint

Functions

- 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_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 apdm_ap_get_version_string (apdm_ap_handle_t ap_handle, char *BYTE, const int destLength)
- APDM_EXPORT int apdm_ap_get_board_version_string (apdm_ap_handle_t ap_handle, char *BYTE, 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, 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_get_accesspoint_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_EXPORT int apdm_ds_get_protocol_subversion (apdm_device_handle_t device_handle, int64_t *dest_protocol_subversion)

5.5.1 Function Documentation

5.5.1.1 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.

5.5.1.2 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, other on error.

References apdm_ap_get_protocol_subversion(), and apdm_ap_get_version_string().

Referenced by apdm_open_all_access_points().

5.5.1.3 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 if successful

Referenced by apdm_ctx_disconnect(), and apdm_open_all_access_points().

5.5.1.4 APDM_EXPORT int apdm_ap_get_board_version_string (apdm_ap_handle_t *ap_handle*, char * *BYTE*, const int *destLength*)

Returns the board hardware version string from the access point.

Parameters

the_handle The handle with respect to what version string you want.

***dest** 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.

References apdm_send_accesspoint_cmd().

5.5.1.5 APDM_EXPORT int apdm_ap_get_case_id (apdm_ap_handle_t *ap_handle*, char * *BYTE*, const int *dest_buffer_length*)

Retrieves the case ID of the AP.

Parameters

ap_handle The AP handle

BYTE 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 otherwise

References apdm_log_error(), and apdm_log_warning().

Referenced by apdm_open_all_access_points().

5.5.1.6 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

the_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 otherwise

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

5.5.1.7 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 otherwise

Referenced by apdm_ap_get_id(), and apdm_open_all_access_points().

5.5.1.8 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_version*** The destination into which to store the protocol version

Returns

APDM_OK on success, error code otherwise.

Referenced by apdm_ap_connect().

5.5.1.9 APDM_EXPORT int apdm_ap_get_version_string (apdm_ap_handle_t *ap_handle*, char * *BYTE*, const int *destLength*)

Returns the firmware version string from the access point.

Parameters

the_handle The handle with respect to what version string you want.

***dest** 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.

References apdm_send_accesspoint_cmd().

Referenced by apdm_ap_connect(), and apdm_ap_verify_supported_version().

5.5.1.10 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 otherwise

5.5.1.11 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

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

Returns

APDM_OK on success, error code otherwise

5.5.1.12 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.

5.5.1.13 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.

5.5.1.14 APDM_EXPORT int apdm_ap_verify_supported_version (apdm_ap_handle_t *ap_handle*)

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

Parameters

the_handle The access point handle

Returns

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

References apdm_ap_get_version_string(), and apdm_log_error().

5.5.1.15 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

5.5.1.16 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 otherwise.

References apdm_log_error(), and apdm_log_info().

5.5.1.17 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.

Returns

APDM_OK if successful, error code otherwise

Referenced by apdm_ctx_get_next_access_point_record().

5.5.1.18 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_version*** The destination into which to store the protocol version

Returns

APDM_OK on success, error code otherwise.

5.5.1.19 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.

References apdm_log_warning().

5.5.1.20 APDM_EXPORT int apdm_get_accesspoint_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_send_accesspoint_cmd().

5.6 DataHandling

Functions

- int apdm_extract_next_sample_set (apdm_ctx_t context, const bool checked_all_aps)

5.6.1 Function Documentation

5.6.1.1 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 necessaraly 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_estimate_now_sync_value(), apdm_log_debug(), apdm_log_error(), apdm_log_warning(), 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().

5.7 Device

Functions

- 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_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_cmd_halt (apdm_device_handle_t device_handle)
- APDM_EXPORT int apdm_sensor_populate_device_info (apdm_device_handle_t device_handle, apdm_device_info_t *dest)
- 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, 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_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)
- 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 *dest, 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_calibration_data (apdm_device_handle_t device_handle, apdm_sensor_compensation_t *sensor_comp)
- APDM_EXPORT int apdm_sensor_cmd_version_string_1 (apdm_device_handle_t device_handle, char *BYTE, const int dest_buff_length)
- APDM_EXPORT int apdm_sensor_cmd_version_string_2 (apdm_device_handle_t device_handle, char *BYTE, const int dest_buff_length)
- APDM_EXPORT int apdm_sensor_cmd_version_string_3 (apdm_device_handle_t device_handle, char *BYTE, 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])
- APDM_EXPORT int apdm_sensor_config_get_label (apdm_device_handle_t device_handle, char *BYTE, 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, 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_peek_status_register (apdm_device_handle_t device_handle, uint16_t *current_value)
- APDM_EXPORT int apdm_sensor_cmd_bit_clear_status_register (apdm_device_handle_t device_handle, uint16_t mask)
- APDM_EXPORT int apdm_sensor_cmd_bit_set_status_register (apdm_device_handle_t device_handle, uint16_t mask)
- APDM_EXPORT int apdm_sensor_cmd_mode (apdm_device_handle_t device_handle, uint8_t *current_mode)
- APDM_EXPORT int apdm_sensor_cmd_timer_adjust_set (apdm_device_handle_t device_handle, uint16_t value)

5.7.1 Function Documentation

5.7.1.1 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(), and apdm_sensor_open().

5.7.1.2 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 otherwise.

References apdm_monitor_decimation_rate_t_to_int(), apdm_monitor_output_select_rate_t_to_int(), and apdm_device_info_t::decimation_factor.

5.7.1.3 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 configureation to be applied to the device.

Returns

APDM_OK on success, error code otherwise.

5.7.1.4 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 otherwise.

5.7.1.5 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 otherwise.

5.7.1.6 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 otherwise.

5.7.1.7 APDM_EXPORT int apdm_sensor_cmd_bit_clear_status_register (apdm_device_handle_t *device_handle*, uint16_t *mask*)

For APDM internal use only

Parameters

device_handle The device handle.

Returns

APDM_OK on success, error code otherwise.

5.7.1.8 APDM_EXPORT int apdm_sensor_cmd_bit_set_status_register (apdm_device_handle_t *device_handle*, uint16_t *mask*)

For APDM internal use only

Parameters

device_handle The device handle.

Returns

APDM_OK on success, error code otherwise.

5.7.1.9 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 compensation 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 otherwise.

5.7.1.10 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 otherwise.

Referenced by apdm_sensor_populate_device_info().

5.7.1.11 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 otherwise.

Referenced by apdm_sensor_populate_device_info(), and apdm_sensor_verify_supported_calibration_version().

5.7.1.12 APDM_EXPORT int apdm_sensor_cmd_case_id (apdm_device_handle_t *device_handle*, char * *BYTE*, const int *dest_buff_length*)

Retrieves the case ID from the motion monitor

Parameters

device_handle The device handle.

∗***dest*** Destination buffer into which to store the case ID

dest_buff_length size of ∗dest

Returns

APDM_OK on success, error code otherwise.

References apdm_log_debug().

Referenced by apdm_sensor_populate_device_info().

5.7.1.13 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 otherwise. Sets what is pointed to by is_valid to 1 if the configuration is valid, sets it to 0 otherwise.

5.7.1.14 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 otherwise.

Referenced by apdm_configure_all_attached_sensors_mesh().

5.7.1.15 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 otherwise.

Referenced by apdm_sensor_config_get_label(), and apdm_sensor_-populate_device_info().

5.7.1.16 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.

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 otherwise.

Referenced by apdm_sensor_config_set_label(), and apdm_sensor_-configure_wireless().

5.7.1.17 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 otherwise.

5.7.1.18 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 otherwise.

5.7.1.19 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 otherwise.

5.7.1.20 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 otherwise.

Referenced by apdm_sensor_get_device_id_list(), and apdm_sensor_-populate_device_info().

5.7.1.21 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 otherwise.

5.7.1.22 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 otherwise.

5.7.1.23 APDM_EXPORT int apdm_sensor_cmd_enter_bootloader (apdm_device_handle_t *device_handle*)

Parameters

device_handle The device handle.

Returns

APDM_OK on success, error code otherwise.

5.7.1.24 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 otherwise.

Referenced by apdm_configure_all_attached_sensors_mesh().

5.7.1.25 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 otherwise.

5.7.1.26 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 otherwise.

5.7.1.27 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 otherwise.

5.7.1.28 APDM_EXPORT int apdm_sensor_cmd_error_name (apdm_device_handle_t *device_handle*, char * *dest*, const int *length*, uint16_t *error_id*)

Retrieves the name of the specified error ID.

Parameters

device_handle The device handle.

dest Destination string into which to store the name of the error

length The size of the *dest string buffer

error_id The ID for which you want to retrieve the error name

Returns

APDM_OK on success, error code otherwise.

5.7.1.29 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 otherwise.

5.7.1.30 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 otherwise.

5.7.1.31 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 otherwise.

5.7.1.32 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 otherwise.

Referenced by apdm_configure_all_attached_sensors_mesh().

5.7.1.33 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 otherwise.

5.7.1.34 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 otherwise.

Referenced by apdm_halt_all_attached_sensors().

5.7.1.35 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 otherwise.

Referenced by apdm_sensor_populate_device_info().

5.7.1.36 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 otherwise.

5.7.1.37 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 otherwise.

5.7.1.38 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 otherwise.

5.7.1.39 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 otherwise.

5.7.1.40 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 otherwise.

5.7.1.41 APDM_EXPORT int apdm_sensor_cmd_mode (apdm_device_handle_t *device_handle*, uint8_t * *current_mode*)

For APDM internal use only.

Parameters

device_handle The device handle.

*****current_mode*** Destination into which to store the current mode, modes are defined in "enum APDM_Mode" in apdm_types.h

Returns

APDM_OK on success, error code otherwise.

5.7.1.42 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 otherwise.

5.7.1.43 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 otherwise.

5.7.1.44 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 otherwise.

5.7.1.45 APDM_EXPORT int apdm_sensor_cmd_peek_status_register (apdm_device_handle_t *device_handle*, uint16_t ∗ *current_value*)

For APDM internal use only

Parameters

device_handle The device handle.

Returns

APDM_OK on success, error code otherwise.

5.7.1.46 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 otherwise.

Referenced by apdm_configure_all_attached_sensors_mesh().

5.7.1.47 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 otherwise.

5.7.1.48 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 otherwise.

5.7.1.49 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 otherwise.

5.7.1.50 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 otherwise.

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 otherwise.

5.7.1.51 APDM_EXPORT int apdm_sensor_cmd_standby (apdm_device_handle_t *device_handle*)

Parameters

device_handle The device handle.

Returns

APDM_OK on success, error code otherwise.

5.7.1.52 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 otherwise.

5.7.1.53 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 otherwise.

5.7.1.54 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 otherwise.

5.7.1.55 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 otherwise.

5.7.1.56 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 otherwise.

5.7.1.57 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 otherwise.

5.7.1.58 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 otherwise.

5.7.1.59 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 otherwise.

5.7.1.60 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 otherwise.

5.7.1.61 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 otherwise.

Referenced by apdm_configure_all_attached_sensors_mesh().

5.7.1.62 APDM_EXPORT int apdm_sensor_cmd_time_get (apdm_device_handle_t, 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 otherwise.

References apdm_log_debug().

5.7.1.63 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 otherwise.

References apdm_log_debug().

Referenced by apdm_sensor_cmd_time_set2().

5.7.1.64 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 otherwise.

References apdm_sensor_cmd_time_set().

5.7.1.65 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 otherwise.

5.7.1.66 APDM_EXPORT int apdm_sensor_cmd_timer_adjust_set (apdm_device_handle_t *device_handle*, uint16_t *value*)

This is for APDM internal use only

Parameters

device_handle The device handle.

Returns

APDM_OK on success, error code otherwise.

5.7.1.67 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 otherwise.

5.7.1.68 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 otherwise.

5.7.1.69 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 otherwise.

5.7.1.70 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 otherwise.

5.7.1.71 APDM_EXPORT int apdm_sensor_cmd_version_string_1 (apdm_device_handle_t *device_handle*, char * *BYTE*, const int *dest_buff_length*)

Parameters

device_handle The device handle.

∗***BYTE*** Destination into which to store the version string.

dest_buff_length length of the ∗BYTE array.

Returns

APDM_OK on success, error code otherwise.

Referenced by apdm_sensor_populate_device_info().

5.7.1.72 APDM_EXPORT int apdm_sensor_cmd_version_string_2 (apdm_device_handle_t *device_handle*, char ∗ *BYTE*, const int *dest_buff_length*)

Parameters

device_handle The device handle.

∗***BYTE*** Destination into which to store the version string.

dest_buff_length length of the ∗BYTE array.

Returns

APDM_OK on success, error code otherwise.

Referenced by apdm_sensor_populate_device_info().

5.7.1.73 APDM_EXPORT int apdm_sensor_cmd_version_string_3 (apdm_device_handle_t *device_handle*, char ∗ *BYTE*, const int *dest_buff_length*)

Parameters

device_handle The device handle.

∗***BYTE*** Destination into which to store the version string.

dest_buff_length length of the ∗BYTE array.

Returns

APDM_OK on success, error code otherwise.

Referenced by apdm_sensor_populate_device_info().

5.7.1.74 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 otherwise.

5.7.1.75 APDM_EXPORT int apdm_sensor_config_get_label (apdm_device_handle_t *device_handle*, char * *BYTE*, const int *buff_size*)

Wrapper function for getting the label config parameter

Parameters

device_handle The device handle.

BYTE 16 character array destination to store the label.

The size of the buffer into which to store the label, must be at least 16

Returns

APDM_OK on success, error code otherwise.

References apdm_log_debug(), apdm_log_error(), and apdm_sensor_cmd_-config_get().

Referenced by apdm_sensor_populate_device_info().

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

Wrapper function for setting the label config parameter

Parameters

device_handle The device handle.

lable_str 16 character array source for label.

Returns

APDM_OK on success, error code otherwise.

References apdm_log_debug(), and apdm_sensor_cmd_config_set().

5.7.1.77 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 seting 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().

5.7.1.78 APDM_EXPORT int apdm_sensor_get_device_id_list (uint32_t * *serial_number_buffer*, const uint32_t *buffer_length*)

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

Parameters

∗***serial_number_buffer*** Pointer to an array of uint32_t into whichthe serial numbers should be stored.

buffer_length The number of elements in the destination array.

Returns

APDM_OK on success, error code otherwise.

References apdm_log_info(), apdm_sensor_close(), apdm_sensor_cmd_-device_id(), and apdm_sensor_open().

5.7.1.79 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 otherwise.

5.7.1.80 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.

Returns

APDM_OK on success, error code otherwise.

References apdm_get_time_ms_64(), apdm_log_debug(), 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_version_string_1(), apdm_sensor_cmd_version_string_2(), apdm_sensor_cmd_version_string_3(), apdm_sensor_config_get_label(), 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_channel0, 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.

5.8 DockingStation

Functions

- 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_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, const int dest_buffer_length)
- APDM_EXPORT int apdm_ds_get_serial_number_by_index (const int 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)

5.8.1 Function Documentation

5.8.1.1 APDM_EXPORT int apdm_ds_get_case_id (apdm_device_handle_t *device_handle*, char * *BYTE*, const int *dest_buffer_length*)

Parameters

device__handle The docking station handle

dest_buffer Destination into which to store the Case ID String

buffer_length The length of the buffer to which dest_buffer is pointing

Returns

APDM_OK on success, error code otherwise.

References apdm_log_error().

5.8.1.2 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 stationhandle

dest Destination into which you want the module stored into.

Returns

APDM_OK on success, error code otherwise

5.8.1.3 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 otherwise.

5.8.1.4 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 desination into which to store the hardware revision number of the dock

Returns

APDM_OK on success, error code otherwise.

Referenced by apdm_configure_all_attached_sensors_mesh().

5.8.1.5 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 otherwise.

References apdm_ds_get_serial_number_by_index().

5.8.1.6 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 otherwise

References apdm_log_warning().

Referenced by apdm_configure_all_attached_sensors_mesh().

5.8.1.7 APDM_EXPORT int apdm_ds_get_serial_number_by_index (const int *index*, uint32_t * *serial_number*)

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

Parameters

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 otherwise.

Referenced by apdm_ds_get_index_by_serial_number().

5.8.1.8 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 otherwise.

References apdm_log_error().

5.8.1.9 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.

5.8.1.10 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

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

Returns

APDM_OK on success, error code otherwise

5.9 Logging

Functions

- APDM_EXPORT int set_apdm_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,...)

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

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

5.9.1.3 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

Referenced by apdm_ap_get_id(), apdm_configure_all_attached_sensors_mesh(), apdm_ctx_get_next_access_point_record(), apdm_ctx_get_wireless_streaming_status(), apdm_ctx_sync_record_list_head(), apdm_extract_next_sample_set(), apdm_halt_all_attached_sensors(), apdm_open_all_access_points(), apdm_process_raw(), apdm_read_hdf_dataset(), apdm_read_hdf_timestamps(), apdm_read_raw_file_info(), apdm_sensor_allocate_handle(), apdm_sensor_cmd_case_id(), apdm_sensor_cmd_time_get(), apdm_sensor_cmd_time_set(), apdm_sensor_config_get_label(), apdm_sensor_config_set_label(), apdm_sensor_free_handle(), apdm_sensor_populate_device_info(), and apdm_write_record_hdf().

5.9.1.4 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

Referenced by apdm_ap_get_case_id(), apdm_ap_verify_supported_version(), apdm_configure_accesspoint(), apdm_configure_all_attached_sensors_mesh(), apdm_ctx_allocate_new_context(), apdm_ctx_free_context(), apdm_ctx_get_next_access_point_record(), apdm_ctx_get_next_access_point_record_list(), apdm_ctx_get_next_record(), apdm_ctx_get_sensor_compensation_data(), apdm_ctx_is_more_data_immediately_available(), apdm_ctx_set_sensor_compensation_data(), apdm_ctx_sync_record_list_head(), apdm_ds_get_case_id(), apdm_ds_is_monitor_data_

forwarding_enabled(), apdm_extract_next_sample_set(), apdm_halt_-all_attached_sensors(), apdm_open_all_access_points(), apdm_send_-accesspoint_cmd(), and apdm_sensor_config_get_label().

5.9.1.5 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

Referenced by apdm_autoconfigure_mesh_sync(), apdm_configure_-accesspoint(), apdm_configure_all_attached_sensors_mesh(), apdm_-halt_all_attached_sensors(), apdm_open_all_access_points(), and apdm_-sensor_get_device_id_list().

5.9.1.6 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

Referenced by apdm_ap_get_case_id(), apdm_configure_all_attached_-sensors_mesh(), apdm_ds_get_serial(), apdm_extract_next_sample_set(), and apdm_free_ap_handle().

5.9.1.7 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

5.9.1.8 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

5.9.1.9 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

filePath The file to which logging data should be saved

Returns

APDM_OK on success.

References apdm_close_log_file().

5.9.1.10 APDM_EXPORT int set_apdm_log_level (int *log_level*)

Sets the current log level to be used by APDM libraries. Valid values are:

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.

Chapter 6

Class Documentation

6.1 __attribute__ Struct Reference

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**
- int64_t **cal_data_2**
- union {
  int64_t **calibration_version**
  apdm_calibration_data_v4_t **v4**
  apdm_calibration_data_v5_t **v5**
  } **data**

- uint32_t **cal_version**
- uint32_t **device_id**
- uint8_t **retrys**
- uint16_t **event_id**

- union {
  struct {
  uint32_t **data_a**
  uint32_t **data_b**
  uint32_t **data_c**
  uint32_t **data_d**
  uint32_t **data_e**
  uint32_t **data_f**
  } **raw**
  struct {
  uint32_t **error_id**
  uint32_t **error_count**
  uint32_t **sync_low_32**
  uint32_t **sync_high_32**
  } **error_event**
  struct {
  uint32_t **pre_block**
  uint32_t **post_block**
  uint32_t **samples_per_block**
  uint32_t **max_latency**
  } **latency_event**
  struct {
  uint64_t **pre_sync**
  uint64_t **post_sync**
  } **sync_event**
  } **packet**

- uint16_t **flags**
- 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 **mc**
- uint32_t **opt_data**
- uint32_t **sync_val_32**
- uint8_t **ap_flags**
- union {
  uint8_t **data_raw_array** [63]
  opal_data_t **opal_data**
  struct {

```
    uint8_t gpio_data
    uint8_t data_type
    uint8_t gpio_change_mask
    uint64_t sync_value
    uint16_t adc_read_value1
    uint16_t adc_read_value2
    uint8_t uart_character
    uint8_t value_population_populated_flags
  } external_sync_data
  struct {
    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
  } opal_event_data
  struct {
    uint8_t generic_packet_type
    union {
      opal_data_t opal_data
      struct {
        union {
          uint8_t raw_packet [32]
          uint8_t type
          struct {
            uint8_t type
            uint8_t retrys
            uint16_t flags
            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 mc
            uint32_t opt_data
            uint32_t sync_val_32
          } opal_data
          struct {
            uint8_t type
```

```
            uint8_t gpio_data
            uint8_t data_type
            uint8_t gpio_change_mask
            uint64_t sync_value
            uint16_t adc_read_value1
            uint16_t adc_read_value2
            uint8_t uart_character
            uint8_t value_population_populated_flags
         } external_sync_data
         struct {
            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
         } opal_event_data
      } packet
      uint8_t padding [2]
      uint64_t sync_value_at_time_of_arrivial
   } promisc_packet
   struct {
      uint8_t event_type
      union {
         struct {
            uint8_t nrf_status_register
            uint64_t sync_a
            uint64_t sync_b
            uint64_t ap_sync_value
            uint32_t device_id
            uint8_t nrf_fifo_status_register
            uint8_t nrf_fifo_status_register2
            uint16_t currentSampleCount
            uint8_t currentSampleCountMod24
            uint8_t currentSampleCountMod32
         } sync_delta
         struct {
            uint8_t nrf_status_register
            uint64_t ap_sync_value
            uint64_t packet_sync64
            uint32_t device_id
            uint8_t packet_data [32]
            uint8_t nrf_fifo_status_register
```

} **invalid_packet_cksum**
} **event_data**
} **ap_event**
} **payload**
} **generic_packet_data**
} **packet_data**

6.1.1 Member Data Documentation

6.1.1.1 uint16_t __attribute__::adc_read_value1

The point in time at which the gpio data was with respect to.

6.1.1.2 uint8_t __attribute__::data_type

Corresponds to the current GPIO input/output values of the given GPIO line

6.1.1.3 uint64_t __attribute__::sync_value

For each bit in gpio_data that was changed, the coresponding bit in this variable will be 1

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

- apdm_internal.h
- apdm_l1_ap.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**
- uint64_t **sync_value_subtractor**
- 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**
- 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

- uint32_t **byte_data_length**
- uint32_t **byte_current_index**
- char **byte_stream_buffer** [8192]

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

- apdm_internal.h

6.6 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

6.7 apdm_context_t Struct Reference

`#include <apdm_internal.h>`

Public Attributes

- 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**
- 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**

6.7.1 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

6.8 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]

6.8.1 Member Data Documentation

6.8.1.1 double apdm_device_dtemp_filter_state_t::error_covariance_matrix[4]

6.8.1.2 double apdm_device_dtemp_filter_state_t::filtered_measurement[1]

6.8.1.3 double apdm_device_dtemp_filter_state_t::measurement[1]

6.8.1.4 double apdm_device_dtemp_filter_state_t::measurement_matrix[2]

initialize to [0;0;0;0;0;0;0];

6.8.1.5 double apdm_device_dtemp_filter_state_t::measurement_noise_matrix[1]

6.8.1.6 double apdm_device_dtemp_filter_state_t::process_noise_matrix[4]

initialize to [0 0; 0 1]

6.8.1.7 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

6.9 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

6.10 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 **spin_off_enable**
- 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]
- uint8_t **calibration_binary_blob** [CALIBRATION_DATA_BUFFER_SIZE]
- uint32_t **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]
- char **firmware_version_string3** [VERSION_STRING_SIZE]
- char **case_id** [CASE_ID_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
- 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**

6.10.1 Detailed Description

6.10.2 Member Data Documentation

6.10.2.1 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_ctx_get_next_access_point_record(), and apdm_initialize_device_info().

6.10.2.2 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_configure_all_attached_sensors_mesh().

6.10.2.3 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()

6.10.2.4 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_configure_all_attached_sensors_mesh(), and apdm_sensor_populate_device_info().

6.10.2.5 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_sensor_populate_device_info().

6.10.2.6 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_sensor_populate_device_info().

6.10.2.7 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_configure_all_attached_sensors_mesh(), and apdm_sensor_populate_device_info().

6.10.2.8 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_sensor_populate_device_info().

6.10.2.9 uint8_t apdm_device_info_t::wireless_channel0

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_sensor_populate_device_info().

6.10.2.10 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_sensor_populate_device_info().

6.10.2.11 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_configure_all_attached_sensors_mesh(), and apdm_sensor_populate_device_info().

6.10.2.12 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_sensor_populate_device_info().

6.10.2.13 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_configure_all_attached_sensors_mesh(), and apdm_sensor_populate_device_info().

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

- apdm_types.h

6.11 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

6.12 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** [7]
- double **mag_y_buffer** [7]
- double **mag_z_buffer** [7]
- int **mag_buffer_index**
- 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

6.13 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

6.14 apdm_external_sync_data_t Struct Reference

Public Attributes

- uint8_t **data**
- uint8_t **data_type**
- uint64_t **sync_value**

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

- apdm_types.h

6.15 apdm_mag_dechop_state_t Struct Reference

Public Attributes

- double **state** [2]
- double state_transition_matrix [4]
- double process_noise_matrix [4]
- double measurement [5]
- double measurement_matrix [10]
- double measurement_noise_matrix [5 ∗5]
- double error_covariance_matrix [4]
- double filtered_measurement [5]
- double stepResponse [10]
- int set_reset_flag
- int polarity
- int iSample

6.15.1 Member Data Documentation

6.15.1.1 double apdm_mag_dechop_state_t::error_covariance_matrix[4]

6.15.1.2 double apdm_mag_dechop_state_t::filtered_measurement[5]

6.15.1.3 int apdm_mag_dechop_state_t::iSample

6.15.1.4 double apdm_mag_dechop_state_t::measurement[5]

6.15.1.5 double apdm_mag_dechop_state_t::measurement_matrix[10]

initialize to [0;0;0;0;0;0;0;0];

6.15.1.6 double apdm_mag_dechop_state_t::measurement_noise_matrix[5 ∗5]

6.15.1.7 int apdm_mag_dechop_state_t::polarity

6.15.1.8 double apdm_mag_dechop_state_t::process_noise_matrix[4]

initialize to [0 0; 0 1]

6.15.1.9 int apdm_mag_dechop_state_t::set_reset_flag

6.15.1.10 double apdm_mag_dechop_state_t::state_transition_-matrix[4]

initialize to [0 0];

6.15.1.11 double apdm_mag_dechop_state_t::stepResponse[10]

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

- apdm_types.h

6.16 apdm_monitor_error_stat_t Struct Reference

Public Attributes

- apdm_monitor_error_id_t **error_id**
- uint32_t **error_count**
- uint64_t **sync_value**

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

- apdm_types.h

6.17 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

6.18 apdm_progress_t Struct Reference

Public Attributes

- char **task** [128]
- double **percent_complete**

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

- apdm_types.h

6.19 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

6.20 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**
- uint8_t **flag_tag_data**
- uint16_t **flags**
- bool gyro_temperature_sensor_selected
- uint32_t **optional_data**
- uint8_t opt_select
- 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 events_num_events
- uint8_t **events_event_list** [MAX_APDM_EVENTS]
- 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 **temperature_si**
- double temperature_derivative_si
- double battery_level

6.20.1 Detailed Description

APDM Sensor Sample

6.20.2 Member Data Documentation

6.20.2.1 bool apdm_record_t::accl_full_scale_mode

raw ADC readings

6.20.2.2 bool apdm_record_t::accl_isPopulated

True indicates accelerometers are in 6G mode, false indicates 2G mode

6.20.2.3 uint16_t apdm_record_t::accl_x_axis

Index of the AP that the sample came in on.

Referenced by apdm_extract_next_sample_set(), and apdm_write_record_hdf().

6.20.2.4 uint16_t apdm_record_t::accl_y_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.5 double apdm_record_t::accl_y_axis_si

meters per second^2

Referenced by apdm_write_record_hdf().

6.20.2.6 uint16_t apdm_record_t::accl_z_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.7 double apdm_record_t::accl_z_axis_si

meters per second^2

Referenced by apdm_write_record_hdf().

6.20.2.8 bool apdm_record_t::batt_voltage_isPopulated

raw ADC readings

6.20.2.9 double apdm_record_t::battery_level

degrees C per sec

6.20.2.10 bool apdm_record_t::device_info_isPopulated

6.20.2.11 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().

6.20.2.12 uint8_t apdm_record_t::device_info_wireless_address

6.20.2.13 uint8_t apdm_record_t::device_info_wireless_channel_id

Device ID

6.20.2.14 uint8_t apdm_record_t::events_num_events

6.20.2.15 uint8_t apdm_record_t::flag_accel_enabled

Indicates that the mag data is populated

6.20.2.16 bool apdm_record_t::gyro_isPopulated

raw ADC readings

6.20.2.17 bool apdm_record_t::gyro_temperature_sensor_selected

Flags packed binary structure, used to derive the flag_XXXX field values.

6.20.2.18 uint16_t apdm_record_t::gyro_x_axis

Indicates that the accel data is populated

Referenced by apdm_write_record_hdf().

6.20.2.19 double apdm_record_t::gyro_x_axis_si

meters per second^2

Referenced by apdm_write_record_hdf().

6.20.2.20 uint16_t apdm_record_t::gyro_y_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.21 double apdm_record_t::gyro_y_axis_si

radians per second

Referenced by apdm_write_record_hdf().

6.20.2.22 uint16_t apdm_record_t::gyro_z_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.23 double apdm_record_t::gyro_z_axis_si

radians per second

Referenced by apdm_write_record_hdf().

6.20.2.24 uint16_t apdm_record_t::mag_common_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.25 bool apdm_record_t::mag_isPopulated

raw ADC readings

6.20.2.26 uint16_t apdm_record_t::mag_x_axis

Indicates that the gyro data is populated

Referenced by apdm_write_record_hdf().

6.20.2.27 double apdm_record_t::mag_x_axis_si

radians per second

Referenced by apdm_write_record_hdf().

6.20.2.28 uint16_t apdm_record_t::mag_y_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.29 uint16_t apdm_record_t::mag_z_axis

raw ADC readings

Referenced by apdm_write_record_hdf().

6.20.2.30 uint8_t apdm_record_t::num_retrys

Internal use only

Referenced by apdm_ctx_get_next_access_point_record().

6.20.2.31 uint8_t apdm_record_t::opt_select

Optional and varying data from the monitor

Referenced by apdm_ctx_get_next_access_point_record().

6.20.2.32 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().

6.20.2.33 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().

6.20.2.34 double apdm_record_t::temperature

Indicates what type of data is in optional_data, see enum apdm_raw_opt_-select_t

Referenced by apdm_ctx_get_next_access_point_record().

6.20.2.35 double apdm_record_t::temperature_derivative_si

degrees celcius

Referenced by apdm_write_record_hdf().

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

- apdm_types.h

6.21 apdm_recording_info_t Struct Reference

Public Attributes

- apdm_device_info_t **device_info**
- uint64_t **start_sync_count**
- uint64_t **end_sync_count**

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

- apdm_types.h

6.22 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

6.23 apdm_sensor_compensation_t Struct Reference

Public Attributes

- uint32_t **converted_calibration_version**
- uint32_t **raw_calibration_version**
- union {
  calibration_v4_t **v4**
  } **data**

6.23.1 Detailed Description

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

- apdm_types.h

6.24 apdm_sensor_device_handle_t Struct Reference

`#include <apdm_internal.h>`

Public Attributes

- FT_HANDLE **ft_handle232**
- 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**

6.24.1 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

6.25 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 **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**
apdm_calibration_data_t **calibration_data**
uint32_t **hw_id**
char **case_id** [16]
} **response_data**

- uint16_t **crc16**

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

- apdm_internal.h

6.26 cal_info_t Struct Reference

Public Attributes

- float **acc_error_matrix** [9]
- float gyro_error_matrix [9]
- float acc_bias [3]
- float acc_scale [3]
- float acc_bias_temp [3]
- float acc_scale_temp [3]
- float acc_crossaxis_sensitivity [3]
- float gyro_bias [3]
- float gyro_scale [3]
- float gyro_bias_temp [3]
- float gyro_scale_temp [3]
- float gyro_crossaxis_sensitivity [3]
- float gyro_misalignment [3]
- float mag_bias [3]
- float mag_bias_temp [3]
- float mag_error_matrix [9]
- float temp_bias
- float temp_scale

6.26.1 Detailed Description

6.26.2 Member Data Documentation

6.26.2.1 float cal_info_t::acc_bias[3]

6.26.2.2 float cal_info_t::acc_bias_temp[3]

6.26.2.3 float cal_info_t::acc_crossaxis_sensitivity[3]

6.26.2.4 float cal_info_t::acc_scale[3]

6.26.2.5 float cal_info_t::acc_scale_temp[3]

6.26.2.6 float cal_info_t::gyro_bias[3]

6.26.2.7 float cal_info_t::gyro_bias_temp[3]

6.26.2.8 float cal_info_t::gyro_crossaxis_sensitivity[3]

6.26.2.9 float cal_info_t::gyro_error_matrix[9]

6.26.2.10 float cal_info_t::gyro_misalignment[3]

6.26.2.11 float cal_info_t::gyro_scale[3]

6.26.2.12 float cal_info_t::gyro_scale_temp[3]

6.26.2.13 float cal_info_t::mag_bias[3]

6.26.2.14 float cal_info_t::mag_bias_temp[3]

6.26.2.15 float cal_info_t::mag_error_matrix[9]

6.26.2.16 float cal_info_t::temp_bias

6.26.2.17 float cal_info_t::temp_scale

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

- apdm_types.h

6.27 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

6.27.1 Detailed Description

6.27.2 Member Data Documentation

6.27.2.1 double calibration_v4_t::accl_error_matrix[3 ∗3]

6.27.2.2 double calibration_v4_t::accl_x_bias_temp

6.27.2.3 double calibration_v4_t::accl_x_scale

6.27.2.4 double calibration_v4_t::accl_x_scale_temp

6.27.2.5 double calibration_v4_t::accl_xy_sensitivity

6.27.2.6 double calibration_v4_t::accl_xz_sensitivity

6.27.2.7 double calibration_v4_t::accl_y_bias

6.27.2.8 double calibration_v4_t::accl_y_bias_temp

6.27.2.9 double calibration_v4_t::accl_y_scale

6.27.2.10 double calibration_v4_t::accl_y_scale_temp

6.27.2.11 double calibration_v4_t::accl_yz_sensitivity

6.27.2.12 double calibration_v4_t::accl_z_bias

6.27.2.13 double calibration_v4_t::accl_z_bias_dtemp

6.27.2.14 double calibration_v4_t::accl_z_bias_temp

6.27.2.15 double calibration_v4_t::accl_z_scale

6.27.2.16 double calibration_v4_t::accl_z_scale_temp

6.27.2.17 double calibration_v4_t::gyro_accl_pitch

6.27.2.18 double calibration_v4_t::gyro_accl_roll

6.27.2.19 double calibration_v4_t::gyro_accl_yaw

6.27.2.20 double calibration_v4_t::gyro_error_matrix[3 ∗3]

6.27.2.21 double calibration_v4_t::gyro_x_bias_temp

6.27.2.22 double calibration_v4_t::gyro_x_bias_temp2

6.27.2.23 double calibration_v4_t::gyro_x_scale

6.27.2.24 double calibration_v4_t::gyro_x_scale_temp

- apdm_types.h

6.28 calibration_v5_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

6.28.1 Member Data Documentation

6.28.1.1 double calibration_v5_t::accl_error_matrix[3 ∗3]

6.28.1.2 double calibration_v5_t::accl_x_bias_temp

6.28.1.3 double calibration_v5_t::accl_x_scale

6.28.1.4 double calibration_v5_t::accl_x_scale_temp

6.28.1.5 double calibration_v5_t::accl_xy_sensitivity

6.28.1.6 double calibration_v5_t::accl_xz_sensitivity

6.28.1.7 double calibration_v5_t::accl_y_bias

6.28.1.8 double calibration_v5_t::accl_y_bias_temp

6.28.1.9 double calibration_v5_t::accl_y_scale

6.28.1.10 double calibration_v5_t::accl_y_scale_temp

6.28.1.11 double calibration_v5_t::accl_yz_sensitivity

6.28.1.12 double calibration_v5_t::accl_z_bias

6.28.1.13 double calibration_v5_t::accl_z_bias_dtemp

6.28.1.14 double calibration_v5_t::accl_z_bias_temp

6.28.1.15 double calibration_v5_t::accl_z_scale

6.28.1.16 double calibration_v5_t::accl_z_scale_temp

6.28.1.17 double calibration_v5_t::gyro_accl_pitch

6.28.1.18 double calibration_v5_t::gyro_accl_roll

6.28.1.19 double calibration_v5_t::gyro_accl_yaw

6.28.1.20 double calibration_v5_t::gyro_error_matrix[3 ∗3]

6.28.1.21 double calibration_v5_t::gyro_x_bias_temp

6.28.1.22 double calibration_v5_t::gyro_x_bias_temp2

6.28.1.23 double calibration_v5_t::gyro_x_scale

6.28.1.24 double calibration_v5_t::gyro_x_scale_temp

6.28.1.25 double calibration_v5_t::gyro_xy_sensitivity

6.28.1.26 double calibration_v5_t::gyro_xz_sensitivity

- apdm_types.h

6.29 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

6.30 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

6.31 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

6.32 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

6.33 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

6.34 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

6.35 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

6.36 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

6.37 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